NATAL, 2018

Realização: SBC

Organização: UFRN

anais 2018

XXXVIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

26º WEI – WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO

CENTRO DE CONVENÇÕES | NATAL•RN | 22 A 26 DE JULHO DE 2018

#COMPUTAÇÃOESUSTENTABILIDADE

CSBC 2018

anais 2018

XXXVIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO
CENTRO DE CONVENÇÕES | NATAL•RN | 22 A 26 DE JULHO DE 2018
#COMPUTAÇÃOESUSTENTABILIDADE

Coordenador Geral

Francisco Dantas de Medeiros Neto (UERN)

Comissão Organizadora

Bartira Paraguaçu Falcão Dantas Rocha (UERN)
Camila Araújo Sena (UERN)
Everton Ranielly de Sousa Cavalcante (UFRN)
Felipe Torres Leite (UFERSA)
Ilana Albuquerque (UERN)
Isaac de Lima Oliveira Filho (UERN)
Priscila Nogueira Krüger (UERN)

Realização

Sociedade Brasileira de Computação

Organização

Universidade do Estado do Rio Grande do Norte

CSBC 2018
XXXVIII Congresso da
Sociedade Brasileira de Computação

Apresentação

Estes anais registram os trabalhos apresentados durante o XXXVIII Congresso da Sociedade Brasileira de Computação (CSBC 2018), realizado em Natal-RN, de 22 a 26 de julho 2018. O evento teve como tema central a Computação e Sustentabilidade, pois se compreende que o avanço da computação e as questões ambientais devem caminhar lado-a-lado, tendo em vista que as técnicas computacionais necessitam ser usadas para possibilitar o desenvolvimento sustentável, e, desse modo, equilibrar as necessidades ambientais, econômicas e sociais.

Organizar o maior evento acadêmico de Computação da América Latina foi um privilégio e um desafio. Foi enriquecedor promover e incentivar a troca de experiências entre estudantes, professores, profissionais, pesquisadores e entusiastas da área de Computação e Informática de todo o Brasil. Ao mesmo foi desafiador termos que lidar, principalmente, com às dificuldades impostas pelo momento de crise que o nosso Brasil vem enfrentando. Uma crise que afeta diretamente nossas pesquisas e, consequentemente, o desenvolvimento e inovação do nosso amado Brasil.

Por meio de seus 25 eventos, o CSBC 2018 apresentou mais de 300 trabalhos, várias palestras e mesas-redondas. O Congresso ainda abrigou diversas reuniões, que incluem a reunião do Fórum de Pós-Graduação, a reunião do CNPq/CAPES, a reunião dos Secretários Regionais SBC, a reunião das Comissões Especiais e a reunião do Fórum IFIP/SBC.

O sucesso do CSBC 2018 só foi possível devido à dedicação e entusiasmo de muitas pessoas. Gostaríamos de agradecer aos coordenadores dos 25 eventos e aos autores pelo envio de seus trabalhos. Além disso, gostaríamos de expressar nossa gratidão ao Comitê Organizador, por sua grande ajuda em dar forma ao evento; e, em especial, à equipe da Sociedade Brasileira de Computação (SBC), por todo apoio.

Por fim, reconhecemos a importância do apoio financeiro da CAPES, do CNPq, do CGI.br, do Governo do Estado do Rio Grande do Norte, da Prefeitura Municipal do Natal, da Prefeitura Municipal de Parnamirim, da CABO Telecom, da ESIG Software e Consultoria, da DynaVideo e do SENAI.

Natal (RN), 26 de julho de 2018.

Chico Dantas (UERN)
Coordenador Geral do CSBC 2018

Anais do CSBC 2018

26º WEI – WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO

XXVI WEI
WORKSHOP SOBRE
EDUCAÇÃO EM COMPUTAÇÃO

Apresentação

O WEI, Workshop sobre Educação em Computação, promovido pela SBC, é o mais importante fórum brasileiro de discussão sobre problemas e temas relacionados à pesquisa, às iniciativas e à avaliação da educação na área de Computação. O fórum discute diferentes temas relacionados à área, apresentando novos estudos, metodologias, ferramentas e discutindo experiências vividas em todo o Brasil.

Em sua 26ª edição, o WEI traz a discussão do ensino de Computação no contexto do tema "Educação para uma sociedade sustentável". As atividades previstas para o WEI têm o intuito de promover ampla discussão sobre o tema, compreendendo apresentação de artigos, palestras convidadas e os minicursos, a grande novidade nesta edição. Os minicursos abordarão os seguintes temas: Computação na Educação Básica, Pesquisa em Educação e Classificação dos Cursos de Graduação em Computação.

O XXVI WEI recebeu 185 submissões de trabalhos, entre artigos de pesquisa, ensaios, relatos de experiências e descrição de ferramentas ou recursos educacionais. Dentre estes, foram aceitos 52 trabalhos, sendo 16 artigos de pesquisa, 23 artigos de intervenção ou relato de experiências, 10 artigos de ferramentas ou recursos educacionais, e 03 ensaios. Para o Comitê de Programa, quatro destes trabalhos se destacaram e concorrerão, durante o XXXVIII CSBC, ao prêmio de Best Paper WEI 2018.

Aos autores, palestrantes, organizadores de minicursos e demais membros da comunidade que contribuíram para a realização de mais uma edição do workshop, expressamos nosso agradecimento especial pelo reconhecimento e confiança no evento. Nosso agradecimento e gratidão ao Comitê de Programa e aos revisores convidados, pelo excelente trabalho de avaliação de artigos, e pela disponibilidade dedicada ao WEI, pois sem eles o WEI não aconteceria.

Esperamos que os trabalhos que compõem este XXVI WEI inspirem novos estudos e intervenções no campo da educação em Computação. A todos os participantes nossas boas vindas e o desejo de que todos possam aproveitar ao máximo os eventos que compõem esta edição do Workshop sobre Educação em Computação!

Leandro Galvão (UFAM)
Jair Leite (UFRN)
Coordenadores do WEI 2018

Coordenação Geral

- Leandro Silva Galvão de Carvalho (UFAM)
- Jair Cavalcanti Leite (UFRN)

Coordenação Local

- Rommel Lima (UERN)

Comitê de Programa

- Abraham de Sousa (UFOPA)
- Adao Cambraia (IFFar)
- Adja Andrade (UFRN)
- Adriano Santos (Unifacisa)
- Alan Henrique Pardo de Carvalho (Fatec São Caetano do Sul)
- Alberto Castro (UFAM)
- Alessandro Cerqueira (CRF-RJ)
- Alexandre Cidral (Univille)
- Alexandre Passito (UFAM)
- Ana Paula Serra (USJT)
- André Campos (UFRN)
- André Drummond (UnB)
- André Lemos (IFTM)
- Anna Beatriz Marques (UFC)
- Anna Schwarzelmuller (UFBA)
- Antonio Carlos Beck F. (UFRGS)
- Antonio Carlos Fernandes da Silva (UTFPR)
- Ayla Débora Dantas de Souza Rebouças (UFPB)
- Carina F. Dorneles (UFSC)
- Carla Delgado (UFRJ)
- Carolina Moreira Oliveira (UFPR)
- Celso Olivete (UNESP)
- Cesar França (UFPE)
- Charles Madeira (UFRN)
- Cicilia Maia (UERN)
- Clovis Fernandes (ITA)
- Cristiano Maciel (UFMT)
- Daltro Nunes (UFRGS)
- Daniela Barreiro Claro (UFBA)
- Danielle Rousy Silva (CI-UFPB)
- Davi Viana (UFMA)
- David Fernandes (UFAM)
- Débora Abdalla (UFBA)
- Denise Bandeira (UNISINOS)
- Ecivaldo Matos (UFBA)
- Edeilson Milhomem Silva (CEULP/ULBRA)
- Edmundo Spoto (UFG)

- Edson Pimentel (UFABC)
- Eduardo Barrére (UFJF)
- Eduardo Figueiredo (UFMG)
- Elaine Oliveira (UFAM)
- Eleandro Maschio (UTFPR)
- Fernando Moraes (PUCRS)
- Francisco Mendes Neto (UFERSA)
- Francisco Zampirolli (UFABC)
- Isabel Nunes (UFRN)
- Isabela Gasparini (UDESC)
- Jair Leite (UFRN)
- José Maria David (UFJF)
- José Palazzo Moreira de Oliveira (UFRGS)
- Juliana Diniz (UFRPE)
- Leandro Galvão (UFAM)
- Lucia Giraffa (PUC-RS)
- Luis Rivero (UFAM)
- Marcelo Duduchi (FATEC-SP / CEETEPS)
- Marcelo Yamaguti (PUCRS)
- Marcia Kniphoff da Cruz (UNISC)
- Márcio Cornélio (UFPE)
- Marcos Borges (Unicamp)
- Maria Augusta Vieira Nelson (PUC Minas)
- Maria de Fátima do Prado Lima (UCS)
- Max Machado (PUC Minas)
- Mirella Moro (UFMG)
- Moisés Carvalho (UFAM)
- Monael Ribeiro (UFABC)
- Morganna Diniz (UNIRIO)
- Natasha Valentim (UFPR)
- Noemi Rodriguez (PUC-Rio)
- Paulo Maia (UECE)
- Raquel Prates (UFMG)
- Renata Araujo (UNIRIO)
- Ricardo Caceffo (UNICAMP)
- Roberto Bittencourt (UEFS)
- Rodrigo de Souza (UFRPE)
- Rommel Lima (UERN)
- Ronaldo Correia (UNESP)
- Ronaldo Mello (UFSC)
- Sandro Rigo (Unisinos)
- Silvia Amelia Bim (UTFPR)
- Simone Martins (UFF)
- Sonia França (UFRPE)
- Taciana Pontual Falcão (UFRPE)
- Tanara Lauschner (UFAM)
- Vitor Bremgartner (IFAM)
- Wilson Veneziano (UnB)

- Yuska Paola Costa Aguiar (UFPB)

Revisores Convidados

- Adriano dos Santos (UFMG)
- Adriano César Pereira (UFMG)
- Alana Oliveira (UFMA)
- Alex Gomes (UFPE)
- Ana Paula Kuhn (UFMT)
- Clodoveu Davis (UFMG)
- Diego Zabot (UFBA)
- Eliomar Lima (UFG)
- Elisa de Fátima Soares (UERN)
- Felipe Cordeiro (UNIRIO)
- Fernando Federson (UFG)
- Fischer Jonatas (UFMG)
- Hercules Sandim (UFMS)
- Isabel Rosseti (UFF)
- Josualdo Dias (UFBA)
- Karen Figueiredo (UFF)
- Kattiana Constantino (UFMG)
- Leonardo Bandeira de Lucena (UERN)
- Leonardo Murta (UFF)
- Ligia Maria de Sousa Dantas Batista (UERN)
- Luis Kowada (UFF)
- Luiz Gonçalves (UFRN)
- Maurício Souza (UFMG)
- Mônica do Amaral (UFOP)
- Milene Silveira (PUCRS)
- Oreste Preti (UFMT)
- Patricia Amorim (UNIRIO)
- Sabrina Sassi (UFMT)
- Tadeu Classe (UNIRIO)

SUMÁRIO

Avaliação de uma Dinâmica Vivencial para o Ensino de Gerenciamento de Projetos em Cursos de Computação 14
Giani Petri, Christiane Gresse von Wangenheim, Bruno Boniati, Alex Weber

Benefícios dos Jogos Não-Digitais no Ensino de Computação 25
Giani Petri, Alejandro Calderón, Christiane Gresse von Wangenheim, Adriano Borgatto, Mercedes Ruiz

Análise da Evasão de Alunos na Licenciatura em Computação 41
Viviane Vasconcelos, Ermeson Andrade

CriptoLab: Um game baseado em Computação Desplugada e Criptografia 49
Graziela Guarda, Débora Juliane Silva, Ione Goulart

Cognition Developing of Computer Higher Education Students Through Gamification in the Algorithm Teaching-Learning Process 59
Tiago Nogueira, Deller Ferreira, Eude de Souza Campos

Grupo de Estudo, Pesquisa e Extensão em Robótica e Automação Como Fator Motivacional Para Estudantes de Computação 72
Matheus Eloy Franco, Breno Barra, Rosana Moreira, Caio Dias

Produção de jogos digitais educacionais por alunos do ensino superior: um relato de experiência 82
Kleber Fernandes, Eduardo Aranha, Márcia Lucena

Validação de Modelos ER 92
Cody Malnor, André Chateaubriand, Obede Carvalho, Ricardo Terra

Identificação Automática de Estilos de Aprendizagem: Uma Revisão Sistemática da Literatura 102
Edilaine Oliveira, Gilvandenys Sales, Pryscilla Pereira, Ramires Moreira

Experiência de uso de Caixas de Ovos no Apoio ao Ensino de Vetores e Matrizes 115
Rita Berardi, Regiane Macuch, Letícia Dal Forno, Silvia Amelia Bim

Percepção de estudantes sobre motivação e aprendizagem em Teoria da Computação com PBL 125
Luiz Gavaza, Lais Nascimento Salvador, David Moises Barreto dos Santos

Uso do Scratch na Introdução de Conceitos de Lógica de Programação: relato de experiência 135
Carina Farias, Anderson Oliveira, Everton Dias de Almeida Silva

Análise da Motivação em um Estudo Integrado de Programação Baseado em PBL 145
Ayala Lemos Ribeiro, Roberto Bittencourt, Bianca Santana

Uma Abordagem Gamificada para o Ensino de Lógica de Programação: relato de experiência 161
Carina Farias, Fellipe Pereira Azevedo, José Elias de Jesus Dias

SoccerCraft: Relato de Atividade para Ensino Aprendizagem de Habilidades do Pensamento Computacional Aplicada no Sexto Ano do Ensino Fundamental 171
Simão Martin, Simone Cavalheiro, Luciana Foss, Renata Reiser, Ana Rita Mazzini, Andre Du Bois, Clause Piana

Um Simulador Educacional para Apoio ao Projeto de Sistemas Computacionais: Hardware, Software e suas Interfaces 181
Guilherme Álvaro Esmeraldo, Edson Lisboa, Cicero Samuel Rodrigues Mendes, Lucas Fontes Cartaxo

Um ensaio sobre a experiência educacional na programação de computadores: a abordagem tradicional versus a aprendizagem baseada em projetos 191
Alexandre Grotta, Edmir Prado

An Interdisciplinary Approach to Software Engineering Teaching: An Experience Report 201
Gláucia Braga e Silva, Daniel Barbosa, Fabrício Silva

TuPy Online - Programação em Português com Visualização de Execução e Abstrações de Estruturas de Dados na Web 211
Giancarlo Roberto, Fabiano Oliveira, Paulo Pinto, Igor Machado Coelho

Performance Analysis of Computer Science Students in Programming Learning 221
Air Rabelo, Luiz Maia, Fernando Silva Parreiras

Formação Docente na Pós-Graduação Stricto Sensu em Ciência da Computação: um recorte das regiões Norte e Nordeste 231
Paulenay Morais, Jean Rosa, Anna Raquel da Silva Marinho, Ecivaldo Matos

Programação para Administração de Redes de Computadores - Uma Experiência com Estudantes de Computação 246
Silmar Antonio de Oliveira, Andréa Mendonça

Kids Block Coding Game: A game to introduce programming to kids 256
Luis Forquesato, Juliana Freitag Borin

Relato de Experiência da Aplicação da Metodologia Ativa de Ensino com Pesquisa na Disciplina de Sistemas de Informação 266
Felipe Sampaio, Jefferson Pereira de Almeida

Uma Experiência sobre a Aplicação de Aprendizagem Baseada em Projetos com Revisão por Pares no Ensino de Gestão de Sistemas de Informação 276
Leo Natan Paschoal, Simone Souza

Laboratório remoto de robótica para o ensino de programação com suporte à análise, codificação e teste 286
Maísa Lopes

Uma Metodologia para Implementação da Disciplina Informática Básica em Cursos Técnicos Integrados ao Ensino Médio 296
Lafayette Melo, Paulo Costa, Marcílio Onofre Filho, Fernando Lordão, Leandro C. de Almeida

Relato de experiência vivenciada no PIBID sobre a utilização da Computação Desplugada, a Hora do Código e do Scratch no Ensino Médio 306
Anna Raquel da Silva Marinho, Pauleany Simões de Morais, Givanaldo Souza, Alba Sandyra Bezerra Lopes

MetricRA: Learning Software Metrics through Augmented Reality 316
Rebeca Motta, Mario Bonicenha, Claudia Susie Rodrigues, Claudia Werner

Um Relato de Experiência: Ensinando Robótica por meio de Microcontroladores em uma Escola Profissional de Ensino Médio 326
Mateus Alves, Robertty Freitas, Paulo Miranda e Silva Sousa, Paulo Ricardo, Carla Ilane Moreira Bezerra

Ensino de bioética em cursos superiores de computação: uma análise crítica 336
Rodrigo Candido Borges, Maria Márcia Bachion

O desenvolvimento da identidade docente por professores de Computação não licenciados atuantes na Educação Profissional de Nível Médio 346
Ranansamir da Silva, Ecivaldo Matos, Monica Massa

Idealizando Jogos Digitais de Pensamento Computacional a Partir do Bebras Challenge: Um Estudo Exploratório 362
Jéssica Laisa, Eduardo Aranha, Wendell Oliveira

Estágio Docente de Licenciatura de Computação: Um Ensaio do Ensino de Computação no Ensino Fundamental 372
José Anderson Soares da Silva, Hérikles Cordeiro, Anselmo Gomes, Sônia Fortes da Silva

As Aventuras de Ada e Turing: Apoiando o Desenvolvimento de Habilidades do Pensamento Computacional por meio de um Jogo 382
Géssica Silva, Leticia Leite, Ana Carolina Lopes de Jesus

Towards better tools and methodologies to teach computational thinking to children 398
Lais Minchillo, Augusto Vellozo, Edson Borin, Juliana Freitag Borin

As ações do PET no desenvolvimento do curso de Ciência da Computação 414
Leonardo Bandeira de Lucena, Giovana Lorena Andrade, Elisa de Fátima Soares, Wilton Júnior, Álvaro Oliveira, Ligia Maria de Sousa Dantas Batista, Daniel Alves Gomes, Rommel Lima

Método baseado nos Referenciais de Formação da SBC para reestruturação de descritivos de disciplinas de Ciência da Computação em conformidade com as DCN de 2016 423
Alcides Calsavara, Ana Paula Serra, Francisco Zampirolli, Leandro Galvão, Miguel Jonathan, Ronaldo Correia

Criando aplicativos sobre o descarte adequado de lixo: experiências utilizando uma abordagem temática em um clube de programação 433
Andrea Charao, Ana Luisa Solórzano, Rafael Gauna Trindade

Cenários Prospectivos: Uma Visão do Futuro da Presença Feminina em Cursos de Ciência da Computação de uma Instituição de Ensino Superior 443
Josilene Moreira, Ricardo Moreira, Maria Carvalho

Uma Abordagem para Orquestração do Conhecimento como Suporte ao Planejamento Curricular em Ciência da Computação 453
Anderson Barbosa, Janderson Aguiar, Ulrich Schiel

A inserção de Computação como disciplina no Ensino Fundamental: Desafios e Conquistas em Estágio Supervisionado 467
Uryel Felix, Arianne Sarmento, Sônia Fortes da Silva, Cleiton Martins

Percepção dos Estudantes sobre a Implantação de uma Disciplina Regular de Pensamento Computacional em um Colégio de Educação Básica 477
Natália Ellery, André Raabe, Elieser Ademir de Jesus, Eduardo Silva

BrC: Proposta de uma Biblioteca em Português para Ensino de Programação em Linguagem C 493
Luciana Guedes, Aleksander Sade Paterno

Cosmo: Um ambiente virtual de aprendizado com foco no Ensino de Algoritmos 503
Dilson Rabelo Júnior, Carlos Soares Neto, Antonio Carlos Raposo, Luis Alves dos Santos Neto

Oficinas de Programação para Meninas: Despertando o Interesse Pela Computação 511
Giorgia Mattos, Josilene Moreira, Ana Flávia Silva Aragão Moura, Andrea Brito Nascimento, Chaenne Carolina Oliveira

Desenvolvimento e avaliação de uma metodologia de Aprendizagem Ativa apoiada pelo uso de QR Code para ensino de Banco de Dados 521
Ronney Moreira de Castro, Sean Siqueira

Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores 531
Carolina Moreira Oliveira, Andrey Pimentel, Eleandro Maschio

Construção Colaborativa de Signos Específicos da Língua Brasileira de Sinais para Termos da Subárea de Engenharia de Software 547
José Augusto Fabris, Soraia Prietch, Kefferson Ricardi

Ponteiros em Papel: O ensino e a aprendizagem de ponteiros em linguagem de programação C com base em envelopes coloridos 557
Gustavo Kira, Luiz Merkle

Kahoot!: um relato de experiência no contexto acadêmico 567
Luciana Mara Diniz, Fischer Jonatas

Uso de Realidade Aumentada para Ensino de Arquitetura de Computadores com MIPS 577
Geofrangite da Silva, Leiva Oliveira, Silvio Fernandes

Avaliação de uma Dinâmica Vivencial para o Ensino de Gerenciamento de Projetos em Cursos de Computação

Giani Petri[1, 2], Christiane Gresse von Wangenheim[2], Bruno B. Boniati[3], Alex R. Weber[2, 4]

[1]Universidade Federal de Santa Maria (UFSM)
Av. Roraima, 1000 – 97.105-900 – Santa Maria, RS – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia Farroupilha (IFFar)
Campus Frederico Westphalen – Frederico Westphalen, RS – Brasil

[4]Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina (IFSC)
Campus Xanxerê – Xanxerê, SC – Brasil

gpetri@inf.ufsm.br, c.wangenheim@ufsc.br, bruno.boniati@iffarroupilha.edu.br, alex.weber@ifsc.edu.br

Abstract. *The current scenario in the software industry requires qualified professionals who know the project management process. In this context, experiential dynamics, used in professional workshops, are being used in the academic context for project management education. Thus, this article aims to evaluate the quality, in terms of perceived learning and player experience, of PMDome, an instructional strategy used to simulate the planning and execution phases of project management in computing courses. The evaluation from 47 students from different educational levels provides evidence on the benefits of PMDome for students' learning, as well as its positive contribution to the social interaction, fun, satisfaction and confidence.*

Resumo. *O cenário atual do setor de software requer profissionais qualificados e que conheçam os processos de gerenciamento de projetos. Neste contexto, dinâmicas vivenciais e de simulação, utilizadas em workshops profissionais, estão sendo utilizadas no contexto acadêmico para o ensino de gerenciamento de projetos. Assim, este trabalho objetiva avaliar a qualidade, em termos de percepção da aprendizagem e experiência do jogador, da dinâmica PMDome, uma estratégia de ensino usada para simular as fases de planejamento e execução do gerenciamento de projetos em cursos de computação. A avaliação, envolvendo 47 alunos de cursos de diferentes níveis de ensino, evidenciam a qualidade da dinâmica PMDome para a aprendizagem dos alunos, além de contribuir positivamente para a interação social, diversão, satisfação e confiança.*

1. Introdução

As estatísticas do cenário mundial de software demonstram que a maioria dos projetos de software ainda apresentam problemas [Standish Group, 2015]. Conforme o Chaos

Report de 2015, cerca de 52% dos projetos precisam de alteração, 19% falham por completo e, apenas, 29% são completados de forma satisfatória, respeitando o prazo e o orçamento planejado [Standish Group, 2015]. Um dos motivos para estes resultados, dentre outros, é a falta de conhecimento, da equipe de desenvolvimento, nos processos de gerenciamento de projetos [Standish Group, 2015].

Nos cursos de computação, a gerência de projetos é ensinada em disciplinas específicas ou como um tópico menor em disciplinas de Engenharia de Software [ACM/IEEE-CS, 2013]. Tipicamente a gerência de projetos é ensinada por meio de aulas expositivas, que são adequadas para apresentar informações e conceitos [Barnes et al., 2008]. Mas, por ser uma estratégia centrada no professor, não apoia a aplicação prática de competências [Barnes et al., 2008; Parsons, 2011]. No entanto, considerando o perfil dos estudantes de computação atualmente [Parsons, 2011], é necessário utilizar estratégias de ensino focadas no aluno, permitindo que eles aprendam fazendo, por meio de suas próprias experiências.

Atualmente, há diversos trabalhos que objetivam complementar o ensino de gerenciamento de projetos por meio de atividades práticas, que proporcione aos alunos uma aprendizagem ativa [Calderón e Ruiz, 2015; Letra et al., 2015; Petri et al., 2017a]. Jogos educacionais estão sendo utilizados como estratégia instrucional para revisar conceitos básicos por meio de quiz games [Petri et al., 2016], simular o ciclo de vida de um projeto de software em um jogo digital [Calderón e Ruiz, 2013], ou praticar os conceitos de Scrum no gerenciamento ágil de projetos [Battistella et al., 2016]. Além disso, dinâmicas que objetivam simular um projeto envolvendo toda equipe no desenvolvimento de algum produto, também estão sendo utilizadas como estratégia instrucional [Gresse von Wangenheim et al., 2013; Paasivaara et al., 2014]. Nos últimos anos, o uso de dinâmicas vivenciais e de simulação, que são utilizadas em workshops profissionais, específicos para a formação de gerentes de projetos, também estão sendo utilizadas no contexto acadêmico [Gresse von Wangenheim et al., 2013, Petri e Marcon Jr, 2014].

Deste modo, acredita-se que os jogos e as dinâmicas vivenciais possam ser estratégias instrucionais eficazes e eficientes para o ensino de gerenciamento de projetos [Gibson e Bell, 2013; Souza e França, 2016; Kosa et al., 2016]. No entanto, estas alegações são questionáveis ou não rigorosamente comprovadas [Calderón e Ruiz, 2015; Petri e Gresse von Wangenheim, 2017]. Na prática, as dinâmicas vivenciais utilizadas para o ensino de gerenciamento de projetos em contexto educacional necessitam mostrar o impacto esperado de aprendizagem e/ou o engajamento que elas prometem [Connolly et al., 2012; Kosa et al., 2016; Çiftci, 2018]. Portanto, é essencial avaliar sistematicamente a qualidade dessas dinâmicas, a fim de obter provas sólidas sobre a sua qualidade [Kosa et al., 2016].

Diante disso, este trabalho objetiva avaliar a qualidade de uma dinâmica vivencial utilizada em workshops profissionais, conhecida como PMDome [PMDome, 2017], como uma estratégia de ensino de gerenciamento de projetos. A dinâmica PMDome objetiva que os participantes assimilem os principais conceitos do gerenciamento de projetos, reconheçam a importância do planejamento e desenvolvam um produto durante a atividade. O produto final do projeto é uma estrutura física denominada Domo Geodésico, construída a partir de folhas de papel e fita adesiva. A escolha da dinâmica é justificada devido a sua aderência aos objetivos de aprendizagem

e perfil dos alunos participantes. Para a avaliação da qualidade da dinâmica PMDome é utilizado o modelo MEEGA+ [Petri et al., 2017b], um modelo sistematicamente desenvolvido para a avaliação de jogos educacionais, que avalia a qualidade do jogo em termos de experiência do jogador e percepção da aprendizagem. A avaliação é realizada por uma série de estudos de caso envolvendo um total de 47 alunos de computação de diferentes níveis de ensino.

2. Método de Pesquisa

Com o objetivo de avaliar a qualidade da dinâmica PMDome em termos de experiência do jogador e percepção da aprendizagem do ponto de vista de alunos de computação, foi realizada uma série de estudos de caso [Yin, 2017], como ilustrado na Figura 1.

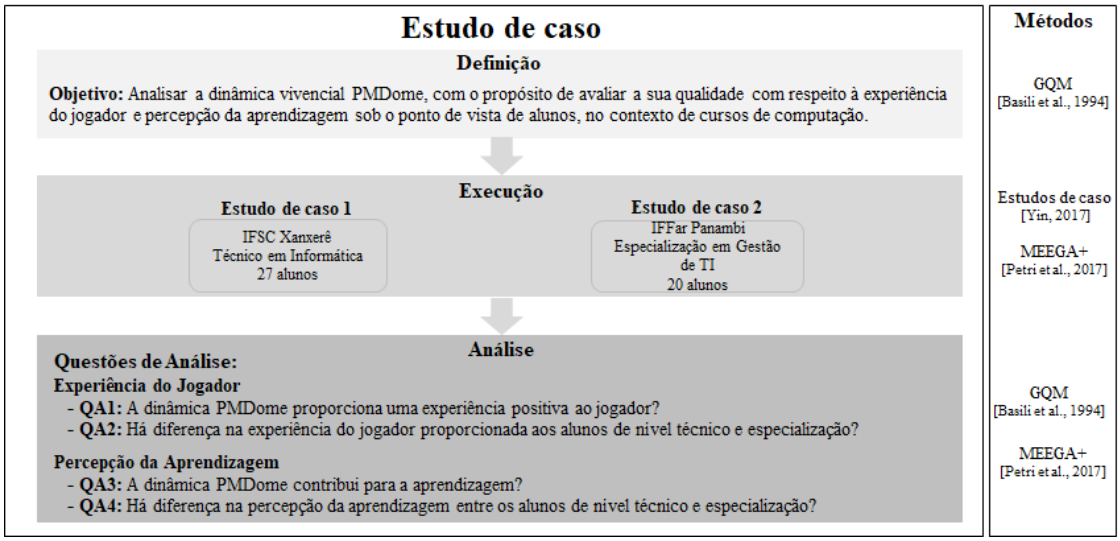


Figura 1. Método de pesquisa

Seguindo a abordagem GQM (*Goal/Question/Metric*) [Basili et al., 1994], na fase de definição, o objetivo do trabalho é decomposto em aspectos de qualidade e questões de análise para serem analisadas com base no modelo MEEGA+ [Petri et al., 2017b]. Na fase de execução, os estudos de caso são planejados e executados com o público-alvo. Após o tratamento (aplicação da dinâmica PMDome), o questionário do modelo MEEGA+, adaptado para o contexto da dinâmica, é respondido pelos participantes (alunos) para coletar os dados da avaliação da dinâmica PMDome. Na fase de análise, os dados coletados nos dois estudos de caso são agrupados para a análise da qualidade da dinâmica PMDome, e então, responder as questões de análise definidas. Os dados foram analisados seguindo o modelo MEEGA+ utilizando técnicas de estatística descritiva, em termos de distribuição da frequência e medida da tendência central (mediana), de modo a responder as questões de análise. A partir dos dados coletados, os percentuais das frequências das respostas foram identificados.

3. A Dinâmica PMDome

A dinâmica PMDome tem como objetivo de aprendizagem transmitir, dentro do contexto de gerenciamento de projetos, a importância do planejamento e o impacto que a falta do mesmo pode ter. Espera-se que após a dinâmica os alunos apresentem uma mudança de atitude, que eles reconheçam a importância do planejamento e respondam

de forma planejada. A dinâmica é aplicada como atividade introdutória de disciplinas de gestão e gerenciamento de projetos, de modo a sensibilizar os alunos a importância do conteúdo da disciplina. Assim, não é necessário que os participantes tenham conhecimentos prévios em gerenciamento de projetos.

A dinâmica simula as fases de planejamento e execução do gerenciamento de projetos, onde os estudantes precisam planejar o tempo e os recursos (canetas, folhas de papel e fita adesiva) que precisarão para desenvolver o produto final, um Domo Geodésio. No início da atividade, os alunos, divididos em grupos de no máximo 10 participantes, recebem uma folha com as instruções do que precisam realizar na atividade. Em seguida, cada grupo deve definir um gerente de projetos, responsável por coordenar as atividades do grupo, planejar e estimar um tempo necessário para a construção do Domo Geodésio. Nos 30 minutos de planejamento, os alunos podem tirar suas dúvidas com o instrutor da atividade sem restrições, porém, na fase de execução, os alunos poderão fazer somente duas perguntas, por escrito, para o instrutor.

Na fase de execução os alunos seguem três etapas para construir tubos de papel e montar o Domo. Na primeira etapa, cada grupo deve construir 65 tubos de papel enrolados em diagonal, partindo de uma extremidade à outra: 35 tubos de 65 centímetros (tubo tamanho G) e 30 tubos de 58 centímetros (tamanho P). Na segunda etapa, cada grupo deve montar seis pentágonos. O passo inicial para montar um pentágono é montar um triângulo isósceles composto por 2 tubos P e 1 tubo G, conforme apresenta a Figura 2a. Em seguida, deve-se acrescentar mais dois tubos (um G e um P), até formar o pentágono, conforme apresenta a Figura 2b.

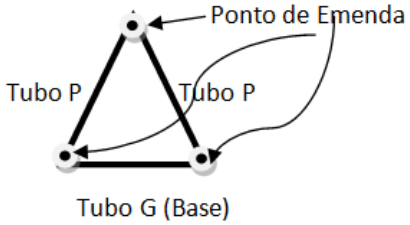


Figura 2a. Construindo triângulos isósceles

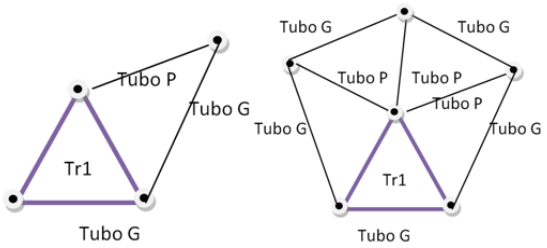


Figura 2b. Construindo pentágonos

A terceira etapa objetiva montar a estrutura do Domo com base nos pentágonos construídos na etapa anterior. São necessários 5 pentágonos para montar a base do Domo e mais 1 pentágono para montar o teto do Domo. Os pentágonos devem ser unidos conforme apresenta a Figura 3.

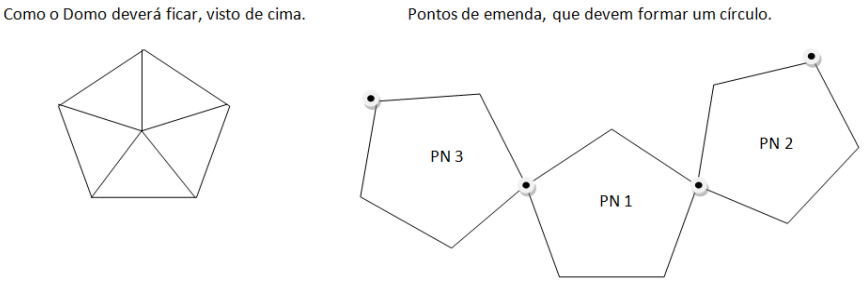


Figura 3. Montando a estrutura do Domo

Os cinco pentágonos devem ser colocados em círculos, unindo-os em apenas um ponto. Ao final, o sexto pentágono deverá fechar a parte de cima e os tubos que restam são usados para fechar a base da estrutura do Domo.

Ao final da dinâmica, o instrutor coordena uma discussão com todos os participantes de modo a refletir sobre as lições aprendidas na atividade, as oportunidades de aprendizagem e o reconhecimento da importância do planejamento. Algumas questões contribuem na discussão, tais como: Como o grupo se planejou? O que deu certo no planejamento? O que deu errado? O planejamento ajudou na execução das atividades? Os participantes executaram as atividades como planejado? Qual a importância do planejamento para a execução das atividades no tempo disponível?

4. Definição e Execução da Avaliação da Dinâmica PMDome

O objetivo deste trabalho é analisar a dinâmica vivencial PMDome, com o propósito de avaliar a sua qualidade com respeito a experiência do jogador e percepção da aprendizagem sob o ponto de vista de alunos, no contexto de cursos de computação. A partir deste objetivo, seguindo o modelo MEEGA+ [Petri et al., 2017b] são definidas as seguintes questões de análise, agrupadas pelos fatores de qualidade avaliados:

Experiência do Jogador

QA1: A dinâmica PMDome proporciona uma experiência positiva aos jogadores?

QA2: Há diferença na experiência do jogador proporcionada aos alunos de nível técnico e especialização?

Percepção da Aprendizagem

QA3: A dinâmica PMDome contribui para a aprendizagem?

QA4: Há diferença na percepção da aprendizagem entre os alunos de nível técnico e especialização?

Foram realizados dois estudos de caso (Tabela 1) utilizando a dinâmica PMDome em duas turmas de computação, em diferentes níveis de ensino, em nível técnico e especialização. Em cada estudo de caso após o tratamento (aplicação da dinâmica PMDome), o questionário adaptado do modelo MEEGA+ foi respondido pelos participantes (alunos) para coletar os dados da avaliação da dinâmica PMDome. A pesquisa realizada foi aprovada pelo Comitê de Ética da Universidade Federal de Santa Catarina, instituição executora do estudo, sob parecer número 1.601.297 de 21/06/2016.

Tabela 1 – Estudos de caso realizados

Estudo de Caso 1 – IFSC Xanxerê	Estudo de Caso 2 – IFFar Panambi
Conduzido no Instituto Federal de Santa Catarina (IFSC), campus de Xanxerê, durante a Semana Nacional de Ciência e Tecnologia, promovida pelo IFSC. 27 alunos participaram da atividade.	Conduzido no Instituto Federal Farroupilha (IFFar), campus Panambi, na aula introdutória do curso de Especialização em Gestão de Tecnologia da Informação, com 20 alunos.
	
Figura 4. Estudo de caso 1 – IFSC Xanxerê	**Figura 5. Estudo de caso 2 – IFFar Panambi**

No total participaram 47 alunos, 35 homens e 12 mulheres. A maioria dos alunos possui menos de 18 anos (27 alunos), 9 alunos possuem entre 18 e 28 anos, 7 alunos possuem de 29 a 39 anos e 4 alunos possuem mais de 40 anos. 34% dos alunos costumam jogar semanalmente, 21% jogam mensalmente e 34% raramente jogam jogos não-digitais e/ou dinâmicas.

5. Análise dos dados

Os dados coletados nos dois estudos de caso foram agrupados em uma única amostra para avaliar a qualidade da dinâmica PMDome de modo a responder as questões de análise QA1 e QA3. No entanto, para analisar as questões de análise QA2 e QA4, os dados de cada estudo de caso são analisados separadamente, e são representados pela mediana das respostas para cada item avaliado.

QA1: A dinâmica PMDome proporciona uma experiência positiva aos jogadores?

De forma geral, em termos de experiência do jogador, os alunos perceberam uma positiva experiência proporcionada (Figura 6). Isso demonstra que o uso da dinâmica PMDome, como estratégia instrucional, proporcionou uma boa experiência aos jogadores, especialmente em termos de diversão, interação e cooperação entre os alunos.

Os alunos confirmaram que a dinâmica PMDome possui uma boa usabilidade, indicando que foi fácil de aprender e começar a jogar e, principalmente, que as regras eram claras e compreensíveis. A primeira impressão dos alunos foi de que a dinâmica não seria fácil para eles. No entanto, a organização do conteúdo envolvido na dinâmica contribuiu para que os alunos se sentissem confiantes de que aprenderiam com ela.

Os alunos avaliaram que a dinâmica foi desafiadora a eles, apresentando variações de desafios e/ou obstáculos em um ritmo adequado. No entanto, cabe salientar que alguns alunos (15%) indicaram que a dinâmica se tornou monótona ou repetitiva. Esta avaliação pode estar relacionada ao momento de criação dos tubos de papel para a montagem do domo, pois é preciso construir 65 tubos e então, pode se tornar uma tarefa repetitiva.

PMDome também proporcionou um sentimento de satisfação e realização aos alunos. E o esforço coletivo foi importante para o grupo avançar na dinâmica. Este resultado também está relacionado a avaliação da interação social, que foi avaliada de forma bastante positiva. Isso demonstra que a dinâmica pode ser uma boa estratégia para promover a interação entre os alunos. Os alunos avaliaram que a dinâmica contribuiu para a cooperação e proporcionou uma experiência divertida com outras pessoas. Além disso, eles foram capazes de interagir com os colegas durante a dinâmica, proporcionando uma sensação de ambiente compartilhado.

Em relação à atenção focada, a dinâmica também foi avaliada de forma positiva. Este resultado mostra que os alunos tiveram uma experiência de envolvimento profundo na dinâmica, esquecendo-se de suas atividades diárias, concentrando-se nas tarefas da dinâmica e perdendo a noção do tempo durante a atividade. Os alunos indicaram que a dinâmica foi divertida e que algumas situações da dinâmica os fizeram sorrir. Os alunos também indicaram que a dinâmica foi relevante para seus interesses, indicando a relação da atividade com o conteúdo aprendido. Além disso, os alunos classificaram a dinâmica

PMDome como um método de ensino adequado e que preferem aprender desta forma do que com outro método de ensino.

QA2: Há diferença na experiência do jogador proporcionada aos alunos de nível técnico e especialização?

Analisando a experiência proporcionada pela dinâmica PMDome nos diferentes níveis de ensino, identificou-se que ambas as turmas a avaliaram como positiva (Figura 6). Isto demonstra que, embora os interesses dos alunos e o foco dos cursos de nível técnico e especialização sejam diferentes, todos os alunos experimentaram a dinâmica PMDome como uma estratégia de ensino positiva e envolvente usando uma dinâmica vivencial. Em termos de usabilidade e diversão, ambas as turmas avaliaram de forma similar, indicando a dinâmica PMDome como uma experiência positiva.

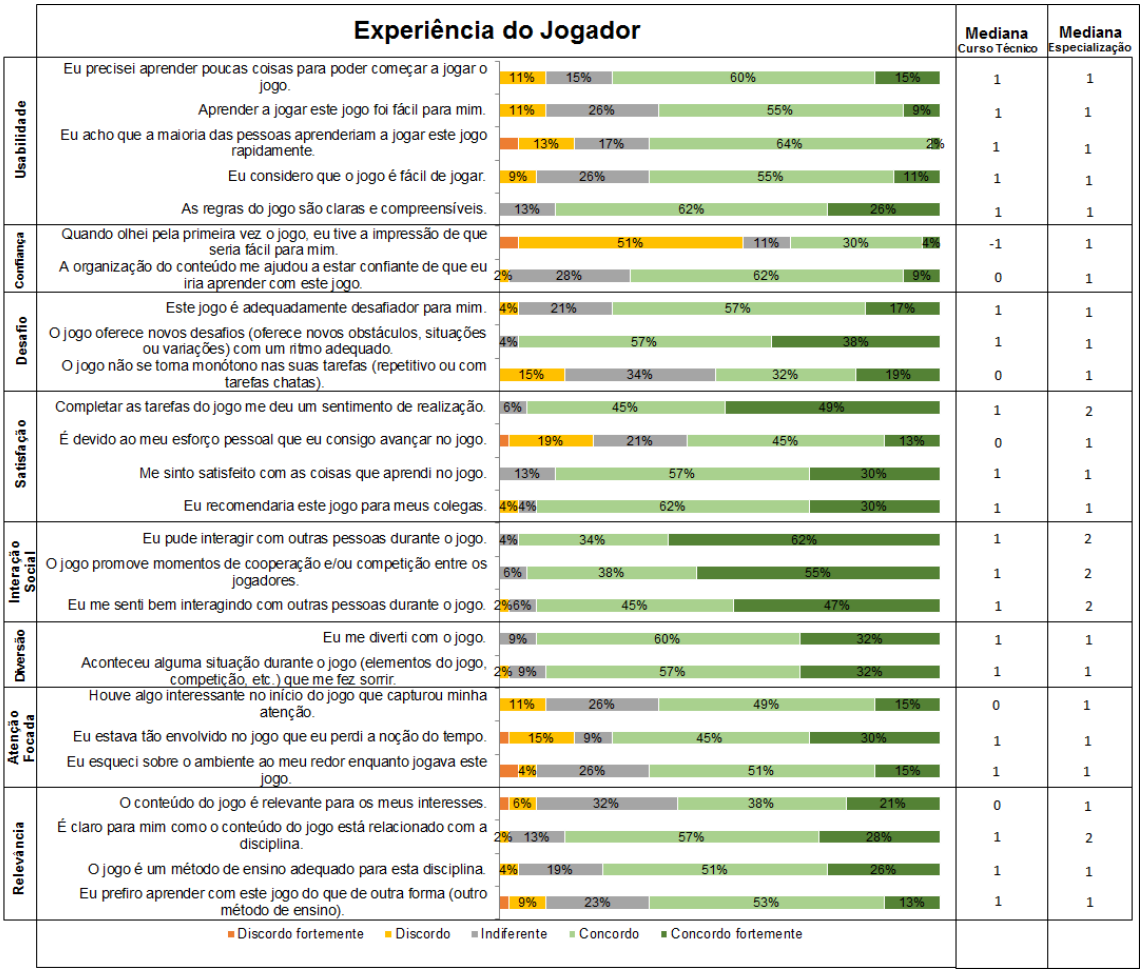


Figura 6. Diagrama de frequência e medianas da avaliação da experiência do jogador

Em termos de confiança, os alunos do curso de especialização sentiram-se mais confiantes de que a dinâmica seria fácil e que aprenderiam com ela, do que os alunos de nível técnico. Este resultado pode estar relacionado a maturidade e experiência dos alunos, pelo fato de que os alunos de nível técnico tiveram pouco acesso a informação sobre a área de gerenciamento de projetos durante o curso.

Em relação ao desafio proporcionado pela dinâmica PMDome, embora com resultado similar, os alunos do curso de especialização avaliaram que a dinâmica não se

torna monótona, ao contrário dos alunos do curso técnico, que indicaram que a dinâmica se torna monótona ou com tarefas chatas. Esta avaliação pode estar relacionada as tarefas repetitivas de criação dos materiais (tubos) para a montagem do domo. E devido a dinamicidade do perfil dos alunos na faixa etária dos 17 anos, podem ter considerado esta tarefa repetitiva e/ou chata. O que evidencia a necessidade de desenvolver atividades bastante dinâmicas quando o público alvo de uma atividade possui este perfil.

A dinâmica proporcionou um sentimento de satisfação aos alunos de ambos os cursos, especialmente aos alunos do curso de especialização. De mesmo modo a interação social foi avaliada positiva por todos os alunos. No entanto, os alunos de especialização avaliaram de forma bastante positiva a interação e cooperação com os colegas durante a realização da dinâmica. Também estimulou a concentração e a atenção focada dos alunos, capturando a atenção dos alunos de especialização logo no início da atividade. Este resultado é importante, pelo fato de promover o engajamento dos alunos na atividade estimulando o pensamento, a criatividade e a interação social. Em termos de relevância, embora a dinâmica tenha sido indicada como relevante por ambas as turmas, os alunos de especialização a avaliaram de forma mais positiva, indicando a forte relação da dinâmica com o conteúdo envolvido.

QA3: A dinâmica PMDome contribui para a aprendizagem?

De forma geral, em termos de aprendizagem, os alunos perceberam uma positiva contribuição da dinâmica para a sua aprendizagem sobre gerenciamento de projetos (Figura 7). Os alunos indicaram uma percepção que a dinâmica PMDome contribuiu para a sua aprendizagem na disciplina; que a dinâmica foi eficiente quando comparado com outras estratégias de ensino utilizadas e também que a dinâmica contribuiu para os alunos reconhecerem a importância do gerenciamento para a realização de um projeto, incluindo o gerenciamento de materiais, tempo, recursos humanos, riscos, etc.

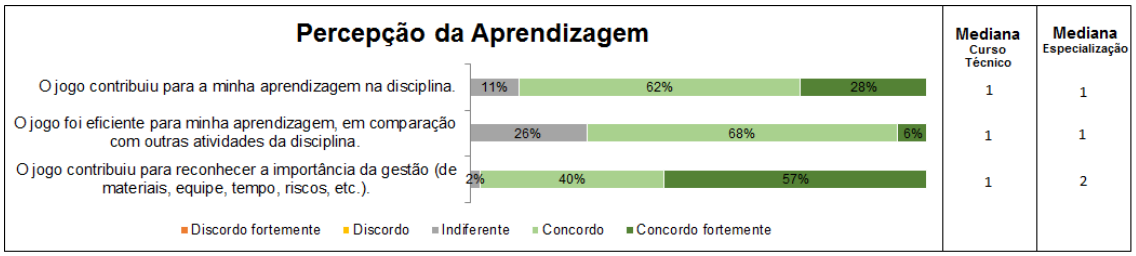


Figura 7. Diagrama de frequência e medianas da avaliação da percepção da aprendizagem

QA4: Há diferença na percepção da aprendizagem entre os alunos de nível técnico e especialização?

Analisando a avaliação da percepção da aprendizagem de cada turma (Figura 7), observou-se que, embora as avaliações são, no geral, positivas, os alunos dos cursos de especialização indicaram forte concordância ao avaliarem que a dinâmica PMDome contribuiu para reconhecer a importância do gerenciamento. Este resultado pode estar relacionado a experiência profissional e a visão sobre a área de computação que os alunos de especialização podem ter mais desenvolvido quando comparado aos alunos de nível técnico, iniciantes na área de computação. No entanto, é perceptível que a dinâmica PMDome contribuiu para a aprendizagem dos alunos, tanto de nível técnico, como os alunos de especialização.

5.1 Ameaças à validade

Devido as características deste tipo de pesquisa, este trabalho está sujeito a algumas ameaças à validade. Deste modo, foram identificadas ameaças potenciais e aplicadas estratégias de mitigação para minimizar o impacto no trabalho. Para tratar as limitações na medição dos fatores de qualidade avaliados (conceitos subjetivos como experiência do jogador), foi usado o modelo MEEGA+ [Petri et al., 2017b] um modelo de avaliação de jogos sistematicamente desenvolvido e avaliado em termos de validade e confiabilidade [Petri et al., 2018]. Embora a pesquisa tenha envolvido um número aceitável de alunos de diferentes níveis de ensino, a aplicação da pesquisa nos próximos anos e em mais disciplinas pode ser uma alternativa para consolidar os resultados a partir deste trabalho. Além disso, como não houve um pré-teste medindo, por exemplo, o nível de conhecimento dos alunos antes da aplicação da dinâmica, pois causaria uma interrupção no fluxo das aulas, não foi possível identificar com exatidão o quanto cada aluno aprendeu sobre o assunto. No entanto, embora sem consenso, existem evidências de que a auto avaliação fornece informações confiáveis, válidas e úteis para este tipo de estudo [Sitzmann et al., 2010].

6. Conclusão

Com base em uma série de estudos de caso obteve-se uma primeira indicação que a dinâmica vivencial utilizada em workshops profissionais (PMDome), pode ser uma estratégia de ensino de gerenciamento de projetos em cursos de nível técnico e especialização com um impacto benéfico. Os resultados evidenciam a qualidade da dinâmica PMDome para a aprendizagem dos 47 alunos participantes de nível técnico e especialização. Além de contribuir positivamente para a interação social dos alunos, diversão, satisfação e confiança.

Portanto, conclui-se que a utilização de dinâmicas utilizadas em workshops profissionais, em especial a dinâmica PMDome, possui também um forte potencial em contribuir para a aprendizagem dos alunos também quando inserida em contexto educacional. Como trabalhos futuros, pretende-se aplicar e avaliar a dinâmica PMDome em outros níveis de ensino de modo a confirmar os resultados e identificar oportunidades de melhoria.

Agradecimentos

Aos alunos do Curso Técnico em Informática do IFSC e da Especialização em Gestão de Tecnologia da Informação do IFFar, pela participação nos estudos de caso.

Este trabalho foi apoiado pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), uma entidade do governo brasileiro focada no desenvolvimento científico e tecnológico.

Referências

ACM/IEEE-CS. (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.

Barnes, T., Powell, E., Chaffin, A., e Lipford, H. (2008). Game2Learn: improving the motivation of CS1 students. Proc. of the 3rd Int. Conf. on Game Development in Computer Science Education, (pp. 1-5). New York, NY, USA.

Basili, V. R., Caldiera, G., e Rombach, H. D. (1994). Goal, Question Metric Paradigm. In J. J. Marciniak, Encyclopedia of Software Engineering. New York, USA.

Battistella, P., Camargo, A., Gresse von Wangenheim, C. (2016). SCRUM-Scape: Jogo educacional de Role-Playing Game (RPG) para ensinar SCRUM. 27° Simpósio Brasileiro de Informática na Educação, (pp. 330-339). Uberlândia, MG, Brasil.

Calderón A. e Ruiz M. (2015). A systematic literature review on serious games evaluation: An application to software project management. Computers & Education, vol. 87, 396-422.

Çiftci, S. (2018). Trends of Serious Games Research from 2007 to 2017: A Bibliometric Analysis. Journal of Education and Training Studies, 6(2), 18-27.

Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. Computers & Education, 59(2), 661-686.

Gibson, B. & Bell, T. (2013). Evaluation of games for teaching computer science. Proc. of the 8th Workshop in Primary and Secondary Computing Education, (pp. 51-60). ACM, New York, NY, USA.

Gresse von Wangenheim, C., Savi, R., e Borgatto, A. F. (2013). SCRUMIA - An educational game for teaching SCRUM in computing courses. Journal of Systems and Software, 86(10), 2675-2687.

Kosa, M., Yilmaz, M., O'Connor, R., & Clarke, P. (2016). Software engineering education and games: a systematic literature review. Journal of Universal Computer Science, 22(12), 1558-1574.

Letra, P., Paiva, A. C. R., & Flores, N. (2015). Game Design Techniques for Software Engineering Management Education. Proc. of the 18th Int. Conf. on Computational Science and Engineering, (pp. 192-199). IEEE, Porto, Portugal.

Parsons, P. (2011). Preparing computer science graduates for the 21st Century. Teaching Innovation Projects, 1(1), article 8.

Paasivaara, M., Heikkilä, V., Lassenius, C. & Toivola, T. (2014). Teaching students Scrum using LEGO blocks. Proc. of the 36th Int. Conf. on Software Engineering, (pp. 382-391). ACM, New York, NY, USA.

Petri, G. e Gresse von Wangenheim, C. (2016). How to Evaluate Educational Games: a Systematic Literature Review. Journal of Universal Computer Science, 22(7), 992-1021.

Petri, G. e Gresse von Wangenheim, C. (2017). How games for computing education are evaluated: a systematic literature review. Computers & Education, vol. 107, 68-90.

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2017b). Evolução de um Modelo de Avaliação de Jogos para o Ensino de Computação. 25° Workshop sobre Educação em Computação, (pp. 2327-2336). São Paulo, SP, Brasil.

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2017a). Quality of Games for Teaching Software Engineering: An Analysis of Empirical Evidences of Digital and Non-digital Games. Proc. of the 39th Int. Conf. on Software Engineering: Software

Engineering Education and Training Track, (pp. 150-159). IEEE/ACM, Buenos Aires, Argentina.

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2018). MEEGA+, Systematic Model to Evaluate Educational Games. In Newton Lee (Eds) Encyclopedia of Computer Graphics and Games, (pp. 1-7). Cham: Springer.

Petri, G., Battistella, P., Cassettari, F., Gresse von Wangenheim, C., e Hauck, J. (2016). Um Quiz Game para a revisão de conhecimento em Gerenciamento de Projetos. 27° Simpósio Brasileiro de Informática na Educação, (pp. 320-329). Uberlândia, MG, Brasil.

Petri, G. e Marcon Júnior, R. P. (2014). Um jogo educacional para o ensino de metodologias ágeis. 7º Fórum de Educação em Engenharia de Software, (pp. 66-69). Maceió, AL, Brasil.

PMDome. (2017). PMDome Workshop. Disponível em: <https://ricardo-vargas.com/pt/workshops/pmdome/> Acesso em: 27 novembro 2017.

Sitzmann, T., Ely, K., Brown, K. G., e Bauer, K. N. (2010). Self-Assessment of Knowledge: A Cognitive Learning or Affective Measure? Academy of Management Learning & Education, 9(2), 169-191.

Souza, M. e França, C. (2016). O que explica o sucesso de jogos no ensino de engenharia de software? Uma teoria de motivação. 24º Workshop sobre Educação em Computação, (pp. 2255-2263). Porto Alegre, RS, Brasil.

Standish Group. (2015). CHAOS Report 2015.

Yin, R.K. (2017). Case study research and applications: design and methods (5 ed.). Sage Publications, Beverly Hills.

Benefícios dos Jogos Não-Digitais no Ensino de Computação

Giani Petri[1,2], Alejandro Calderón[3], Christiane G. von Wangenheim[2], Adriano F. Borgatto[2], Mercedes Ruiz[3]
[1]Universidade Federal de Santa Maria (UFSM), Santa Maria/RS, Brasil
[2]Departmento de Informática e Estatística (INE), Universidade Federal de Santa Catarina (UFSC), Florianópolis/SC, Brasil
[3]Universidade de Cádiz (UCA), Puerto Real (Cádiz), Espanha

gpetri@inf.ufsm.br, alejandro.calderon@uca.es, c.wangenheim@ufsc.br, adriano.borgatto@ufsc.br, mercedes.ruiz@uca.es

Abstract. *Non-digital educational games (board and card games, paper & pencil, etc.) are also being used for computing education. However, there are few empirical studies providing sound evidence on the benefits of this kind of games. Consequently, a question that arises is to which regard the expected benefits of these games are real. Thus, the objective of this study is to analyse the benefits of non-digital educational games used for computing education in order to evaluate quality in terms of the players' experience and perceived learning adopting the MEEGA+ games' evaluation model. The results of the evaluation, involving 26 case studies and responses from 509 students, provides evidence that these non-digital games contribute to the students' learning, as well as contribute positively to social interaction, relevance, and fun.*

Resumo. *Jogos educacionais não-digitais (jogos de tabuleiro, cartas, lápis & papel, etc.) também estão sendo usados para o ensino de computação. No entanto, há poucos estudos empíricos que mostram evidências sobre os benefícios deste tipo de jogos. Consequentemente, uma questão em aberto é se os benefícios esperados destes jogos são, de fato, reais. Assim, o objetivo deste trabalho é analisar os benefícios dos jogos educacionais não-digitais usados para o ensino de computação de modo a avaliar a qualidade em termos de experiência dos jogadores e percepção da aprendizagem adotando o modelo de avaliação de jogos MEEGA+. Os resultados da avaliação, envolvendo 26 estudos de caso e respostas de 509 alunos, evidenciam que os jogos não-digitais contribuem para a aprendizagem dos estudantes, além de contribuir positivamente para a interação social, relevância e diversão.*

1. Introdução

Nos últimos anos, os jogos educacionais têm sido utilizados como estratégia instrucional para o ensino de computação de modo a contribuir para uma aprendizagem mais efetiva [Battistella e Gresse von Wangenheim, 2016a; Souza e França, 2016]. Esses jogos educacionais, usados como uma estratégia instrucional ativa, possuem características similares aos jogos de entretenimento, envolvendo competição e desafios entre os jogadores, sendo organizados por regras e restrições de modo a alcançar um determinado objetivo [Abt, 2002]. No entanto, os objetivos dos jogos educacionais vão além do entretenimento, eles são projetados especificamente para ensinar as pessoas sobre um determinado assunto, expandir conceitos, reforçar o desenvolvimento, ou

ajudá-los a explorar ou aprender uma habilidade ou uma mudança de atitude [Abt, 2002].

Embora a maioria dos jogos educacionais existentes para o ensino de computação sejam digitais, atualmente, há também uma tendência na adoção de jogos não-digitais (jogos de tabuleiro, cartas, papel & lápis, etc.) [Battistella e Gresse von Wangenheim, 2016a]. Esta tendência pode ser justificada pelo fato de que a maioria dos jogos educacionais é desenvolvida pelos próprios professores, com recursos e tempo muito limitados. E assim, como o desenvolvimento de jogos digitais requer um esforço considerável, muitas vezes esses jogos não alcançam todo o seu potencial de atratividade, complexidade e modo de interação [Petri et al., 2017a]. Por outro lado, jogos não-digitais tendem a ter um menor esforço no seu desenvolvimento, e parecem produzir, de forma intrínseca, um ambiente de interação social e imersão dos alunos sobre a tarefa de aprendizagem [Petri et al., 2017a; Petri et al., 2018a].

Neste contexto, diversos jogos educacionais não-digitais estão sendo desenvolvidos para potencializar o processo de aprendizagem em diferentes áreas da computação, tais como: Engenharia de Software (por exemplo, *Problems and Programmers* [Baker et al., 2005]), Gerenciamento de Projetos de Software (por exemplo, SCRUMIA [Gresse von Wangenheim et al., 2013b] e PlayScrum [Fernandes e Sousa, 2010]), Programação (por exemplo, C-Jump [Singh et al., 2007] e SORTIA [Battistella et al., 2017]), Segurança (por exemplo, *Security Protocol Game* [Hamey, 2003]), entre outros.

Deste modo, acredita-se que os jogos educacionais não-digitais potencializam diversos benefícios, como o aumento da eficácia da aprendizagem, aumento no interesse e motivação dos estudantes [Tahir e Wangmar, 2017; Çiftci, 2018]. Estes jogos podem também criar um ambiente divertido e seguro, onde os alunos podem aprender com seus próprios erros e experiências práticas em um ambiente compartilhado [Pfahl et al., 2001]. Assim, acredita-se que os jogos educacionais não-digitais possam ser uma estratégia instrucional adequada para o ensino de computação. No entanto, estas alegações são questionáveis ou não rigorosamente comprovadas, pelo fato de que a maioria das avaliações de jogos educacionais são realizadas de forma *ad-hoc* em termos de design de pesquisa, medição, coleta e análise de dados [Calderón e Ruiz, 2015; Petri e Gresse von Wangenheim, 2017]. Assim, uma questão em aberto é se estes benefícios esperados dos jogos educacionais são, de fato, reais.

Desta forma para obter uma compreensão mais abrangente dos benefícios dos jogos educacionais não-digitais usados para o ensino de computação, foi conduzida uma série de estudos de caso de avaliação de jogos. Os estudos de caso foram conduzidos com base no modelo de avaliação de jogos educacionais MEEGA+ [Petri et al., 2017b; Petri et al., 2018b], uma evolução do modelo de avaliação de jogos mais utilizado na prática [Calderón e Ruiz, 2015; Petri e Gresse von Wangenheim, 2016; Petri e Gresse von Wangenheim, 2017]. O MEEGA+ é um modelo sistematicamente desenvolvido para a avaliação de jogos educacionais, que avalia a qualidade do jogo em termos de experiência do jogador e percepção da aprendizagem. A análise deste estudo é baseada em um conjunto de dados de 26 estudos de caso conduzidos usando o modelo MEEGA+, avaliando 9 diferentes jogos não-digitais usados para o ensino de computação em diferentes instituições de ensino do Brasil e do exterior, envolvendo uma população total de 509 alunos.

Os resultados deste estudo podem orientar os professores na seleção de jogos não-digitais como estratégia instrucional para o ensino de computação e/ou orientar desenvolvedores de jogos em relação a aspectos a serem considerados no desenvolvimento de novos jogos não-digitais para maximizar seus benefícios aos estudantes.

2. Método de Pesquisa

Com o objetivo de analisar os benefícios dos jogos educacionais não-digitais usados para o ensino de computação, foi executada uma série de estudos de caso [Yin, 2017; Wohlin et al., 2012], estruturado como ilustrado na Figura 1.

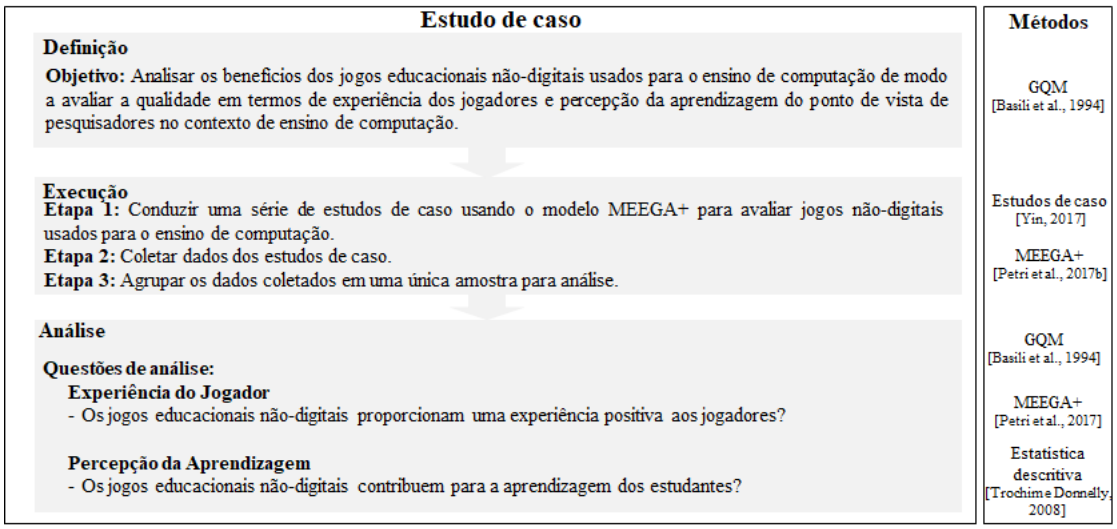


Figura 1. Método de pesquisa

Seguindo a abordagem GQM (*Goal/Question/Metric*) [Basili et al., 1994], na fase de definição, o objetivo do trabalho é decomposto em aspectos de qualidade e questões de análise para serem analisadas com base nos dados coletados nas avaliações dos jogos. O objetivo deste estudo é analisar os benefícios dos jogos educacionais não-digitais usados para o ensino de computação de modo a avaliar a qualidade em termos de experiência dos jogadores e percepção da aprendizagem do ponto de vista de pesquisadores no contexto de ensino de computação.

A fase de execução é organizada em três etapas. Primeiro, foram conduzidos 26 estudos de caso avaliando 9 diferentes jogos educacionais não-digitais para o ensino de computação. Os jogos usados nos estudos de caso foram selecionados de acordo com a sua adequação e relevância para o ensino de conteúdos de computação para o público alvo. Os estudos de caso foram conduzidos adotando o modelo de avaliação de jogos educacionais MEEGA+ [Petri et al., 2017b; Petri et al., 2018b]. O objetivo do modelo MEEGA+ é analisar jogos educacionais de modo a avaliar a percepção da qualidade do ponto de vista de estudantes de computação. O modelo MEEGA+ foi sistematicamente desenvolvido decompondo fatores de qualidade em um conjunto de dimensões que derivaram em um conjunto de itens que compõem um instrumento de medição para operacionalizar a coleta de dados nos estudos de caso, após a aplicação do jogo, adotando um design de pesquisa não-experimental com pós-teste. A decomposição dos fatores de qualidade do modelo MEEGA+ é apresentada na Figura 2.

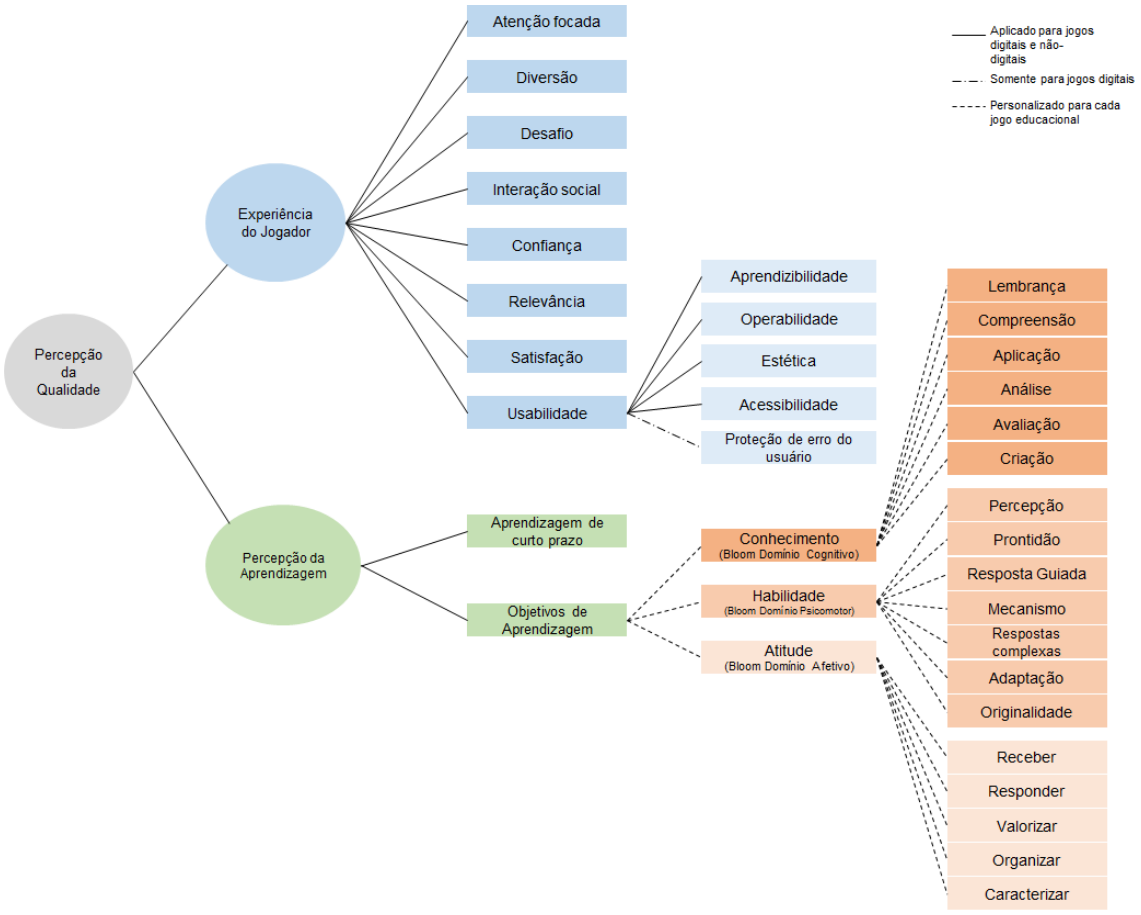


Figura 2. Decomposição dos fatores de qualidade no Modelo MEEGA+ [Petri et al., 2017b]

Cada estudo de caso para a avaliação de um jogo iniciou com a aplicação do tratamento (jogo educacional) e depois da sessão de jogo o questionário do modelo MEEGA+ foi respondido pelos estudantes de modo a coletar suas percepções (Etapa 2). Então, na etapa 3, os dados coletados em todos os estudos de caso conduzidos (Tabela 1) são agrupados em uma única amostra para a análise.

Na fase de análise, foi conduzida uma análise abrangente sobre os benefícios dos jogos. Os dados coletados foram analisados de forma acumulada de modo a sintetizar evidências empíricas sobre os benefícios dos jogos educacionais não-digitais em termos de experiência proporcionada as jogadores e percepção da aprendizagem. Seguindo o modelo MEEGA+ foi analisada cada uma das questões de análise definidas usando métodos de estatística descritiva. Os resultados foram interpretados e discutidos atingindo o objetivo da pesquisa.

3. Definição e Execução do Estudo

O objetivo deste estudo é analisar os benefícios dos jogos educacionais não-digitais usados para o ensino de computação de modo a avaliar a qualidade em termos de experiência dos jogadores e percepção da aprendizagem do ponto de vista de pesquisadores no contexto de ensino de computação. A partir deste objetivo, seguindo o modelo MEEGA+ [Petri et al., 2017b; Petri et al., 2018b], são definidas as seguintes as questões de análise (QA), agrupadas pelos fatores de qualidade avaliados:

Experiência do Jogador

QA1: Os jogos educacionais não-digitais proporcionam uma experiência positiva aos jogadores?

Percepção da Aprendizagem

QA2: Os jogos educacionais não-digitais contribuem para a aprendizagem dos estudantes?

De modo a responder estas questões de análise, foram conduzidos 26 estudos de caso, avaliando 9 jogos educacionais não-digitais em cursos de computação usando o modelo MEEGA+. Todos os 26 estudos de caso foram conduzidos sob supervisão de ao menos um autor deste trabalho. Como resultado, obteve-se dados coletados de 509 estudantes em 4 diferentes instituições de ensino do Brasil e da Espanha, como sumarizado na Tabela 1.

Tabela 1. Resumo dos estudos de caso conduzidos

Jogo	Contexto	Disciplina/Semestre	Instituição/ País	Nº de alunos
PMDome [PMDome, 2017]	Especialização em Gestão de Tecnologia da Informação	Introdução a Gestão de TI/2016-2	Instituto Federal Farroupilha/Brasil	20
	Curso Técnico em Informática	Oficina de Jogos/2016-2	Instituto Federal de Santa Catarina/Brasil	27
Risk Management Game	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos/2016-2	Universidade Federal de Santa Catarina/Brasil	31
	Graduação em Sistemas de Informação	Gestão de Projetos /2016-2		23
	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos /2017-1		21
	Graduação em Sistemas de Informação	Gestão de Projetos /2017-1		21
	Curso Técnico em Informática	Oficina de Jogos /2017-1	Instituto Federal de Santa Catarina/Brasil	36
	Curso Técnico em Informática	Oficina de Jogos /2017-1		31
PMMaster [Gresse von Wangenheim, 2012]	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos /2016-2	Universidade Federal de Santa Catarina/Brasil	24
	Graduação em Sistemas de Informação	Gestão de Projetos /2016-2		21
	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos /2017-1		17
	Graduação em Sistemas de Informação	Gestão de Projetos /2017-1		18
Detective Game – what killed the project? [Gresse von Wangenheim et al., 2014]	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos /2016-2	Universidade Federal de Santa Catarina/Brasil	26
		Planejamento e Gestão de Projetos /2017-1		17
SCRUMIA [Gresse von Wangenheim et al., 2013b]	Graduação em Ciência da Computação	Planejamento e Gestão de Projetos /2016-2	Universidade Federal de Santa Catarina/Brasil	26
		Planejamento e Gestão de Projetos /2017-1		19
Ball Point Game [Gloger, 2017]	Graduação em Engenharia Informática	Workshop Jogos Educacionais/2017-2	Universidade de Cádiz/Espanha	10
Dealing with difficult people [Gresse von Wangenheim et al., 2013a]	Graduação em Engenharia Informática	Workshop Jogos Educacionais/2017-2	Universidade de Cádiz/Espanha	10
SORTIA - Quicksort Tabuleiro [Battistella et al., 2017]	Graduação em Ciência da Computação	Estrutura de Dados/2016-02	Universidade Federal de Santa Catarina/Brasil	21
	Graduação em Sistemas de Informação	Estrutura de Dados/2016-02		6
	Graduação em Ciência da Computação	Estrutura de Dados/2017-01		25
	Graduação em Sistemas de Informação	Estrutura de Dados/2017-01		6
SORTIA - Heapsort	Graduação em Ciência da Computação	Estrutura de Dados/2016-02	Universidade Federal de Santa	17
	Graduação em Sistemas de Informação	Estrutura de Dados/2016-02		7

Tabuleiro [Battistella et al., 2017]	Graduação em Ciência da Computação	Estrutura de Dados/2017-01	Catarina/Brasil	23
	Graduação em Sistemas de Informação	Estrutura de Dados/2017-01		6
Total				**509**

Em síntese, os estudos de caso envolveram 9 diferentes jogos educacionais não-digitais para o ensino de computação. A maioria dos jogos avaliados (7 jogos) concentram-se na área de Engenharia de Software, sendo aplicados, principalmente, em disciplinas de Planejamento e Gestão de Projetos. Estes jogos, tipicamente, são utilizados para revisar conhecimentos por meio de perguntas e respostas, como no jogo PMMaster, simular a execução de um projeto, como nos jogos SCRUMIA, PMDome e *Detective Game* e também, para o desenvolvimento de habilidades interpessoais, como no jogo *Dealing with Difficult People*. Além disso, outros dois jogos avaliados, objetivam simular na prática, em um tabuleiro, a execução dos algoritmos de ordenação Quicksort e Heapsort, em disciplinas de Estruturas de Dados. Uma visão geral de cada jogo avaliado é apresentada na Tabela 2.

Tabela 2. Visão geral dos jogos avaliados e fotos dos estudos de caso

PMDome [PMDome, 2017] é uma dinâmica de simulação que tem como objetivo de aprendizagem transmitir a importância do planejamento e o impacto que a falta do mesmo pode ter. A dinâmica simula as fases de planejamento e execução do gerenciamento de projetos, onde os estudantes precisam planejar o tempo e os recursos (canetas, folhas de papel e fita adesiva) que precisarão para desenvolver o produto final, um Domo Geodésio.

Figura 3. Aplicações da dinâmica PMDome

Risk Managemente Game é um jogo de tabuleiro e objetiva motivar a importância do planejamento de riscos no gerenciamento de projetos. No jogo, os jogadores precisam chegar até a entrega (final do tabuleiro), começando no planejamento e passando pelas Sprints com os recursos que lhe forem entregues no início da partida. No início do jogo os jogadores devem investir seus recursos em cartas de imprevistos (financeiro, material, equipe, concorrência e econômico) que especifica de onde os recursos serão tirados caso não planejado o suficiente. Cartas de Sprint determinam como os jogadores enfrentam os imprevistos na partida.

Figura 4. Aplicações do jogo *Risk Management Game*

PMMaster [Gresse von Wangenheim, 2012] é um jogo de tabuleiro com perguntas sobre gerenciamento de projetos em diferentes áreas de conhecimento, como escopo, tempo e gerenciamento de qualidade. O jogador que primeiro responder corretamente uma questão de cada uma das áreas de conhecimento do PMBOK, vence o jogo. Tem como objetivo de aprendizagem revisar e reforçar os conceitos básicos de gerenciamento de projetos de acordo com o PMBOK (4ª edição), focando especificamente em gestão de projetos de software.

Figura 5. Aplicações do jogo PMMaster

***Detective Game – what killed the project*?** [Gresse von Wangenheim et al., 2014] é um jogo de papel & caneta dedutivo que objetiva aplicar o Gerenciamento de Valor Agregado para o monitoramento e controle de um projeto de software como parte do ensino de gerenciamento de projetos. O objetivo de aprendizagem do jogo é reforçar os conceitos e ensinar a competência de aplicar cálculos básicos sobre o Gerenciamento de Valor Agregado.

Figura 6. Aplicações do jogo Detective

SCRUMIA [Gresse von Wangenheim et al., 2013b] é um jogo de papel & caneta de simulação em grupo com a finalidade de planejar e executar Sprints de um projeto hipotético aplicando SCRUM como parte de um curso de gerenciamento de projetos. O objetivo de aprendizagem da atividade é reforçar os conceitos e ensinar a competência de aplicar o conhecimento sobre o gerenciamento ágil de projetos usando SCRUM.

Figura 7. Aplicações do jogo SCRUMIA

Ball Point Game [Gloger, 2017] objetiva ilustrar a dinâmica de uma equipe trabalhando iterativamente e em melhoria contínua. No jogo, organizado em iterações, as equipes devem passar tantas bolas quanto possível por todos os participantes, até retornarem ao ponto de partida, para marcar pontos. As bolas não podem ser passadas para os vizinhos à direita e à esquerda, não podem tocar o chão e as bolas devem passar pelo ar para o outro jogador. No final das iterações, a equipe que marcou mais pontos é a vencedora.

Figura 8. Aplicações do jogo Ball Point Game

Dealing with difficult people [Gresse von Wangenheim et al., 2013a] é uma dinâmica que objetiva a ilustrar as dificuldades relacionadas à gerência de equipes em projetos de software. Os alunos em grupo simulam uma reunião de *kick-off* de um projeto de software. Um dos alunos assume o papel de gerente de projeto, que conduz a reunião e ao final precisa ter recebido as assinaturas de todos os membros do grupo. Cada membro do grupo recebe uma caracterização de uma personalidade difícil (p.ex. pessoa negativa, explodindo com facilidade, reclamando de tudo) e deve agir de acordo com ela até o momento em que o gerente de projeto começar a reagir de forma adequada.

Figure 9. Aplicações do jogo Dealing with difficult people

SORTIA - Quicksort e Heapsort [Battistella et al., 2017] é um jogo de tabuleiro usado para representar o modelo de memória de uma sequência de chamadas recursivas do algoritmo Quicksort e também permite simular a execução das etapas individuais do algoritmo Quicksort e do algoritmo Heapsort. O jogo é projetado para ser jogado em grupos de 4-6 alunos. No jogo, os alunos precisam ordenar 21 valores (números aleatórios de 0-100) em uma matriz, de acordo com o algoritmo Quicksort/Heapsort.

Figura 10. Aplicações do jogo SORTIA – Quicksort/Heapsort

No total de 26 estudos de caso conduzidos, participaram do estudo 509 alunos, 430 homens (85%), 76 mulheres (15%) e 3 não responderam. A maioria dos alunos possui entre 18 e 28 anos (74%), 18% possuem menos de 18 anos, e 36 alunos possuem acima de 29 anos. A maioria dos participantes (56%) indica que costuma jogar jogos não-digitais raramente, 21% jogam ao menos uma vez ao mês, 11% jogam semanalmente, 8% nunca jogaram e, 4% dos alunos afirmam que costumam jogar diariamente.

4. Análise dos dados

Os dados coletados nos estudos de caso (Tabela 1) foram agrupados em uma única amostra, utilizando-os cumulativamente para analisar os benefícios dos jogos (e não para avaliar um jogo específico). Os estudos conduzidos são semelhantes em termos de definição (com o objetivo de avaliar um jogo não-digital para o ensino de computação em relação à experiência do jogador e percepção de aprendizagem), design de pesquisa (estudos de caso) e contexto (ensino de computação). Além disso, todos os estudos de caso conduzidos são padronizados em termos de medidas (fatores/dimensão de qualidade), método de coleta de dados (questionário do modelo MEEGA+) e formato de resposta (Escala Likert de 5 pontos) variando de -2 (discordo fortemente) a 2 (concordo fortemente).

Os dados coletados são analisados usando estatística descritiva em termos de distribuição de frequência e tendência central (mediana) para responder a cada uma das questões de análise definidas.

QA1: Os jogos educacionais não-digitais proporcionam uma experiência positiva aos jogadores?

De acordo com o modelo MEEGA+, a experiência proporcionada aos jogadores é avaliada em termos de usabilidade do jogo, confiança, desafio, satisfação, interação social, diversão, atenção focada e relevância. Assim, analisando os dados coletados em relação a este fator de qualidade, identificou-se que os jogos educacionais não-digitais, de forma geral, proporcionam uma experiência positiva aos estudantes de computação (Figura 11), em especial, em termos de interação social, diversão e relevância. Isso demonstra que os estudantes sentiram os jogos como uma estratégia de aprendizado positiva e engajadora.

Analisando a usabilidade dos jogos avaliados, em geral, os alunos a avaliaram de forma positiva, indicando que os jogos possuem uma boa usabilidade, que as fontes usadas nos jogos são legíveis, as cores são compreensíveis, os textos, cores e fontes combinam e são consistentes e também que as regras são claras. Em especial, os jogos PMMaster e *Detective Game* demostraram um alto grau de usabilidade, sendo avaliados com um design atraente e consistente. Os alunos também avaliaram de forma bastante positiva a facilidade de aprender a jogar o jogo *Ball Point Game*, indicando que as regras estavam claras e bastante compreensíveis, o que facilitou a iniciar a jogá-lo.

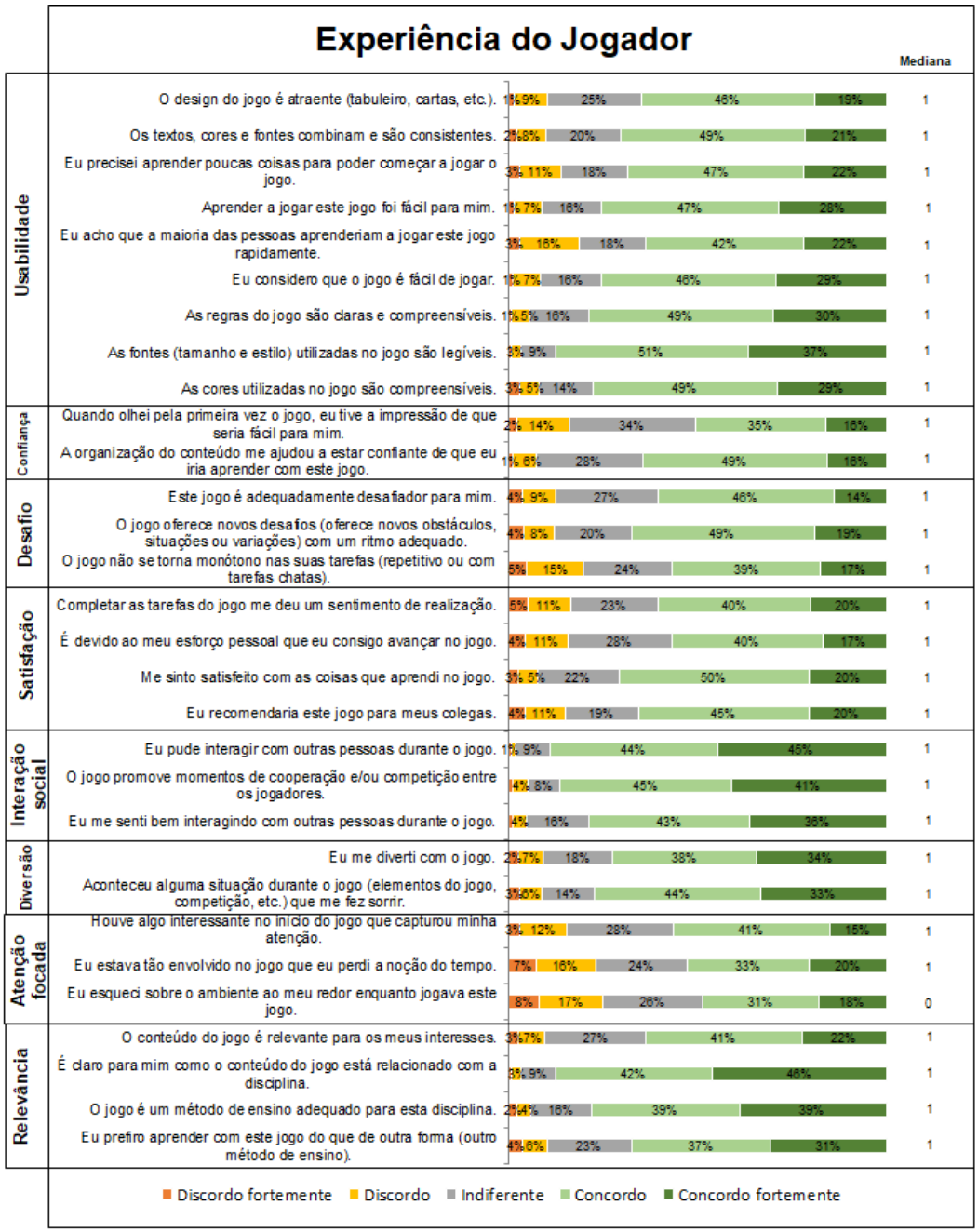


Figura 11. Diagrama de frequência e medianas da avaliação da experiência do jogador baseado no total de 509 respostas

No entanto, identificou-se que os itens relacionados a capacidade de aprender a jogar os jogos, em geral, tiveram uma avaliação mais baixa, assim, indicando que os jogos não foram tão fáceis de aprender a jogar e que possivelmente as pessoas não aprenderiam a jogar rapidamente. Um exemplo está no jogo *Risk Management Game*, onde os alunos devem tomar decisões iniciais de investimento de seus recursos em cartas de imprevistos (financeiro, material, equipe, concorrência e econômico) para usarem em toda partida do jogo, no entanto, o grande número de regras do jogo, pode fazer com que os alunos tomem decisões sem entender completamente o funcionamento global do jogo e, assim, o avaliaram como difícil de aprender a jogar. Outro exemplo

está no jogo SORTIA, onde exige que os alunos saibam as etapas de execução dos algoritmos Heapsort e Quicksort, sendo indicado como difícil para as pessoas aprenderem a jogar rapidamente, pois, de fato, necessita um conhecimento prévio dos algoritmos citados. Em síntese, estes resultados, podem também estar relacionado as características deste tipo de jogos, que como são jogos não-digitais, no início da partida o instrutor deve explicar as regras e a mecânica do jogo, e assim, a forma como o instrutor as explica pode fazer com que os jogadores, em um primeiro momento, vejam o jogo com mais dificuldade, até entenderem todas as regras e começarem a jogar.

De igual modo, analisando a confiança dos jogadores ao jogar, embora avaliado de forma positiva pela maioria, alguns estudantes também indicaram que ao olhar o jogo pela primeira vez, tiveram impressão de que não seria fácil de jogar. Porém, as regras compreensíveis e a organização do conteúdo do jogo ajudaram os estudantes a estarem confiantes de que iriam aprender com o jogo.

Analisando os desafios proporcionados pelos jogos, a maioria dos estudantes os avaliou de forma positiva, indicando que os jogos são desafiadores, que os jogos vão oferecendo novos desafios aos jogadores em um ritmo adequado e que, de forma geral, os jogos não se tornam repetitivos ou com tarefas monótonas. No entanto, cabe destacar, que este último item também teve avaliações baixas, ou seja, alguns jogos foram percebidos monótonos em suas atividades. Um exemplo é o jogo PMDome, onde os alunos tinham que criar 65 tubos de papel para a montagem do Domo Geodésio, sendo avaliado como uma tarefa repetitiva pelos alunos. Deste modo, este resultado mostra que é importante avaliar o jogo para, de fato, identificar seus pontos fracos e então, melhorá-lo de modo a torná-lo mais atrativo e divertido para os alunos, minimizando as tarefas que são consideradas monótonas e/ou repetitivas pelos alunos.

Em termos de satisfação proporcionada pelos jogos, os alunos confirmaram que estão satisfeitos com os conteúdos que aprenderam jogando e recomendariam os jogos para seus colegas. Além disso, a maioria dos alunos também confirmou que conseguiram avançar no jogo devido ao seu esforço pessoal – o que corresponde a um elemento essencial de um jogo educacional, que só deve permitir que os alunos vençam se alcançaram os respectivos objetivos de aprendizagem [Abt, 2002].

A interação social foi a dimensão melhor avaliada em todos os jogos. A maioria dos estudantes concordou fortemente que os jogos promoveram momentos de cooperação e competição entre os jogadores, confirmando também que se divertiram ao interagir com outras pessoas durante o jogo. Esse comportamento pôde ser observado durante as aplicações dos jogos, o que aconteceu de forma muito agradável, proporcionando um sentimento de ambiente compartilhado. A interação social promovida pelos jogos em grupos, em vários casos, também estimulou os alunos a se familiarizarem também em ambientes fora da sala de aula.

A diversão proporcionada pelos jogos, também foi uma das dimensões melhor avaliadas pelos estudantes, confirmando que eles se divertiram jogando e que alguma situação dos jogos os fizeram sorrir. Esta característica dos jogos também é importante, pelo fato de proporcionar um ambiente mais agradável para os alunos, ao mesmo tempo que estão aprendendo enquanto estão jogando [Abt, 2002].

Analisando a atenção focada proporcionada pelos jogos, embora também avaliada de forma positiva, cabe destacar as avaliações neutras e negativas para estes itens. Os jogos não-digitais, por usarem tabuleiros, cartas e papeis, normalmente com

um design simples, em um primeiro momento, pode não capturar a atenção dos estudantes de computação que, em geral, podem estar acostumados com os gráficos 3D de jogos digitais que costumam jogar [Battistella e Gresse von Wangenheim, 2016b]. Assim, é importante que outros fatores dos jogos (como desafio, interação social e diversão) sejam explorados para que os jogadores possam imergir na atividade de aprendizagem perdendo a noção do entorno e focando nas atividades do jogo.

A relevância dos jogos para os estudantes também foi avaliada de forma bastante positiva. Os alunos claramente perceberam que os conteúdos abordados nos jogos estão relacionados com a disciplina em que o jogo foi utilizado. A maioria dos alunos aponta que os jogos são um método de ensino adequado para a disciplina e que preferem aprender jogando, mesmo que alguns alunos indicam que prefiram aprender com outros métodos de ensino. Este resultado mostra que os alunos, apesar de estarem mais acostumados com jogos digitais, vivenciaram uma experiência positiva e relevante para a aprendizagem dos conteúdos envolvidos no jogo. Deste modo, os jogos educacionais não-digitais podem ser uma estratégia instrucional que contribui positivamente na experiência do aluno em sala de aula.

QA2: Os jogos educacionais não-digitais contribuem para a aprendizagem dos estudantes?

Analisando a percepção de aprendizagem promovida pelos jogos avaliados, em geral, pode-se observar que os jogos contribuíram para a aprendizagem dos conteúdos de computação (Figura 12). Os estudantes também indicaram que os jogos não-digitais usados para o ensino de computação foram eficientes para a sua aprendizagem, quando comparado a outras atividades realizadas na disciplina em que o jogo foi aplicado. No entanto, algumas avaliações baixas também foram identificadas. Como por exemplo, no jogo *Risk Management Game*, que possui como objetivo de aprendizagem motivar os alunos sobre a importância do planejamento de riscos no gerenciamento de projetos e, de fato, voltado a aprendizagem de atitude e não de conhecimento. Deste modo, este resultado pode estar indicando que alguns alunos podem preferir outros métodos de ensino.

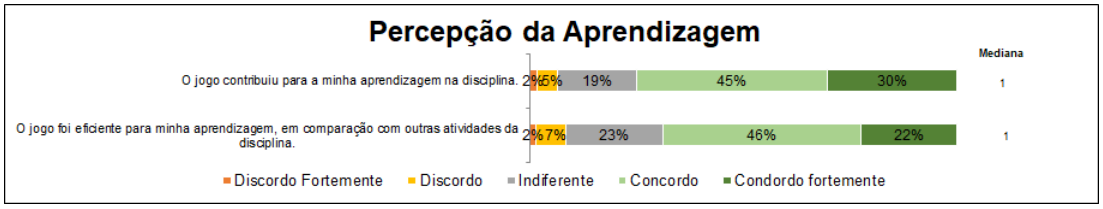


Figura 12. Diagrama de frequência e medianas da avaliação da percepção da aprendizagem

Adicionalmente, os objetivos de aprendizagem de cada jogo também foram avaliados pelos estudantes. Os resultados da avaliação de cada jogo são apresentados na Tabela 3.

Tabela 3. Resultados da avaliação dos objetivos de aprendizagem

Jogo	Objetivos de aprendizagem	Mediana
PMDome	O jogo contribuiu para reconhecer a importância da gestão (de material, equipe, tempo, riscos, etc.).	2
Risk Management Game	O jogo contribuiu para reconhecer a importância do planejamento de riscos.	1
PMMaster	O jogo contribuiu para reforçar e fixar os conceitos sobre os grupos de processos de gerenciamento de	1

	projetos (iniciação, planejamento, etc.).	
	O jogo contribuiu para reforçar e fixar os conceitos sobre as áreas de conhecimento de gerenciamento de projetos (escopo, tempo, custo, etc.).	1
Detective Game	O jogo contribuiu para compreender a técnica de valor agregado.	1
	O jogo contribuiu para aplicar na prática a técnica de valor agregado.	1
	O jogo contribuiu para compreender os processos de monitoramento e controle.	1
SCRUMIA	O jogo contribuiu para relembrar os conceitos sobre o Planejamento de uma Sprint.	1
	O jogo contribuiu para relembrar os conceitos sobre o Monitoramento de uma Sprint.	1
	O jogo contribuiu para relembrar os conceitos sobre a Reunião de revisão de uma Sprint.	1
	O jogo contribuiu para relembrar o que é o Taskboard.	1
	O jogo contribuiu para diferenciar os papéis, reuniões e artefatos envolvidos no Planejamento de uma Sprint.	1
	O jogo contribuiu para diferenciar os papéis, reuniões e artefatos envolvidos no Monitoramento de uma Sprint.	1
	O jogo contribuiu para diferenciar os papéis, reuniões e artefatos envolvidos na Reunião de revisão de uma Sprint.	1
	O jogo contribuiu para diferenciar a estrutura de organização do Taskboard.	1
	O jogo contribuiu para praticar o Planejamento de uma Sprint.	1
	O jogo contribuiu para praticar o Monitoramento de uma Sprint.	1
	O jogo contribuiu para praticar a Reunião de revisão de uma Sprint.	1
	O jogo contribuiu para praticar a organização de um Taskboard.	1
Ball Point Game	O jogo contribuiu para perceber a importância do trabalho em equipe para alcançar os objetivos.	2
	O jogo contribuiu para perceber a importância da re (organização) em uma equipe de trabalho.	2
	O jogo contribuiu para perceber a importância da comunicação em uma equipe de trabalho.	2
	O jogo contribuiu para perceber a importância da avaliação do trabalho para sua melhora contínua.	2
Dealing with difficult people	O jogo contribuiu para perceber as dificuldades colocadas pelas diferentes personalidades que podem existir em uma equipe de trabalho.	2
	O jogo contribuiu para perceber como a personalidade dos membros de uma equipe influencia na produtividade da equipe de trabalho.	2
	O jogo contribuiu para perceber a importância de saber gerenciar as diferentes personalidades dos membros em uma equipe de trabalho.	1
	O jogo contribuiu para perceber a importância da comunicação dentro de uma equipe de trabalho na hora de gerar ideias.	2
SORTIA - Quicksort Tabuleiro	O jogo contribuiu para reforçar os conhecimentos sobre o Algoritmo Quicksort.	1
	O jogo contribuiu para compreender o Algoritmo Quicksort.	1
	O jogo contribuiu para aplicar na prática os conhecimentos sobre o Algoritmo Quicksort.	1
SORTIA - Heapsort Tabuleiro	O jogo contribuiu para reforçar os conhecimentos sobre o Algoritmo Heapsort.	2
	O jogo contribuiu para compreender o Algoritmo Heapsort.	2
	O jogo contribuiu para aplicar na prática os conhecimentos sobre o Algoritmo Heapsort.	2

Analisando os resultados dos objetivos de aprendizagem, pode-se observar que, de forma geral, todos os 9 jogos educacionais não-digitais aplicados foram avaliados de forma bastante positiva, indicando assim, uma percepção por parte dos alunos que os jogos alcançaram seus objetivos de aprendizagem. No entanto, observa-se uma avaliação levemente mais positiva referente aos jogos PMDome, *Ball Point Game* e *Dealing with Defficult People*. Estes jogos possuem como objetivo de aprendizagem motivar a importância do planejamento e trabalhar habilidades de trabalho em grupo, comunicação e gestão de equipes, assim, podem ter sido avaliados de forma mais positiva pelos alunos pelo fato de desenvolver, em forma de jogo, habilidades importantes para os estudantes de computação. De forma similar, os alunos também concordaram fortemente que o jogo SORTIA-Heapsort alcançou os objetivos de aprendizagem de reforçar, compreender e aplicar na prática os conhecimentos sobre o algoritmo Heapsort. Este resultado, quando comparado ao resultado do jogo SORTIA-Quicksort, pode estar relacionado aos conhecimentos prévios dos alunos sobre o

algoritmo Heapsort, necessários para a execução do jogo. Em síntese, os jogos não-digitais, além de serem um método de ensino que proporciona uma experiência positiva aos alunos, também contribui para a aprendizagem dos estudantes de computação.

4.1 Ameaças à validade

Devido às características deste tipo de pesquisa, este trabalho está sujeito a ameaças à validade. Deste modo, foram identificadas ameaças potenciais e aplicadas estratégias de mitigação para minimizar o impacto no trabalho.

Algumas ameaças estão relacionadas ao design de pesquisa [Wohin et al., 2012]. Considerando o contexto deste estudo, relacionado a ensino de computação, isso significa que deve ser possível realizar a avaliação do jogo de forma rápida e não intrusiva, de modo a não interromper o fluxo normal das aulas. Portanto, optou-se por conduzir uma série de estudos de caso de avaliação que permitem realizar uma pesquisa aprofundada de um indivíduo, grupo ou evento (Yin, 2014; Wohlin et al., 2012). Experimentos, por outro lado, causariam uma interrupção maior nas aulas, além de exigir a definição de grupos de controle que podem ser prejudicados ao usar métodos de ensino alternativos considerados inferiores. Além disso, para obter resultados estatisticamente significativos de tais experimentos, é necessário um tamanho de amostra considerável, o que dificultaria a condução devido ao número reduzido de alunos comumente matriculados em disciplinas de computação.

Outro risco se refere a confiabilidade e validade dos dados coletados nos estudos de caso de avaliação. Para minimizar este risco, todos os estudos de caso foram conduzidos adotando o modelo MEEGA+ [Petri et al., 2017b; Petri et al., 2018b]. O MEEGA+ é um modelo sistematicamente desenvolvido para a avaliação de jogos educacionais e amplamente avaliado em termos de validade e confiabilidade [Petri et al., 2018b].

Outra ameaça está na avaliação da aprendizagem, como não houve um pré-teste medindo, por exemplo, o nível de conhecimento dos alunos antes da aplicação dos jogos, não foi possível identificar com exatidão o quanto cada aluno aprendeu sobre os conteúdos abordados nos jogos. No entanto, embora sem consenso, existem evidências de que a auto avaliação fornece informações confiáveis, válidas e úteis para este tipo de estudo [Sitzmann et al., 2010].

Em termos de validade externa, uma ameaça à possibilidade de generalizar os resultados está relacionada ao tamanho da amostra e à diversidade dos dados utilizados para a avaliação. Com relação ao tamanho da amostra, a avaliação usou dados coletados de 26 estudos de caso que avaliaram 9 diferentes jogos não-digitais, envolvendo uma população de 509 alunos. Em termos de significância estatística, este é um tamanho de amostra satisfatório que permite a geração de resultados significativos [Sitzmann et al., 2010]. Os dados foram obtidos a partir de aplicações de jogos em 4 diferentes instituições de ensino. No entanto, como a coleta de dados foi restrita às avaliações de jogos não-digitais para o ensino de computação avaliados com o modelo MEEGA+, a maioria dos dados é do Brasil, onde é usado de forma mais proeminente, com apenas duas aplicações conduzidas pelos autores em uma universidade da Espanha.

Em termos de confiabilidade, uma ameaça refere-se a que medida os dados e a análise dependem dos pesquisadores específicos. Para mitigar essa ameaça, foi

documentada uma metodologia de pesquisa sistemática, definindo claramente o objetivo do estudo, o processo de coleta e análise de dados.

5 . Conclusão

De modo a obter uma compreensão mais abrangente sobre os benefícios dos jogos educacionais não-digitais usados para o ensino de computação, uma série de estudos de caso aplicando e avaliando jogos não-digitais foi conduzida. Os resultados da análise de 26 estudos de caso, envolvendo 509 alunos e avaliando 9 diferentes jogos para o ensino de computação, fornecem evidências sobre os benefícios dos jogos educacionais não-digitais que podem produzir um efeito positivo na aprendizagem dos estudantes de computação, proporcionando uma experiência agradável e envolvente aos alunos e motivando-os ao estudo.

Deste modo, pode-se concluir que os jogos educacionais não-digitais podem ser uma alternativa efetiva para o ensino de computação, quando comparado ao custo/benefício e esforço despendido para o desenvolvimento de jogos digitais [Petri et al., 2017]. Principalmente por facilmente possibilitar uma interação social entre os alunos, promovendo um ambiente agradável e divertido para a aprendizagem, estimulando a cooperação e compartilhamento de ideias entre os estudantes. Adicionalmente, os resultados também apontaram que para ter estes benefícios positivos é importante desenvolver jogos com tarefas que não sejam monótonas e/ou repetitivas, mas sim, que proporcionam desafios, diversão, competitividade e cooperação entre os jogadores.

Como trabalhos futuros pretende-se continuar a pesquisa de avaliação de jogos não-digitais em diferentes contextos, de forma a confirmar os resultados deste estudo e também identificar novas oportunidades neste tipo de jogos de modo a contribuir para sua efetiva e contínua melhoria.

Agradecimentos

Gostaríamos de agradecer aos alunos de todas as instituições de ensino que participaram das aplicações dos jogos.

Este trabalho foi realizado no âmbito do Programa de Doutorado Sanduíche no Exterior (PDSE) na Universidade de Cádiz, Espanha, patrocinado pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), fundação do Ministério da Educação, Brasil (nº 88881.131485 / 2016- 01).

Este trabalho foi apoiado pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), uma entidade do governo brasileiro focada no desenvolvimento científico e tecnológico.

Este trabalho também foi apoiado pela Agencia Española de Investigación (AEI) com fundos FEDER no âmbito do projeto BadgePeople (TIN2016-76956-C3-3-R) e do Plano Andaluz de Pesquisa, Desenvolvimento e Inovação (TIC-195).

Referências

Abt, C. C. (2002). Serious Games. Lanhan: University Press of America.

Baker, A., Navarro, E. O, e van der Hoek, A. (2005). An experimental card game for teaching software engineering processes. Journal of Systems and Software, 75(1–2), 3-16.

Basili, V. R., Caldiera, G., e Rombach, H. D. (1994). Goal, Question Metric Paradigm. In J. J. Marciniak, Encyclopedia of Software Engineering. New York, USA.

Battistella, P. E. e Gresse von Wangenheim, C. (2016a). Games for teaching computing in higher education - A systematic review. IEEE Technology and Engineering Education, 9(1), 8-30.

Battistella, P. E. e Gresse von Wangenheim, C. (2016b). Caracterização do Público-Alvo de Jogos Educacionais na área da Computação. 24º Workshop sobre Educação em Computação, (pp. 2016-2025). Porto Alegre, RS, Brasil.

Battistella, P. E., Gresse von Wangenheim, C., von Wangenheim, A., e Martina, J. E. (2017). Design and Large-scale Evaluation of Educational Games for Teaching Sorting Algorithms. Informatics in Education, 17(2), 141-164.

Calderón A. e Ruiz M. (2015). A systematic literature review on serious games evaluation: An application to software project management. Computers & Education, 87, 396-422.

Çiftci, S. (2018). Trends of Serious Games Research from 2007 to 2017: A Bibliometric Analysis. Journal of Education and Training Studies, 6(2), 18-27.

Fernandes, J. M., Sousa, S. M. (2010). PlayScrum - A Card Game to Learn the Scrum Agile Method. International Conference on Games and Virtual Worlds for Serious Applications, (pp.52-59). Braga, Portugal.

Gloger, B. (2017). Ball Point Game. Disponível em: <https://borisgloger.com/wp-cotent/uploads/2016/08/Ball_Point_Game.pdf> Acesso em: 27 novembro 2017.

Gresse von Wangenheim, C. (2012). PMMaster. Disponível em: <http://www.gqs.ufsc.br/pm-master/> Acesso em: 06 fevereiro 2018.

Gresse von Wangenheim, C., Carvalho, O. P., e Battistella, P. E. (2013a) Ensinar a Gerência de Equipes em Disciplinas de Gerência de Projetos de Software. Revista Brasileira de Informática na Educação, 21(1), 15-22.

Gresse von Wangenheim, C., Rausis, B. Soares, G., Savi, R., e Borgatto, A. F. (2014). Project Detective A Game for Teaching Earned Value Management. International Journal of Teaching and Case Studies, 5(3/4), 216-234.

Gresse von Wangenheim, C., Savi, R., e Borgatto, A. F. (2013b). SCRUMIA - An educational game for teaching SCRUM in computing courses. Journal of Systems and Software, 86(10), 2675-2687.

Hamey, L. G. C. (2003). Using the Security Protocol Game to teach computer network security. Symposium on the Improving Learning Outcomes Through Flexible Science Teaching, (pp. 96-101). Sydney, Australia.

Petri, G. e Gresse von Wangenheim, C. (2016). How to Evaluate Educational Games: a Systematic Literature Review. Journal of Universal Computer Science, 22(7), 992-1021.

Petri, G. e Gresse von Wangenheim, C. (2017). How games for computing education are evaluated: a systematic literature review. Computers & Education, vol. 107, 68-90.

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2017a). Quality of Games for Teaching Software Engineering: An Analysis of Empirical Evidences of Digital and Non-digital Games. Proc. of the 39th Int. Conf. on Software Engineering: Software Engineering Education and Training Track, (pp. 150-159). Buenos Aires, Argentina.

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2017b). Evolução de um Modelo de Avaliação de Jogos para o Ensino de Computação. 25° Workshop sobre Educação em Computação, (pp. 2327-2336). São Paulo, SP, Brasil.

Petri, G., Gresse von Wangenheim, C., Borgatto, A. F., Calderón, A., e Ruiz, M. (2018a). Digital Games for Computing Education: What are the Benefits? In Krassmann, A. L. et al. (Eds). Handbook of Research on Immersive Digital Games in Educational Environments. IGI Global, cap. 2 (aceito para publicação).

Petri, G., Gresse von Wangenheim, C., e Borgatto, A. F. (2018b). MEEGA+, Systematic Model to Evaluate Educational Games. In Newton Lee (Eds) Encyclopedia of Computer Graphics and Games, (pp. 1-7). Cham: Springer.

Pfahl, D., Ruhe, G., e Koval, N. (2001). An experiment for evaluating the effectiveness of using a system dynamics simulation model in software project management education. International Symposium on Software Metrics, (pp.97-109). London, GB.

PMDome. (2017). PMDome Workshop. Disponível em: <https://ricardo-vargas.com/pt/workshops/pmdome/> Acesso em: 27 novembro 2017.

Singh, J., Dorairaj, S. K., e Woods, P. (2007). Learning computer programming using a board game - case study on C-Jump. Symposium on Information and Communications Technologies. Kuala Lumpur, Malaysia.

Sitzmann, T., Ely, K., Brown, K. G., e Bauer, K. N. (2010). Self-Assessment of Knowledge: A Cognitive Learning or Affective Measure? Academy of Management Learning & Education, 9(2), 169-191.

Souza, M. e França, C. (2016). O que explica o sucesso de jogos no ensino de engenharia de software? Uma teoria de motivação. 24º Workshop sobre Educação em Computação, (pp. 2255-2263). Porto Alegre, RS, Brasil.

Tahir, R. e Wangmar, A. I. (2017). State of the art in Game Based Learning: Dimensions for Evaluating Educational Games. Proc. of the European Conference on Games Based Learning, (pp. 641-650). Graz, Austria.

Trochim, W. M. e Donnelly, J. P. (2008). Research methods knowledge base (3 ed.). Mason: Atomic Dog Publishing.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., eWesslén, A. (2012). Experimentation in Software Engineering. Springer-Verlag Berlin Heidelberg.

Yin, R. K. (2017). Case study research and applications: design and methods (5 ed.). Beverly Hills: Sage Publications.

Análise da Evasão de Alunos na Licenciatura em Computação

Viviane Vasconcelos[1], **Ermeson Andrade**[1]

[1]Departamento de Informática e Estatística (DEINFO)
Universidade Federal Rural de Pernambuco(UFRPE)
Pernambuco, Recife, 50670-901

{viviane.vilela,ermeson.andrade}@ufrpe.br

Abstract. *The lack of a proper background in logic and mathematics by students in Computing Licenciate and the volume of new ideas required of them in the first semesters of the course appear as the main reasons behind the high dropout rates. However, studies conducted rarely propose solutions. Through questionnaires given to teachers and students in the course, the current paper collects data about the perceptions of both groups at the Universidade Federal Rural de Pernambuco, and identifies a few problems that can be solved as a way to lower the dropout rates. Solutions are offered, and can be used in other institutions that face the same dropout problem.*

Resumo. *A falta de embasamento lógico-matemático dos alunos de Licenciatura em Computação e o volume de ideias novas requeridas nos primeiros semestres do curso aparecem como motivos principais da alta taxa de evasão. No entanto, os estudos realizados raramente sugerem soluções para o problema. Através de questionários aplicados a professores e alunos do curso, o presente trabalho levanta dados sobre as percepções de ambos grupos integrantes do curso na Universidade Federal Rural de Pernambuco, e identifica alguns problemas que podem ser corrigidos como forma de diminuir a alta taxa de evasão do curso. Soluções são oferecidas, e podem ser utilizadas em outras instituições que também enfrentam o problema da alta taxa de evasão.*

1. Introdução

O avanço e disseminação das tecnologias tem criado e facilitado oportunidades em todos os contextos em que estão inseridas. Cada vez mais o mercado de trabalho exige profissionais aptos a lidarem com as tecnologias em todas as áreas de atuação. Cursos superiores voltados à tecnologia vêm crescendo, e junto com essa ocorrência, tem aumentado a necessidade de formar profissionais aptos a ensinar Computação. Isso tem feito com que várias universidades de todo o país incluam em seus catálogos o curso de Licenciatura em Computação(LC). Como outras Licenciaturas, o curso de LC é regido pela Lei de Diretrizes e Bases, que determina a formação de professores da Educação Básica através da graduação em Licenciatura Plena, e a oferta da Licenciatura em Computação vem aumentando gradativamente, à medida que surgem novas possibilidades no Ensino da Computação.

No entanto, existe uma alta taxa de evasão nos cursos de LC. Gomes, Henriques e Mendes (2008) ressaltam a disparidade entre os métodos de ensino utilizados por grande parte dos professores e os estilos de aprendizado diversificados dos alunos. Similarmente, os métodos de estudo dos alunos também precisam ser modificados, uma vez

que os alunos geralmente estão acostumados a memorizar conceitos e fórmulas, o que não é eficiente no aprendizado da programação. O alto nível de abstração requerido dos alunos em disciplinas introdutórias, como na Introdução à Computação, e uma falta de preparo lógico-matemático por parte dos alunos ingressantes no curso também aparecem como fatores relevantes para a alta taxa de evasão.[Gomes et al. 2008] Júnior e Rapkiewicz (2004) mostram que pesquisas realizadas na área apontam uma preocupação em listar e entender as razões que expliquem porque as disciplinas introdutórias de cursos de Computação possuem as mais altas taxas de reprovação quando comparadas a outros cursos da área de Exatas, no entanto poucas soluções são oferecidas.

Na Universidade Federal Rural de Pernambuco (UFRPE), embora a oferta de vagas para o curso de LC seja 40 por semestre letivo, esse número começa a cair já a partir do primeiro semestre. À medida que os semestres progridem, é comum ver disciplinas com somente um ou dois alunos matriculados, porque muitos alunos reprovam cadeiras que prendem outras disciplinas na matriz curricular, ou perdem o interesse no curso devido a essas dificuldades iniciais. Com isso em mente, e levando em conta a facilidade de acesso aos alunos e professores do curso de LC na UFRPE, o presente estudo foi executado, para levantar dados sobre possíveis razões para a alta taxa de evasão no curso, e assim propor algumas possíveis soluções que podem ser estendidas a outros cursos de LC em outras universidades do país.

O artigo está organizado da seguinte forma: a seção 2 apresenta o embasamento teórico para o estudo; a seção 3 constitui-se de uma revisão da literatura relacionada; a seção 4 aborda a metodologia utilizada no estudo, enquanto que a seção 5 apresenta os resultados e discussões, assim como propostas de solução. Por fim, a seção 6 apresenta conclusões a serem implementadas não só na UFRPE como em outras instituições de ensino.

2. Fundamentos

O curso de LC Na UFRPE tem como objetivo formar professores aptos a abordarem os conteúdos da ciência da computação no ensino Fundamental, Médio e Técnico. São ofertadas 40 vagas por semestre, e o ingresso ocorre através do Exame Nacional do Ensino Médio (ENEM). A matriz curricular está organizada em nove períodos noturnos cujo conteúdo contempla disciplinas obrigatórias na área de Computação assim como disciplinas de Educação, como Fundamentos da Educação, Didática, e Psicologias I e II. No primeiro período, os alunos precisam cursar as seguintes disciplinas obrigatórias: Introdução à Programação, Matemática Discreta, Elementos de Informática, Psicologia I, Fundamentos Filosóficos, Históricos, e Sociológicos da Educação. No segundo período, as disciplinas obrigatórias são: Algoritmos e Estruturas de Dados, Cálculo A I, Introdução à Teoria da Computação, Prática de Ensino de Algoritmos, e Psicologia II. Existe o pareamento entre a base teórica específica na área estudada com uma disciplina de Prática de Ensino, que, no entanto, não são oferecidas pelo Departamento de Educação, mas por professores do próprio departamento em que a LC se encontra inserida.

De acordo com especificações do Departamento de Estatística e Informática (DEINFO) da UFRPE, o curso de LC forma profissionais habilitados ao ensino da Computação no Ensino Fundamental, Médio e Técnico. Algumas das habilidades pretendidas são:

- Desenvolver atividades de pesquisa e docência em computação e educação;
- Conduzir investigações e contribuir para o desenvolvimento do conhecimento na área de computação de maneira multi, inter, e transdisciplinar;
- Analisar problemas educacionais;
- Atuar no planejamento e execução de currículos e programas de capacitação profissional, em organizações diversas, e em especial aquelas relacionadas a métodos, processos e técnicas da Computação;
- Projetar e implementar ferramentas e soluções computacionais de apoio aos processos de ensino-aprendizagem e de administração escolar para atender as demandas das escolas e instituições de ensino;
- Contribuir para a geração de inovações nos processos de ensino e aprendizagem de maneira a atender as demandas de formação de educadores comprometidos com a transformação social e tecnológica.

Os alunos ingressantes em LC têm experiências diversificadas nas disciplinas introdutórias da computação: enquanto alguns apresentam facilidade no aprendizado e compreensão dos conteúdos, outros enfrentam muitas dificuldades nessas disciplinas[Ambrósio et al. 2011]. Dos Santos e Costa (2006) ressaltam que as disciplinas de Programação e Algoritmos constituem a base para a Ciência da Computação, e seus conteúdos contemplam o ensino de linguagens de programação assim como os conceitos, princípios, e modelos de programação; a programação em si consiste na aplicação direta da resolução de problemas. Nesse sentido, as linguagens de programação tornam-se ferramentas para a concretização da resolução de um problema, que resulta no software em si[dos Santos and Costa 2006].

Nota-se que as disciplinas introdutórias do primeiro e segundo período do curso requerem dos alunos um embasamento teórico superior ao conhecimento dos alunos ingressantes. Devido à extensão dos tópicos introduzidos nas disciplinas de Introdução à Programação, Matemática Discreta, e Algoritmos e Estruturas de Dados, a maioria dos alunos ingressantes precisa cursar as disciplinas mais de uma vez para conseguirem absorver os conteúdos e atingirem notas satisfatórias. As reprovações ocasionam na desmotivação dos alunos ingressantes, assim como uma desconfiança por parte da comunidade acadêmica, o que pode diminuir o número de alunos interessados nos cursos de LC [SILVA et al. 2009]. Considerando que a maioria dos alunos ingressantes no curso de Licenciatura em Computação não conhece uma linguagem de programação antes de iniciarem o curso, talvez seria mais importante construir uma base sólida nesses alunos novatos do que almejar que eles adquiram as habilidades em dois semestres.

3. Trabalhos Relacionados

Prietz e Pazeto (2009) apresentam um estudo de caso aplicado ao curso de Licenciatura Plena em Informática na Universidade Federal do Mato Grosso. Dados sobre o ensino, atividades de pesquisa e extensão são apresentados de modo a caracterizarem o curso e a análise quantitativa referente às experiências dos alunos. As autoras ressaltam que Programação I é a disciplina com menor índice de aprovação, e que a média de alunos formados no curso é de onze alunos [Prietch and Pazeto 2009].

Gomes, Henriques e Mendes (2008) destacam que os elevados níveis de insucesso em disciplinas onde são ensinados os conceitos mais básicos de programação, em qualquer grau e sistema de ensino, é um problema universal que tem sido alvo de variadas

pesquisas, resultando também em diversificados sistemas, sem que contudo o panorama tenha melhorado significativamente. Existe um conjunto de razões que estão na origem do problema, nomeadamente, métodos de ensino e aprendizagem desadequados, falta de vários tipos de competências por parte dos alunos, em particular no que respeita à resolução de problemas, a difícil natureza do tema e uma forte conotação negativa que lhe está associada. As metodologias tradicionalmente utilizadas para aprender/ensinar estes assuntos não são suficientes [Gomes et al. 2008].

Em contrapartida, Ambrósio et. al (2011) descrevem tais dificuldades tomando o discurso de professores e alunos, contrastando os alunos com bom e fraco rendimento. Neste sentido, após um levantamento da literatura internacional na área, os autores descrevem as percepções de professores e alunos sobre as exigências cognitivas e acadêmicas da aprendizagem de programação. Os resultados obtidos sugerem que os fracos alunos apresentam claras dificuldades na abstração dos problemas e tendem para uma resolução por tentativa e erro dos problemas. Os bons alunos descrevem uma maior concentração dos seus esforços na análise aprofundada do problema, sua partição e representação mental, avançando para a resolução com maior clareza cognitiva, organizando a solução em passos sequenciais[Ambrósio et al. 2011].

França et. al (2015) discutem que atualmente é requerido dos estudantes desenvolver diversas habilidades, dentre elas o pensamento computacional. Contudo, no Brasil o ensino de tal habilidade não integra o currículo escolar. Nesse sentido, este artigo discute desafios ao ensino do pensamento computacional na educação básica brasileira, apresentando oportunidades de pesquisa na área. Além disso, é apresentada uma proposta para minimizar alguns dos problemas apontados, a qual demonstrou contribuir com a aprendizagem de estudantes do nível médio durante um curso de desenvolvimento de jogos digitais[França and Tedesco 2015].

Prietz e Pazeto (2010) listam vários fatores que podem gerar a evasão no curso de Licenciatura Plena em Informática na Universidade Federal do Mato Grosso. Dentre os motivos estão a distorção por parte dos alunos das habilidades necessárias para o sucesso no curso, embora informações sobre o curso estejam disponíveis no campus. Fatores socioeconômicos também limitam o turno em que os alunos podem dedicar a um curso superior, uma vez que eles precisam trabalhar para auxiliar no sustento das famílias. [Prietch and Pazeto 2010]

O presente estudo tenta definir as principais dificuldades enfrentadas por alunos novatos do curso de Licenciatura em Computação na Universidade Federal Rural de Pernambuco. Embora algumas razões citadas nos artigos acima são observadas no local do estudo, o foco desse trabalho é nos relatos das experiências dos próprios alunos e professores de LC na UFRPE, para que as alternativas sugeridas para o ensino dos conceitos básicos de Computação abordados nas disciplinas introdutórias do primeiro e segundo períodos do curso sejam relevantes.

4. Metodologia

Júnior e Rapkiewicz (2004) pontuam o envolvimento dos alunos e dos professores como grupos distintos e responsáveis pela origem dos problemas no ensino-aprendizagem dos fundamentos de programação[Júnior and Rapkiewicz 2004]. Com isso em mente, para esse estudo foram conduzidas duas pesquisas utilizando a ferramenta Google Forms, onde

professores e alunos de LC na UFRPE compartilharam dados e opiniões sobre o curso.

Os alunos e professores de LC da UFRPE foram escolhidos para participarem da pesquisa devido à facilidade de acesso aos mesmos por meio de interações diárias. Os formulários foram disponibilizados através das mídias sociais, assim como através da lista interna para alunos e professores da área, e ficaram disponíveis por duas semanas durante o primeiro semestre letivo de 2017. Com os resultados obtidos nas duas pesquisas, um estudo quantitativo foi executado para saber o número de alunos que precisaram cursar as disciplinas de Matemática Discreta, Introdução à Programação e Algoritmos e Estruturas de Dados mais de uma vez. Similarmente, um estudo qualitativo foi realizado para descobrir as opiniões dos alunos com relação a suas experiências com os métodos de ensino adotados pelos professores de LC, assim como a opinião dos professores com relação às possíveis razões para o alto número de reprovações e evasão do curso.

5. Resultados e Discussões

Após o fechamento dos questionários, os resultados foram armazenados e foi realizada uma análise das respostas. As respostas abertas foram resumidas levando em consideração a uniformidade de seus relatos.

5.1. Questionário dos Alunos

Os trinta alunos que responderam ao questionário indicaram ter ingressado no curso de LC entre os semestres 2012.1 e 2017.1, sendo os quatro que cursam atualmente o primeiro período redirecionados para algumas seções específicas do questionário, visto que suas experiências no curso ainda não contemplam a disciplina de Algoritmos e Estruturas de Dados. As perguntas iniciais pediram que os alunos classificassem suas experiências com programação e conhecimentos lógico-matemáticos antes de iniciar o curso. Os resultados aparecem nas tabelas abaixo.

Tabela 1. Experiência prévia em Programação.

Experiência em Programação	
Ótima	15.4%
Regular	26.9%
Ruim	19.2%
Péssima	3.8%
Inexistente	34.6%

Tabela 2. Conhecimentos Lógico-Matemáticos prévios.

Conhecimento Lógico-Matemático	
Ótimo	19.2%
Regular	65.4%
Ruim	7.7%
Péssimo	7.7%

Esses resultados indicam que os próprios alunos têm consciência de suas defasagens ao ingressarem no curso de LC. Embora terem sido aprovados para a entrada no

curso, os alunos não necessariamente contam com o embasamento apropriado para acompanharem a rápida progressão dos conteúdos.

Com relação às disciplinas do primeiro período do curso, 46.2% dos alunos cursaram Introdução à Programação uma vez, no entanto, 42.2% precisaram cursar duas vezes. Já em Matemática Discreta, 30.8% dos alunos cursaram uma vez, enquanto 50% precisaram cursar duas vezes. Na primeira disciplina citada, o assunto mais compreendido pelos alunos foi Arquivos, e na segunda, Grafos. Na opinião da maioria dos alunos, o ponto forte dos professores do curso de LC é o domínio do assunto, no entanto o ponto fraco concentra-se na dificuldade de passar o assunto aos alunos, principalmente devido ao fato deles não terem uma formação fundada no domínio didático. Vários alunos mencionaram a dificuldade em acompanhar os assuntos devido ao pouco tempo existente para a obtenção dos conhecimentos.

Das respostas à pergunta sobre sugestões de mudanças ao curso, catorze mencionam a grade curricular assim como detalhes da organização das disciplinas. Alguns alunos reiteram a falta de didática dos professores, mencionando especificamente as disciplinas de Introdução à Programação e Matemática Discreta.

Por fim, 73.1% dos alunos cursando a partir do segundo semestre afirmam já terem considerado desistir do curso, enquanto 50% dos alunos do primeiro semestre também já consideraram essa opção. Esses dados reforçam a correlação entre o alto número de reprovações nas disciplinas introdutórias e a desmotivação dos alunos de LC.

5.2. Questionário dos Professores

Dos nove professores que responderam ao questionário, cinco possuem formação acadêmica em Ciência da Computação, um em Sistemas de Informação, e os outros não informaram. Com relação à percepção dos conhecimentos lógico-matemáticos dos alunos, seis professores os consideram fracos ou médios, enquanto dois não consideram ter dados suficientes para opinar. No entanto, sete professores consideram os conhecimentos de programação dos alunos como fracos.

Seis professores indicaram utilizar métodos de ensino tradicionais, como aulas expositivas, e apenas dois mencionaram métodos mais contemporâneos, como o Project-Based Learning (PBL) [Thomas 2000] e o uso de jogos. Todos os professores que responderam ao questionário listam a prova como o método de avaliação utilizado, ainda que alguns também listam outras formas de verificar o aprendizado, como seminários, apresentações e listas de exercício.

Os professores listam como razões para a alta taxa de evasão no curso o turno e dificuldades por parte dos alunos em se dedicarem aos estudos. Apenas dois professores mencionam as dificuldades e reprovações nas disciplinas iniciais como possível razão. Um professor destaca a falta de acompanhamento pedagógico, e um outro atribui a evasão ao fraco embasamento dos alunos.

Cinco professores sugerem mudanças na matriz curricular e um professor afirma que o corpo docente precisa estar mais bem alinhado à proposta do curso. Um professor também destaca que a matriz curricular de LC está sendo reformulada, a fim de atender as demandas do mercado e dos alunos.

Levando em consideração que uma das habilidades que compõem o perfil do gra-

duado em LC é a contribuição à criação de inovações no ensino e aprendizagem, nota-se que essa prática não vem sendo modelada aos alunos. A rejeição por parte dos professores em acolherem métodos de ensino não tradicionais pode se dar em decorrência de a maioria deles possuírem graduação e pós-graduação em Ciência da Computação em vez de Licenciatura, onde a obrigatoriedade das disciplinas de educação expõem e enfatizam o uso de metodologias de ensino variadas. Entende-se que os professores não têm como modificar a estrutura das disciplinas que lecionam, mas as formas de ensino podem ser reavaliadas visando o melhor aprendizado por parte dos alunos.

5.3. Soluções Propostas

Embora a matriz curricular de LC atualmente esteja em processo de revisão, é importante observar que mudanças estruturais isoladas podem não alterar a alta taxa de evasão do curso. É preciso reavaliar a eficácia de se oferecer disciplinas de Prática de Ensino ao mesmo tempo que o assunto específico está sendo construído por parte dos alunos, pois essa estrutura aumenta as chances dos alunos não possuírem a base necessária para fazer escolhas conscientes sobre os métodos de ensino mais apropriados para um tópico qualquer.

Caberia à coordenação do curso considerar a remoção de disciplinas de Prática de Ensino para abrir espaço na matriz curricular para que disciplinas como Introdução à Computação e Matemática Discreta pudessem ser divididas, visto que suas ementas contêm um volume extenso de tópicos e habilidades desejadas ao fim do semestre. Ao reduzir o volume de novos conhecimentos impostos aos alunos logo no primeiro semestre, talvez o aprendizado e fixação das ideias ocorreria de forma mais orgânica, motivando os alunos a permanecerem no curso em vez de abandoná-lo. É importante destacar que a própria introdução dos alunos a um curso superior causa outros impactos não discutidos no presente estudo, mas existentes e relevantes mesmo assim.

Uma maior conscientização por parte dos alunos deve existir que a construção do aprendizado em LC depende do embasamento lógico-matemático, assim como habilidades de solução de problemas que estão presentes ao longo do curso inteiro. Similarmente, os professores poderiam variar os métodos de ensino, uma vez que o volume de assuntos presentes nas ementas introdutórias podem aumentar a ansiedade dos alunos ingressantes, e assim interferir no seu aprendizado. Embora existam monitorias para o acompanhamento dos alunos menos experientes, esse recurso raramente é utilizado, visto que a maioria dos alunos de LC trabalha durante o dia e não têm tempo de frequentar as sessões.

6. Conclusão

A alta taxa de evasão no curso de LC na UFRPE pode ser explicada por vários motivos, como o fraco embasamento lógico-matemático e de programação dos alunos ingressantes, a organização da matriz curricular, e os métodos de ensino tradicionais. Tanto os professores como os alunos do curso percebem essas dificuldades, e uma grande parcela do alunado precisa repetir disciplinas introdutórias do curso, o que pode levar alguns deles a desistirem do curso. Embora a reestruturação da matriz curricular esteja sendo construída atualmente, uma reflexão mais profunda é necessária para assegurar a eficácia do ensino e aprendizado dos alunos.

O presente estudo identificou algumas percepções e observações do curso de LC na UFRPE por parte dos alunos e professores, e algumas sugestões de soluções foram

apresentadas. Esse estudo indaga sobre o embasamento matemático-lógico dos alunos ingressantes no curso, assim como as metodologias de ensino empregadas nas disciplinas de Introdução à Computação, Matemática Discreta, e Algoritmos e Estruturas de Dados, e sugere uma possível reestruturação dos conteúdos, para diminuir a taxa de reprovação assim como a alta ocorrência da evasão do curso. Trabalhos futuros poderiam testar algumas dessas sugestões, ou até investigar os aspectos sociopedagógicos que podem influenciar na decisão de abandonar o curso. Outros trabalhos poderiam estender a pesquisa a cursos de LC em outras instituições ou até em outros tipos de Licenciatura. O bom preparo dos profissionais da educação é imprescindível para o futuro da humanidade, visto que professores são responsáveis pela formação de profissionais de todas as outras áreas. E visto que a Computação é uma área em constante transformação, cabe também aos formadores de futuros profissionais refletirem sobre o impacto de sua práxis na realidade e construção do conhecimento de seus alunos.

Referências

Ambrósio, A. P. et al. (2011). Programação de computadores: compreender as dificuldades de aprendizagem dos alunos.

dos Santos, R. P. and Costa, H. A. X. (2006). Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *INFOCOMP*, 5(1):41–50.

França, R. and Tedesco, P. (2015). Desafios e oportunidades ao ensino do pensamento computacional na educação básica no brasil. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1464.

Gomes, A., Henriques, J., and Mendes, A. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias-ISSN 1646-933X*, 1(1):93–103.

Júnior, J. and Rapkiewicz, C. E. (2004). O processo de ensino-aprendizagem de fundamentos de programação: uma visão crítica da pesquisa no brasil. In *Anais do XII Workshop sobre Educação em Computação (SBC)*.

Prietch, S. S. and Pazeto, T. A. (2009). Análise, sugestões e perspectivas de um curso de licenciatura em informática. *XVII WEI/CSBC, Bento Gonçalves/RS*.

Prietch, S. S. and Pazeto, T. A. (2010). Estudo sobre a evasão em um curso de licenciatura em informática e considerações para melhorias. *WEIBASE, Maceió/AL*.

SILVA, I., Silva, I. M. M., and Santos, M. S. (2009). Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação. *Universidade Federal Rural de Pernambuco, Recife–PE*.

Thomas, J. W. (2000). A review of research on project-based learning.

CriptoLab: Um game baseado em Computação Desplugada e Criptografia

Débora Juliane Guerra Marques da Silva[1], Graziela Ferreira Guarda[1], Ione Ferrarini Goulart[2]

[1]Departamento de Computação – Universidade Católica de Brasília (UCB)
Campus I - QS 07 – Lote 01 – EPCT – Águas Claras – Brasília – DF CEP: 71966-700

[2]Área de Informação e Comunicação – Instituto Federal de Brasília (IFB)
Campus Brasília - SGAN 610 Módulo D, E, F e G - CEP: 70830-450.

deborajuliane@gmail.com, grazielafguarda@gmail.com, ionefg@gmail.com

Abstract. *The research project called Logicamente was created with the aim of teaching contents of computation focused on the guidelines of computational thinking for children and adolescents of basic education. Among the planned activities, it was idealized the realization of play workshops with the purpose of fixing contents work during the meetings. In this context, came the game CriptoLab that explores encryption in a disrupted computing environment. The present article consists in reporting the experiences about the application of the game in question that has as objective the crossing of a labyrinth whose course will be realized by means of assembly of logical sequences based on the commands of the application of MIT - Scratch.*

Resumo. *O projeto de pesquisa chamado Logicamente foi criado com o objetivo de ensinar conteúdos de computação com enfoque nas diretrizes do pensamento computacional para crianças e adolescentes da educação básica. Dentre as atividades previstas, se idealizou a realização de oficinas lúdicas com o propósito de fixar conteúdos trabalhos durante os encontros. Neste contexto, surgiu o jogo CriptoLab que explora a criptografia em um ambiente de computação desplugada. O presente artigo consiste em relatar as experiências acerca da aplicação do jogo em questão que tem por objetivo a travessia de um labirinto cujo percurso será realizado através de montagem de sequências lógicas baseadas nos comandos do aplicativo do MIT - Scratch.*

1. Introdução

Os jogos, são uma atividade rica e de grande efeito que respondem às necessidades lúdicas, intelectuais e afetivas. Estimula a vida social e representa, uma importante contribuição na aprendizagem. Através destes, crianças desenvolvem capacidades, conhecimentos, atitudes e habilidades, entre elas, se destacam: o favorecimento da mobilidade, a imaginação, a diversão, a aceitação de regras, o desenvolvimento do raciocínio lógico, entre outros.

Utilizar novos recursos didáticos no contexto educacional é primordial especialmente em um momento em que o uso das tecnologias da informação e conhecimento (TIC´s) se faz tão presente no cotidiano desse público infantil (Unesco, 2015). Neste sentido, pode ser destacado o lúdico, como uma maneira de contribuir para

motivar os estudantes a buscar, pesquisar, gerar novos conhecimentos, trabalhar de forma cooperativa como uma estratégia para manter o educando na escola, não por obrigação, mas por motivação.

Em paralelo, a criptografia pode ser compreendida como um conjunto de métodos e técnicas para cifrar ou codificar informações legíveis por meio de um algoritmo, convertendo um texto original em um texto ilegível, sendo possível mediante o processo inverso a recuperação das informações originais. (Simon, 1999). Incorporar a criptografia aos jogos lúdicos é algo interessante pois irá despertar nos estudantes o interesse por conteúdos de computação que são fundamentais nos dias atuais.

O presente artigo visa contribuir para a construção de um processo de ensino-aprendizagem gamificado, no qual o desenvolvimento do raciocínio lógico e computacional são estimulados através de práticas que abordam a criptografia com uso de sequências lógicas de forma estruturada amparada pela computação desplugada.

O jogo foi uma das atividades realizadas pelo projeto de pesquisa chamado Logicamente que visa contribuir para a construção de um processo de ensino-aprendizagem gamificado, no qual o desenvolvimento do raciocínio lógico e computacional são estimulados com o intuito de contribuir para a melhoria do rendimento escolar dos estudantes no contexto das ciências exatas. As atividades do projeto se baseiam na realização de oficinas em laboratório de informática apoiado pelo uso dos jogos digitais educativos, bem como, por atividades lúdicas – foco do presente trabalho – com o propósito de estimular o desenvolvimento do raciocínio lógico sob a ótica do Pensamento Computacional (PC).

O artigo está dividido da seguinte maneira: a seguir, na Seção 2, é apresentada uma explicação sobre o jogo e a metodologia utilizada, bem como, sua organização estrutural. Os resultados parciais são descritos na Seção 3. Por fim, os objetivos e metas desta experiência serão destacados na Seção 4, de forma a concluir o propósito do jogo diante dos resultados já obtidos, bem como, relatar a perspectiva de resultados futuros e melhorias que poderão ser integradas posteriormente.

2. Proposta e Metodologia

A ideia do jogo surgiu oriunda da necessidade de dar continuidade, de maneira prática, a assuntos que foram temas trabalhados pelo projeto Logicamente. Neste aspecto, o presente jogo tem por objetivo aplicar os conceitos de criptografia e lógica de programação relacionado as habilidades do PC como: Abstração – capacidade de filtrar informações essenciais e descartar as informações desnecessárias em um determinado contexto; Decomposição – dividir um problema grande em partes menores, facilitando sua solução; Coleta de Dados – localizar dados necessários para resolver um problema; e Construção de Algoritmo – sequências de passos ordenados para se atingir um determinado objetivo (Pessoa *et al*, 2017).

O jogo foi estruturado em cinco partes. As quatro primeiras abrangeram as mesmas orientações: sortear e responder uma questão de raciocínio lógico, decodificar uma mensagem, procurar uma ficha e montar partes de um código fonte – programa desplugado – baseado na linguagem do aplicativo *Scratch*. Esse processo foi repetido quatro vezes, pois o código foi propositalmente dividido a fim de exercitar mais as capacidades desenvolvidas pelo PC de decomposição.

Após conclusão dessas quatro primeiras partes, era possível simular a passagem pelo labirinto impresso numa folha de papel A4, que se refere a quinta parte do jogo conforme demonstrado na Tabela 1 a seguir.

Tabela 1. Esquema das Fases do Jogo.

	Parte 1:	**Parte 2:**	**Parte 3:**	**Parte 4:**	**Parte 5:**
Etapa 1	Responder questão	Responder questão	Responder questão	Responder questão	Atravessar o labirinto
Etapa 2	Decodificar mensagem	Decodificar mensagem	Decodificar mensagem	Decodificar mensagem	-
Etapa 3	Buscar a ficha e trocar pelo envelope	Buscar a ficha e trocar pelo envelope	Buscar a ficha e trocar pelo envelope	Buscar a ficha e trocar pelo envelope	-
Etapa 4	Montar trecho do código	Montar trecho do código	Montar trecho do código	Montar trecho do código	-

O lúdico foi realizado em uma sala de aula – Figura 1. O público-alvo foram estudantes dos 5° e 6° ano do ensino fundamental de uma escola particular do Distrito Federal (DF). Preliminarmente, dezesseis fichas foram entregues para quatro funcionários da escola que se encontravam em localizações bem conhecidas pelos estudantes, sendo 4 fichas para cada funcionário referente a cada uma das 4 etapas (Tabela 1).

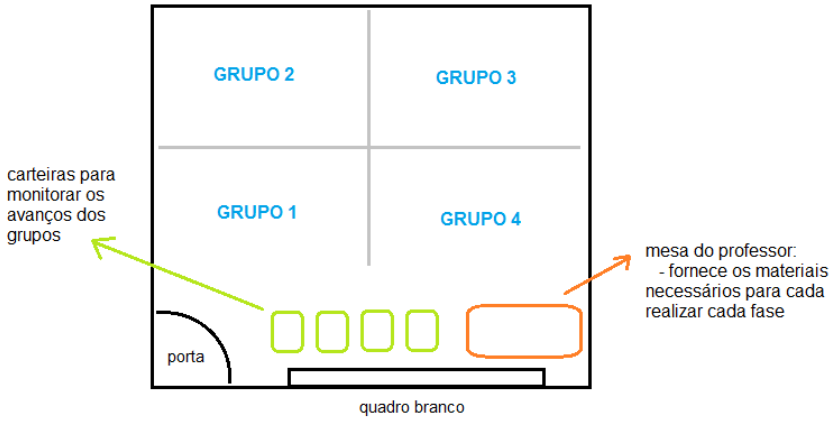


Figura 1. Organização do Ambiente Utilizado.

A turma foi dividida em quatro grupos compostos por quatro a cinco estudantes cada. Cada grupo se reuniu em um canto da sala conforme o exposto na Figura 1 e, antes do jogo começar, cada um deles escolheu um representante para sortear as questões e outro para exercer a função de “corredor” - o corredor tinha a responsabilidade de correr atrás dos funcionários da escola que estavam com as fichas para recolhê-las.

Posteriormente, as fichas seriam trocadas pelos envelopes que continha trechos de código baseado no aplicativo *Scratch*, que seriam utilizadas na etapa seguinte conforme demonstrado na Figura 2 - a esquerda acima os envelopes, a esquerda abaixo parte de trechos de código e a direita, o um exemplo de esquema lógico montado por uma das equipes.

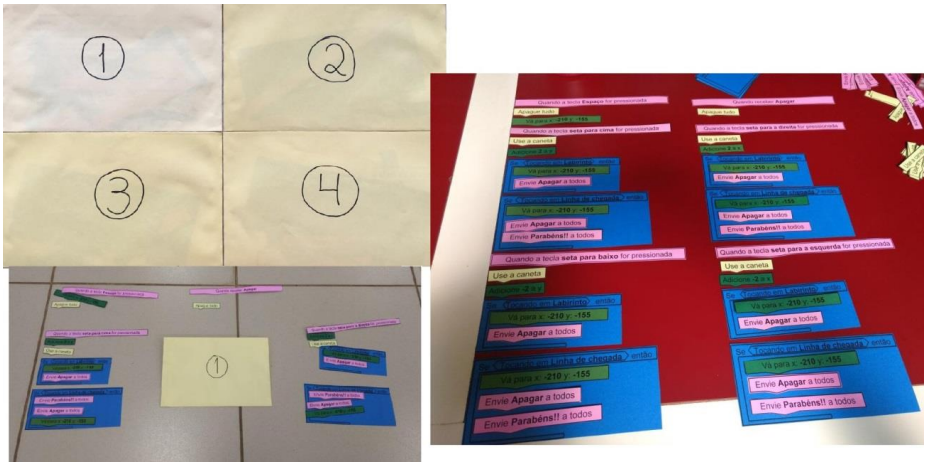


Figura 2. Etapas do Jogo.

Os responsáveis pela aplicação e monitoramento do jogo ficaram reunidos na mesa do professor (Figura 1), onde ocorreram os sorteios das questões, as correções das mesmas e as decodificações das mensagens, assim como, ficaram responsáveis pelo recebimento das fichas que dava acesso os envelopes com os trechos de código necessários para que a travessia pelo labirinto fosse possível. No total, foi necessário que cada grupo recolhesse quatro envelopes. Em paralelo, para monitorar o avanço de cada equipe, foi disponibilizada uma carteira para cada grupo, onde as atividades concluídas foram depositadas.

A primeira etapa da primeira parte do jogo se referiu ao sorteio de uma questão que deveria ser respondida pelas equipes. As perguntas foram de múltipla escolha e tinham como assuntos o raciocínio lógico e matemático alinhado com os conhecimentos acadêmicos dos estudantes de 5º e 6º anos do ensino fundamental e demais temas trabalhados pelo projeto Logicamente. As questões foram retiradas e adaptadas de sítios de concursos públicos e desafios. Assim que finalizada as respostas, as mesmas eram corrigidas e, estando corretas, a próxima fase era desbloqueada. A tabela 2 mostra o conjunto de questões que foram respondidas pelos estudantes na fase 1.

Tabela 2. Questões Fase 1.

N°:	Enunciado:	Alternativas:	Gabarito:
1	Considere a tabela de preços, **por quilo**, do empório do Seu Joaquim, para resolver a questão: ARROZ R$ 1,90 FUBÁ R$ 1,20 FEIJÃO R$ 3,40 FARINHA R$ 2,10 Um freguês comprou 5 Kg de farinha, 3 Kg de fubá e 1 Kg de feijão e ainda sobrou dinheiro para comprar 10 Kg de arroz. O total de dinheiro desse freguês é:	a) R$ 17,50 b) R$ 29,50 c) R$ 36,50 d) R$ 35,50	Letra C
2	Em um jogo de futebol, o 1º gol da partida foi marcado aos 5 minutos do 1º tempo e o 2º gol foi marcado faltando exatamente 7 minutos para o término do 2º tempo. Sabendo que cada tempo durou exatamente 45 minutos e que o intervalo durou exatamente 15 minutos, então, entre o 1º e o 2º gol, passou-se um tempo total de:	a) 1h e 25m b) 1h e 30m c) 1h e 33m d) 1h e 35m e) 1h e 38m	Letra C

3	Sabe-se que Ana é a irmã mais nova e que possui mais seis irmãos. Considere que todos nasceram em anos pares e com uma diferença de dois anos entre cada um deles. Se Ana nasceu em 2002, quantos anos o seu irmão mais velho completará em 2015?	a) 22 anos b) 25 anos c) 24 anos d) 23 anos	Letra B
4	Alice nasceu no dia 1º de março de 1980. Em que ano ela completará 67 anos?	a) 1997 b) 2007 c) 2017 d) 2047	Letra D
5	Dirigindo ao caixa de uma papelaria, o comprador disse: *"Gostaria de comprar 2 pastas com elástico a R$ 1, 00 cada, 3 canetas coloridas a R$ 3,00 cada e 4 borrachas, mas o preço das borrachas eu não sei"*. O caixa entregou uma conta ao comprador de R$ 19,00. De acordo com texto, o valor de cada borracha, em reais, é:	a) 2,20 b) 1,50 c) 1,70 d) 3,00 e) 2,00	Letra E
6	Em um determinado país, as temperaturas registradas em graus Celsius, em certo período do dia, foram as seguintes: -4º, -1º, 0º, -2º, -3º e -5º. A temperatura, em graus Celsius, mais alta registrada nesse país foi de:	a) 0º b) - 5º c) - 1º d) - 4º e) - 3º	Letra A
7	Joãozinho decidiu fazer uma viagem para poder pescar no rio durante cinco dias. No primeiro dia, ele pescou 20. Já no segundo, pescou 48 peixes. No terceiro, conseguiu 65. No quarto dia, foram 54. Finalmente, no quinto dia ele conseguiu pescar 38 peixes. Foram pescadas quantidades que representam números divisíveis por 3 nos dias: **Obs.:** números divisíveis por 3 são aqueles que, ao serem divididos por 3, o resto da divisão é 0.	a) 1 e 2 b) 2 e 4 c) 3 e 5 d) 4 e 5 e) 1 e 3	Letra B
8	Rafael e Orlando combinaram de jogar sinuca valendo R$ 2,00 ao vencedor de cada partida. Rafael chegou para o jogo com R$ 60,00 e Orlando, com R$ 28,00. Ao final do jogo, ambos ficaram com quantias iguais. Nesse caso hipotético, é correto afirmar que a quantidade de partidas que Orlando ganhou a mais que Rafael foi igual a:	a) 4 b) 5 c) 6 d) 7 e) 8	Letra E
9	Gilberto precisa embalar seis dezenas de ovos em caixas com capacidade para uma dúzia ovos cada. A quantidade de caixas necessárias para que Gilberto realize essa tarefa é igual a:	a) 3 b) 5 c) 4 d) 6	Letra B
10	O dobro do triplo da metade de 10 é:	a) 2,5 b) 120 c) 90 d) 100 e) 30	Letra E
11	Assinale a alternativa em que os números estão dispostos do menor para o maior.	a) $\frac{1}{4}$; $\frac{3}{10}$; $\frac{13}{10}$ b) $\frac{3}{10}$; $\frac{1}{4}$; $\frac{13}{10}$ c) $\frac{13}{10}$; $\frac{1}{4}$; $\frac{3}{10}$ d) $\frac{13}{10}$; $\frac{3}{10}$; $\frac{1}{4}$ e) $\frac{3}{10}$; $\frac{13}{10}$; $\frac{1}{4}$	Letra A
12	Abel tem 1,80 metros de altura, Bia tem 1,58 metros, Carlos tem 1,75 metros, Duda tem 1,65 metros e Edu tem 1,98 metros. O mais	a) Abel b) Duda	Letra C

	baixo entre eles é:	c) Bia d) Edu e) Carlos	
13	Cada um dos tempos de um jogo de futebol tem 45 minutos. Se a partida começou às 15 horas e 25 minutos, o primeiro tempo terminará às:	a) 16h10m b) 16h c) 16h25m d) 16h45m e) 16 h15m	Letra A
14	Um escritório de contabilidade imprime cerca de 15 páginas por dia útil. Considerando que o ano possui 250 dias úteis, pode-se afirmar que o total de páginas impressas será:	a) 2.000 b) 2.500 c) 2.750 d) 3.500 e) 3.750	Letra E
15	Ana pretende viajar nas férias para São Paulo. Para conseguir viajar, ela precisa juntar R$ 1.200,00. O salário de Ana é R$ 1.000,00 e ela só pode reservar mensalmente para a viagem a quinta parte do seu salário. Em quantos meses Ana conseguirá realizar sua viagem?	a) 3 meses b) 4 meses c) 5 meses d) 6 meses e) 7 meses	Letra D
16	Um supermercado está realizando uma promoção para os consumidores. A cada R$ 100,00 em compras, o cliente ganha um desconto de R$ 5,00. Se um cliente fizer uma compra de R$ 800,00 o desconto total será:	a) R$ 60,00 b) R$ 50,00 c) R$ 40,00 d) R$ 80,00 e) R$ 70,00	Letra C

A segunda etapa do jogo, consiste em decodificar uma mensagem, cujo conteúdo revelava qual funcionário da escola estaria com a ficha responsável por liberar a etapa seguinte conforme demonstrado na Figura 3 - exemplo de mensagem criptografada de uma das equipes:

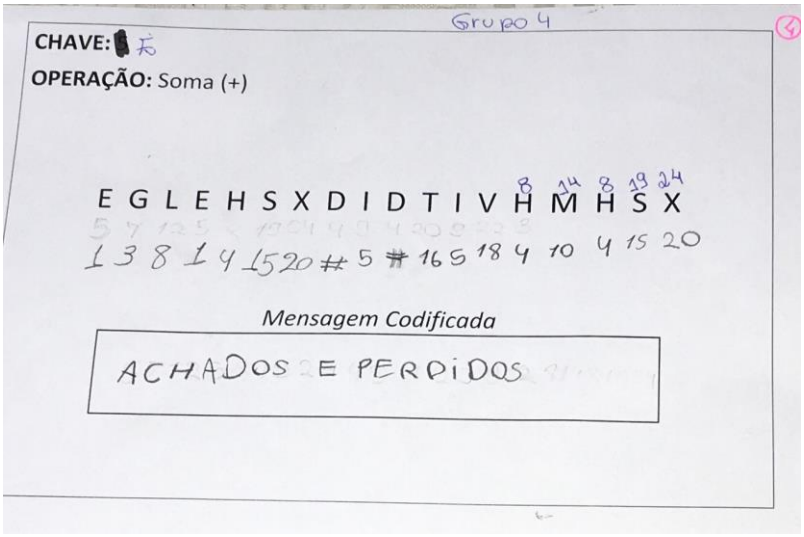


Figura 3. Mensagem Codificada.

Já na terceira etapa, o estudante "corredor" troca a ficha pelo envelope com os trechos de código que tornam possível a passagem pelo labirinto. Posteriormente, a quarta etapa se refere à montagem do código de modo que, o acesso à próxima parte do jogo só será permitido se a construção do mesmo estiver correta. Esse conjunto de etapas se repete quatro vezes, mudando apenas o conteúdo.

Depois que as quatro primeiras etapas foram concluídas, os grupos receberam uma folha com o labirinto e com a tabela que deveria ser escrita a solução do problema – Figura 4, que consistiu em expor quantas vezes e quais seriam as teclas que devem ser acionadas, simulando como se a atividade fosse no computador (cima, baixo, esquerda e direita). Essas instruções foram trabalhadas na aula anterior ao lúdico e reforçadas no dia da aplicação dinâmica, um aspecto a ser destacado é que todos os conteúdos abordados no jogo foram previamente trabalhados em sala, durante as atividades do projeto Logicamente.

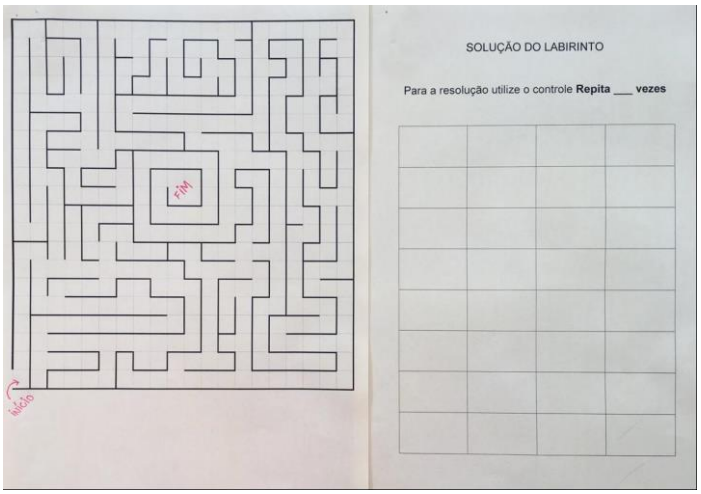


Figura 4. Labirinto e Tabela de Solução.

3. Resultados e Discussões

O CriptoLab foi jogado por um total de quarto equipes, cada equipe continha entre 4 e 5 integrantes. Foi estabelecido como regra geral que cada fase não poderia ultrapassar 20 minutos e esse tempo foi cronometrado por equipe. Todas as equipes conseguiram concluir as 5 fases do jogo. A Tabela 3 abaixo mostra o tempo que cada equipe gastou na finalização de cada fase:

Tabela 3. Tempo Gasto por Equipe.

Fase / Grupo:	**1**	**2**	**3**	**4**
1	19 min	20 min	20 min	18 min
2	18 min	17 min	18 min	17 min
3	15 min	18 min	18 min	13 min
4	17 min	18 min	19 min	13 min
5	6 min	6 min	7 min	5 min
Σ tempo	75 min	79 min	82 min	66 min

De acordo com o exposto na Tabela 3, a equipe vencedora foi a equipe 4, que finalizou todas as etapas do jogo em um total de 66 minutos e a equipe que concluiu por último demorou 82 minutos. Em complemento, a Tabela 4 demonstra o conjunto de características de cada equipe formada.

Tabela 4. Características das Equipes.

Características / Grupo:	**Sexo:**	**Deveres de casa:**	**Concentração:**	**Organização / Divisão das Tarefas:**
1	Masculino	Entregavam parcialmente	Alta	Boa

2	Misto	Não entregavam as atividades	Alta	Ótima
3	Masculino	Não entregavam as atividades	Média	Razoável
4	Feminino	Entregavam com frequência as atividades propostas	Alta	Ótima

Considerando os dados da Tabela 4, observa-se que a equipe campeã foi composta exclusivamente por estudantes do sexo feminino, cuja entregas dos deveres de casa em relação as atividades como um todo do projeto Logicamente eram frequentes - o que justifica uma concentração mais alta por parte desta equipe, uma vez que, os conhecimentos prévios já estavam mais evoluídos. Consequentemente, se observou uma melhor organização e divisão das tarefas durante a execução do jogo.

Durante a execução do jogo, foi possível perceber que os estudantes apresentaram diferentes tipos de dificuldade, algumas em relação ao raciocínio lógico matemático, outros em relação a montagem das sequências do código fonte, e também, em relação a divisão das tarefas.

Aqueles que tiveram dificuldade com a parte matemática do jogo foram aqueles estudantes com perfil mais dispersos em sala de aula e os que faltaram à aula de criptografia. Os estudantes que tiveram mais dificuldade com a montagem do código fonte foram aqueles que não costumavam entregar os deveres de casa, os deveres neste caso, se referiram as atividades usando o aplicativo *Scratch.*

Ainda em relação às dificuldades observadas, destacam-se: dispersão, dificuldade de compreensão dos comandos do *Scratch* – essa dificuldade é devida a falta de prática de exercícios na ferramenta, que se configura em dificuldades conceituais dos conteúdos ministrados. Nos casos em que a mensagem criptografada ultrapassava dez caracteres, se percebeu ansiedade na solução e alguns grupos tentaram burlar os cálculos usando adivinhação de padrões.

Dentre os aspectos positivos, se pode destacar que todas as equipes conseguiram concluir as 5 fases do jogo com sucesso dentro do tempo estabelecido. A divisão do problema em partes menores facilitou o trabalho das equipes. Em relação ao aprendizado dos conteúdos necessários, se observou que os estudantes que se dedicaram a aprender sobre o tema anteriormente a dinâmica, passaram pelas fases com relativa facilidade.

A inclusão da criptografia como parte das atividades foi outro aspecto muito positivo, pois instigou a curiosidade dos estudantes e o despertar para estudos acerca do tema. Além disso, foi observado que, a divisão de um grande problema em partes menores, colaborou positivamente para que todas as equipes concluíssem o jogo. Essa condição de dividir um problema em partes menores vai em consonância com o documento – Referenciais de formação em computação: Educação Básica (SBC, 2017).

Por fim, foi aplicado um instrumento de avaliação, cujo objetivo foi mapear as opiniões dos estudantes que participaram da atividade para fins de ajustes e correções futuras tanto do jogo quanto em relação as atividades do projeto como um todo conforme o exposto na Tabela 5.

Tabela 5. Avaliação do CriptoLab e Atividades do Projeto.

Q:	Enunciado:	Resposta 1:	Resposta 2:	Resposta 3:
1	Como foi trabalhar em equipe no lúdico?	Difícil (3)	Fácil (17)	Indiferente (0)
2	As aulas do projeto ajudaram no lúdico?	Sim, bastante (20)	Não (0)	Não, o assunto dado em sala não foi o suficiente para ajudar no lúdico (0)
3	Em que você teve mais dificuldade?	Matemática (10)	Na utilização da lógica para resolver os desafios (8)	Matemática e lógica (2)
4	Qual foi a sua maior dificuldade para montar o labirinto?	Achar a sequência certa para montar o algoritmo (10)	Identificar os tipos de comando (8)	Tempo insuficiente (2)
5	Ao participar do projeto o seu rendimento escolar aumentou?	Sim, consigo resolver problemas com mais rapidez (16)	Não, o rendimento continua o mesmo (4)	Não, o rendimento caiu (0)
6	Depois de entrar no projeto, você tem conseguido aprender com mais facilidade as matérias dadas na escola?	Sim (18)	Não (1)	Não, continuo do mesmo jeito em que entrei no projeto (1)
7	Você consegue aplicar o conteúdo dado no projeto em seu dia a dia?	Sim (15)	Não (4)	Tenho dificuldades para aplicar o conteúdo no dia a dia (1)

Os números entre parênteses nas colunas de respostas representam as quantidades de votos que cada alternativa recebeu. De acordo com os dados da Tabela 5, os estudantes não apresentaram dificuldades para trabalhar em equipe, as aulas e conteúdos estudados preliminarmente contribuíram de forma efetiva para o bom andamento da atividade proposta, foi identificado que existem dificuldades tanto da matemática quanto da lógica na mesma proporção, do mesmo modo, que houve um equilíbrio entre a dificuldade da organização das sequências lógicas e em relação ao uso dos comandos do *Scratch*.

Em relação as impressões do impacto das atividades do projeto, se percebe, de acordo com a visão dos estudantes, que o rendimento escolar aumentou, que a facilidade de compreensão de conteúdos escolares também foi aumentada e que eles conseguem fazer a relação dos conteúdos aprendidos com a vida real.

4. Conclusões

O projeto Logicamente tem como finalidade motivar crianças e adolescentes a aprender programação, lógica e assuntos relacionados ao pensamento computacional de maneira criativa, que são habilidades essenciais para a vida de qualquer indivíduo. Se estima com a inserção o PC no âmbito da educação básica, oportunizar a formação de habilidades e competências computacionais, apoiando a ciência e suas áreas de conhecimento. Essas habilidades e competências potencializam a capacidade de resolver problemas.

Ações nesse sentido vão em consonância com o que já vem sendo praticado em diversos países, como por Alemanha, Argentina, Austrália, Coreia do Sul, Escócia, França, Inglaterra, Estados Unidos da América, Finlândia, Grécia, Índia, Israel, Japão, e Nova Zelândia, entre outros, que adotaram o ensino de computação nas escolas para desenvolver habilidades relacionadas à resolução de problemas complexos. (SBC, 2017).

O jogo teve por objetivo aplicar os conceitos de criptografia relacionado as habilidades do PC. As crianças, hoje, já nascem imersas em um mundo digital, mas, ao contrário do que se possa imaginar, elas não conhecem o funcionamento desse mundo, apenas utilizam suas ferramentas passivamente. A opção pela inclusão do tema criptografia foi de grande valia para que pudesse ser abordado com o público-alvo posteriormente, assuntos como a segurança eletrônica, que envolve questões importantes sobre os riscos que somos expostos ao utilizar as tecnologias, os quais se pode destacar: casos de violação de contas bancárias, acesso a informações sigilosas, invasão e destruição de sistemas, entre outros.

De acordo com os resultados expostos no capítulo 3 do presente artigo, o jogo foi considerado uma iniciativa interessante, que despertou curiosidade pelos temas abordados, interesse em pesquisas sobre o tema e favoreceu a compreensão dos assuntos relacionados reforçando os conteúdos trabalhados pelo projeto Logicamente.

5. Referências Bibliográficas

UNESCO (2015) “TIC na educação do Brasil”, http://www.unesco.org/new/pt/brasilia/communication-and-information/access-to-knowledge/ict-in-education/, Novembro.

Pessoa, F. I. R; Araújo, A. S. O.; Andrade W. L.; Guerrero, D. D. S (2017) "T-mind: um Aplicativo Gamificado para Estímulo ao Desenvolvimento de Habilidades do Pensamento Computacional". In Anais do SBIE. DOI: 10.5753/cbie.sbie.2017.645.

Sociedade Brasileira de Computação (2017) Referenciais de Formação em Computação: Educação Básica. http://www.sbc.org.br/noticias/10-slideshow-noticias/1996-referenciais-de-formacao-em-computacao-educacao-basica/, Julho.

Simon S. (1999). The Code Book. Fourth Estate, 1st edition.

Cognition Developing of Computer Higher Education Students Through Gamification in the Algorithm Teaching-Learning Process

Tiago do Carmo Nogueira[1], Eudes de Souza Campos[2], Deller James Ferreira[3]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Tocantins – IFTO
Rodovia TO 040 – Km 349 Loteamento Rio Palmeira – Lote 1
Dianópolis - TO

[2]Universidade Estadual de Goiás – UEG
Centro de Ensino e Aprendizagem em Rede – Cear
Avenida Brasil Sul, Nº 2.800 – Jardim Gonçalves
Anápolis – GO

[3]Instituto de Informática – Universidade Federal de Goiás – UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Goiânia – GO

tiago.nogueira@ifto.edu.br

Abstract. *The scientific logical reasoning became an important skill in the students' cognitive development in algorithm teaching-learning processes, stimulating their reasoning and creativity. From this perspective, gamification has been adopted as a mediating tool in this process. Studies report that the inclusion of gamification in algorithm teaching-learning processes stimulates the students to develop new skills, making the knowledge more efficient. Therefore, this paper's purpose is to measure and understand the cognitive development and the experiences lived by students at the addition of gamification in algorithm teaching, evaluating the scientific logical knowledge acquired by them. Consequently, 44 computer higher education students were selected. They were divided into two groups: students that used the Gamification-Mediated Algorithm Teaching Method and those who participated in the traditional teaching method. To evaluate the cognitive development between these two groups, the Scientific Logical Reasoning Test was applied. The results showed that a significant number of students that used the Gamification-Mediated Algorithm Teaching Method reached the transitory intermediary and transitory scientific knowledge levels, with greater right answer rates. We also noticed that both genders gave more right answers using the gamification-mediated algorithm teaching method.*

1. Introduction

One of the great algorithm teaching characteristics is the possibility to develop logical knowledge in the student through skills such as: reasoning, creativity, and patience [Tsai et al. 2016]. From these skills, the scientific logical reasoning plays a significant role in the students' cognitive development in the algorithm teaching-learning processes, especially in disciplines presented in the computer higher education.

In the current formal education, scientific reasoning is a central and resulting purpose in the learning process for scientific education [Piraksa et al. 2014] [Ding et al. 2016]. With the inclusion of gamificated processes in algorithm teaching, through engaging methods, several papers report significant differences in the students' motivational aspects, which may contribute for their mental development [Seng and Yatim 2014] [Lopes et al. 2016].

New processes allow us to assist in these students' learning in algorithm disciplines and make them effective. Therefore, this research's purpose is to measure and understand the cognitive development and the experiences lived by students with the inclusion of gamification in algorithm teaching, directly measuring the scientific logical knowledge acquired by them.

Consequently, this research counted with 44 computer higher education students of the federal teaching network, in the State of Goiás. Twenty-three students participated in the algorithm introductory discipline, using the traditional teaching method, that is, based on theoretical classes and practical exercises. The remaining 21 students were submitted to the Gamification-mediated algorithm teaching method, using the Scratch tool *Scratch*[1].

To evaluate the cognitive capacity between both sample groups, we used, as a measuring tool, Lawson's Classroom Test of Scientific Reasoning (LCTSR) [2] . This test consists of identification questions and variable control, of hypothetical-deductive, proportional, and probabilistic reasoning [Lawson 2004].

The results show that, in the LCTSR, most students reached a concrete scientific knowledge level, about 81%; on the other hand, the transitory intermediary knowledge levels were of 13.95%. We noticed that there were significant differences between the traditional teaching method and the gamification-mediated algorithm teaching method regarding the number of right answers. Consequently, the students that used the gamification-mediated algorithm teaching method reached significantly greater means.

This paper presents a theoretical referential on cognitive development and scientific reasoning (Subsection 2.1), gamification processes for algorithm teaching (Subsection 2.2), methodological procedures (Section 3), results and discussions (Section 4), and final considerations (Section 5).

2. Cognitive Development and Gamification in Algorithm Teaching

This section presents papers related to cognitive development, approaching features of the students' scientific reasoning (Subsection 2.1) and the gamification processes for algorithm teaching (Subsection 2.2).

2.1. Cognitive Development and Scientific Reasoning

Scientific reasoning became the focus of scientific education, affecting the students' academic performance, regarding the teaching-learning processes, and directly affecting their daily decision making [Ding et al. 2016] [Piraksa et al. 2014] [Jensen et al. 2017].

[1]Available at: https://scratch.mit.edu/

[2]Available at: https://www.physport.org/assessments/assessment.cfm?A=CTSR

Studies that correlate the students' learning and scientific reasoning point to a greater success in context-based queries through several gain indicators in teaching-learning processes [Acar 2014]. Therefore, the scientific reasoning capability may be determined as an important factor for the students' performance, learning, and mental development promotion [Piraksa et al. 2014] [Opitz et al. 2017].

However, the cognitive development evaluation proposed by [Piaget et al. 2013], consisting in a set of specific devices through answers, with the purpose of measuring the experiences lived by the subjects, can classify the cognitive development in sensory-motor, pre-operational, concrete and formal operation [Pessoni et al. 2015]. Therefore, formal operations may be understood as a person's ability to understand more abstract concepts of the scientific reasoning.

Examining the gains of the students' scientific reasoning, it is possible to identify and classify them into three types of reasoning: concrete, formal, and post-formal. Therefore, it is possible to apply methods that identify the conceptual knowledge, comparing the performance between student groups before and after the instructions. The results may show that concrete rationalist students have a better performance in comparison with formal and post-formal rationalist students, surpassing them in a conceptual knowledge subscale [Acar and Patton 2016].

Still under this perspective, it is possible, through scientific reasoning classification, to examine the influence of the students' cognitive and motivational factors, verifying if the gender difference is a significant factor for a scientific accomplishment model. In this sense, many researchers report that the gender effects the students' understanding and their attitude towards the teaching-learning [Piraksa et al. 2014].

Studies with this purpose showed that female students surpassed male students on scientific yielding. Additionally, some results show that the most appropriate model that corresponds with this behavior includes the initial conceptual knowledge, scientific reasoning, and science value as variables that predict scientific accomplishment [Acar et al. 2015].

Therefore, through methods that enhance the scientific reasoning's capability, it is possible to identify a strong correlation between creativity technique and scientific reasoning, measuring which skills significantly contribute for the students' creativity increase in teaching-learning processes [Tsai et al. 2016].

2.2. Gamification Processes for Algorithm Teaching

Algorithmic thinking is fundamental for professional development, specifically to solve logical problems that are essential for cognitive development, stimulating the students' scientific reasoning in higher education.

In this sense, gamification becomes an encouraging tool to students in algorithm teaching-learning processes. For [Seng and Yatim 2014], for a while, computer games have been adopted as a part of the teaching-learning tools. Some studies show that traditional teaching methods, aligned with the inclusion of gamification, through games, stimulate students to search for new skills, with more efficient knowledge [Seng and Yatim 2014] [Lopes et al. 2016].

Therefore, pedagogic solutions and the use of serious games in teaching-learning

processes became more inclined to encourage and engage students, efficiently allowing knowledge development [Ouahbi et al. 2015] [Topîrceanu 2017]. These solutions increase the students' interest in activities employed in the process of code revision in knowledge managing, in change managing, and in problem traceability for algorithmic proposals [Khandelwal et al. 2017] [Knutas et al. 2017].

Through the game use and student encouraging evaluation in algorithm learning with the creation of simple games, using the Scratch environment, we notice that the students' learning is encouraged, enabling them to continue their programming/algorithm studies [Ouahbi et al. 2015].

For [Khandelwal et al. 2017], while measuring gamification's impact in the algorithmic code revision processes, we notice an improvement in the codes' quality and in the students' learning. Additionally, if the industrial demand features were considered, gamificated processes may significantly increase productivity in algorithm implementations. However, simulation-mediated methods that use game concepts have significant differences in comparison with the traditional teaching methods. In this sense, students that use algorithm teaching methods, through gamification, present a better performance regarding the learning of basic concepts on algorithms in comparison with students that use the traditional teaching method [Lopes et al. 2016].

Therefore, this paper focus on the application of gamificated methods in algorithm teaching-learning processes, evaluating cognitive aspects under the perspective of higher education students' scientific reasoning knowledge, which will lead to mental and intellectual development.

3. Methodology

This approach evaluated the execution of an applied research, characterized by quantitative and qualitative tools to correlate algorithm traditional teaching methods and gamification-mediated methods. Therefore, in this section, we present the profile of the research's participants (Subsection 3.1), propose a gamification-mediated algorithm teaching method (Subsection 3.2), and describe Lawson's Classroom Test of Scientific Reasoning (Subsection refLawson).

3.1. Profile of the Participants

This paper counted with the participation of 44 computer higher education students of the federal teaching network of the State of Goiás (IFGoiano, IFG, and UFG). All were students coming from initial classes of first semester. However, none of the students had previous contact with programming and computational thinking.The workshops for application of the methods occurred for two weeks.

The students' average age was 22 years old. The group had 11 female students and 33 male students. They were divided into two groups:

- those that used the traditional teaching method (23 students);
- the participants of the gamification-mediated teaching method (21 students).

3.2. Gamification-Mediated Algorithm Teaching Method

Based on the method proposed by [Lopes et al. 2016], the serious game-oriented algorithm teaching method's difference is the use of gamification resources in the teaching

processes. Therefore, Figure 1 presents a sketch of the stages employed between the traditional teaching method and the simulation-mediated teaching method (gamification).

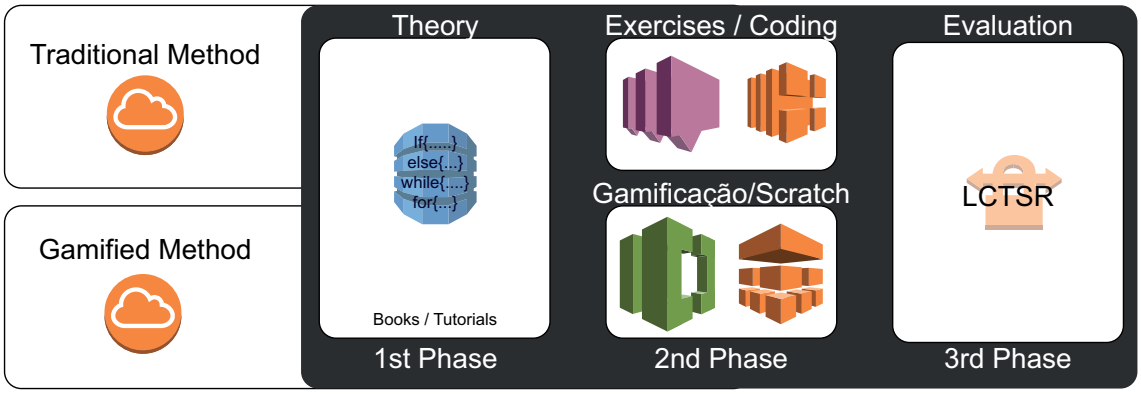


Figure 1. Stages employed in the traditional teaching and the gamification-mediated method

Figure 1 comparatively presents the stages employed between the two teaching methods measured by this research. We notice that the first stage (theory) is based in the inclusion of pedagogic tools that help the students to obtain a theoretical and referential understanding. This stage is common on both teaching methods and may be based on the concepts of the main decision and repetition commands: “if-then”; “else-then”; “if-then-else”; “repeat-until”; “while-do”, and “to-until”.

In the second stage, in the traditional teaching method, the students are encouraged to perform practical exercises on algorithms, that is, to code and implement new functions, based on the concepts studies in stage 1 (theory).

In this stage, in the gamification-mediated teaching method, the students are engaged to performed practical exercises through game building, using the basic concepts studied in stage 1 (theory), presenting them to the game and gamification concepts. In this sense, the Scratch tool was used to pedagogically help in the exercises’ execution in the algorithm teaching-learning process (Figure 2).

This tool uses logical blocks with items of sounds and images through interactive stories, making use of ’gamification’ concepts. The most common concept can summarize gamification as the utilization of game mechanics in other contexts.For [Lopes et al. 2016], an evident feature of this tool is the ability to stimulate cooperative learning, sharing knowledge along the learning process, developing creative skills in the students to solve programming/algorithm problems.

Scratch was developed by the Lifelong Kindergarten group at the MIT Media Lab. The tool was developed in 2003 and especially designed for students between 8 and 16 years of age, but is used by people of all age groups [Resnick et al. 2009]. *Scratch* is used in over 150 countries, it is available in over 40 languages and it is provided free of charge for all major operating systems, Linux, Windows and Mac OS.

In the third stage, method evaluation and comparison, we proposed the application of the LCTSR. In the next section, the LCTSR’s main features are presented.

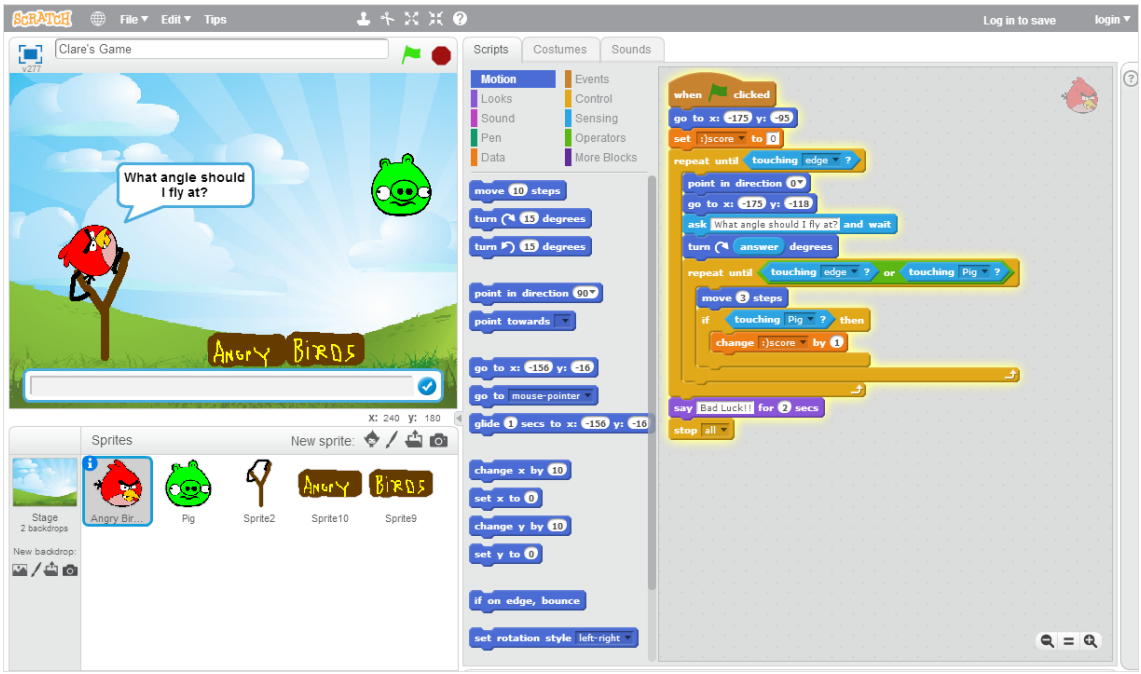


Figure 2. Scratch Simulation Tool

3.3. Lawson's Classroom Test of Scientific Reasoning

The first LCTSR version was proposed in 1978 and updated in 2000 [Piraksa et al. 2014] [Tsai et al. 2016]. The test has 24 questions. Each question was proposed to measure the students' scientific understanding. The questions are grouped in the following measures: mass and volume conservation – questions 1, 2, 3, and 4; proportional reasoning – questions 5, 6, 7, and 8; variable control – 9, 10, 11, 12, 13, and 14; probabilistic reasoning – questions 15, 16, 17 and 18; correlation reasoning – questions 19 and 20; and hypothetical-deductive reason – questions 21, 22, 23, and 34 [Lawson 2000].

The LCTSR was applied in both student groups: the ones that use the traditional teaching method and the participants of the gamification-mediated algorithm teaching method. Therefore, through the test, it was possible to classify the students by scientific reasoning knowledge levels. The adopted classification levels were: concrete (0 to 8 right answers), transitory (9 to 14 right answers), formal (15 to 20 right answers), and post-formal (21 to 24 right answers).

To compare the variables regarding their performance on the LCTSR and of individual characterization between the methods, Fisher' exact test was used for the qualitative variables and the Mann-Whitney test for the quantitative variables [Hollander et al. 2013]. Fisher's exact test verifies if there is a hypothesis association of two variables in a given sample, determining the exact observed frequency. The Mann-Whitney test is a non-parametric test employed in independent statistical samples.

Objectifying the verification of the students' score association with their respective ages and sexes, the Spearman correlation [Hollander et al. 2013] and the Mann Whitney test [Agresti and Kateri 2011] were used.

4. Results and Discussions

This section presents the results and discussion of the sample descriptive analyses (Subsection 4.1), the comparative results between the traditional and the gamification-mediated methods (Subsection 4.3), and the discussions on the students' performance by gender (Subsection 4.4).

4.1. Descriptive Analysis of the Sample

For the data extraction procedure in the qualitative variable descriptive analyses, absolute and relative frequencies were used. In the quantitative variable description, position, central tendency, and dispersion measurements were used. Therefore, Table 1 quantitatively presents the descriptive variable analysis result.

Table 1. Variable descriptive analysis

Variables		N	%
Method	Gamification	21	47.73%
	Traditional	23	52.27%
Level	Concrete (C)	35	81.40%
	Transitory Intermediary (ET)	6	13.95%
	Transitory (LT)	2	4.65%
Sex	Female	11	25.00%
	Male	33	75.00%
Right Answers	Mean(S.D)	22.28 (15.32)	
Score	Mean(S.D)	3.16 (2.05)	

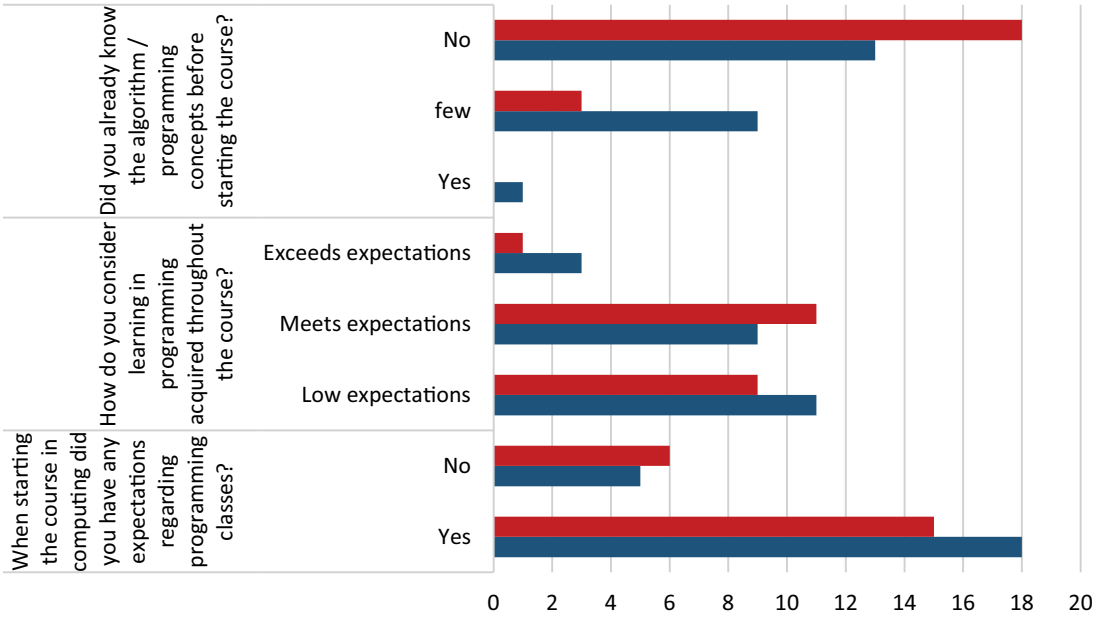
Through Table 1, we can notice that the most frequent method among the sample students was the traditional one—about 51%—in comparison with the gamification-mediated algorithm teaching method. Evaluating the levels identified by the LCTSR, most of the students reached the concrete scientific logical knowledge level, with 81.4%, followed by the transitory intermediary level, with 13.95%.

Under the perspective of the students' right answer numbers in the LCTSR, we noticed that the right answer mean was of 22.28%, with standard deviation (15.32). Regarding the students' performance, we noticed a mean of 3.16, with standard deviation (2.05).

4.2. Analyzing the Experience and Expectations of the Research Participants

Regarding the knowledge of the concepts of algorithm and/or programming before starting the technical course, it was verified that about 39% of the subjects who used the traditional method had little knowledge about algorithm and/or programming (N=9) and approximately 56% reported that there was no knowledge (N=13). In relation to the gamification-mediated algorithm teaching method, about 14% of the subjects reported that they had little knowledge about algorithm and/or programming (N=3) (Figure 3).

It is observed that, through the analysis of the results, about 48% of the subjects who used the traditional method of programming teaching reported that learning by this method was below their expectations (N=11). In this sense, 39% of the subjects reported



	When starting the course in computing did you have any expectations regarding programming classes?		How do you consider learning in programming acquired throughout the course?			Did you already know the algorithm / programming concepts before starting the course?		
	Yes	No	Low expectations	Meets expectations	Exceeds expectations	Yes	few	No
Gamification	15	6	9	11	1	0	3	18
Traditional	18	5	11	9	3	1	9	13

Figure 3. Experience and expectations of research participants

that learning through this method met their expectations (N=9) and approximately 13% reported that the traditional method exceeded their expectations (N=3).

Regarding the learning through the gamification-mediated algorithm teaching method, 43% of the participants reported that it was below their expectations (N=9). Approximately 53% of the participants reported that the method learning met their expectations (N=11) and 5% reported that the method exceeded their expectations (N=1).

Regarding the main difficulties encountered by the participants of this research in the algorithm and/or programming disciplines, approximately 65% of the subjects who used the Traditional Method reported that they had greater difficulties in relation to the teaching/didactic strategies of the process. About 21% of the subjects reported that the difficulties were caused by the lack of attention or effort of the student himself and 14% reported that the content was one of the biggest barriers faced.

In this sense, it can be noticed the inefficiency of traditional methods in comparison to the gamification-mediated algorithm teaching method. Thus, analyzing the method

proposed by this research from the perspective of user satisfaction, approximately 43% of the participants indicated that the gamification-mediated algorithm teaching method obtained a satisfactory result. 50% reported that the results using this method were regular and about 7% reported that the results obtained by this method were unsatisfactory.

Regarding the satisfaction of users in the traditional method, approximately 74% of the participants reported that the results obtained by this method were below their expectations, that is, regular or poor. Thus, about 93% of the subjects using this method indicated the need to use other methods in teaching algorithm and/or programming.

4.3. Comparison between the Traditional and the Gamification-Mediated Methods

To compare the two methods—traditional and gamification-mediated—we considered the right answer, score, and age variables. Consequently, Table 2 presents the comparison of such variables between both teaching methods.

Table 2. Variable comparison between the traditional and gamification-mediated methods

Right Answers							
Method	N	Mean	E.P.	1Q	2Q	3Q	p-value
Gamification	21	28.10	4.06	17.00	25.00	33.00	0.018
Traditional	22	16.73	1.83	8.00	17.00	25.00	
Score							
Method	N	Mean	E.P.	1Q	2Q	3Q	p-value
Gamification	21	4.00	0.51	3.00	4.00	5.00	0.012
Traditional	23	2.39	0.29	1.50	3.00	3.00	
Age							
Method	N	Mean	E.P.	1Q	2Q	3Q	p-value
Gamification	21	22.71	0.12	16.00	17.00	17.00	0.000
Traditional	23	21.13	0.11	15.00	15.00	15.00	

Through Table 2, we can notice that there were significant differences between the traditional and the gamification-mediated teaching method (p-value = 0.018), regarding the students' right answer numbers.

Therefore, the students that used the gamification-mediated algorithm teaching method presented a significantly higher mean in comparison with the students that used the traditional teaching method. Figure 4 presents the results of the correlations between the methods and the right answer, score, age, and gender variables.

We also notice that the gamification-mediated algorithm teaching method got a significantly greater mean score than the traditional teaching method (p-value = 0.012). Regarding the students' age (Figure 4), the subjects that used the gamification-mediated algorithm teaching method got a significantly greater mean age in comparison with the ones that used the traditional teaching method (p-value = 0.000).

Correlating the students' gender, between both methods, under the scientific logical knowledge and sex level, we identified significant differences. Therefore, Table 3 presents the results of the comparisons between the traditional and gamification-mediated teaching method.

We notice that there is a significant difference between both teaching methods (p-value = 0.034). Therefore, we noticed that the concrete scientific logical knowledge

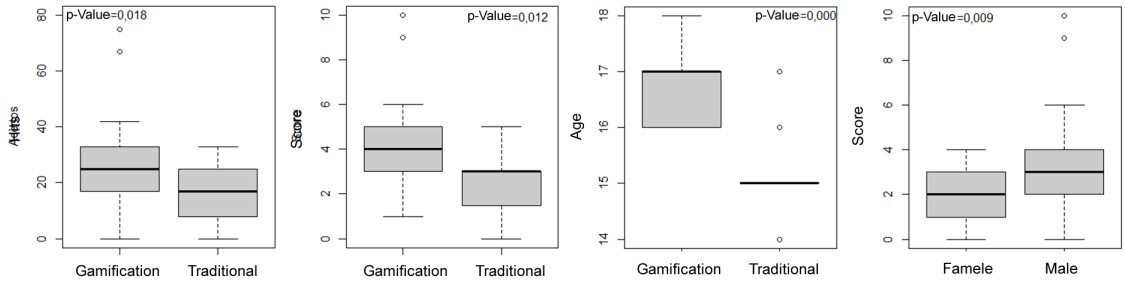


Figure 4. Right answer, score, and age variable comparison between the traditional and the gamification-mediated teaching methods

Table 3. Comparison of the level and sex variables between the traditional and gamification-mediated methods

Variables		Method				p-value
		Gamification		Traditional		
		N	%	N	%	
Level	C	14	66.7%	21	95.5%	0.034
	ET	5	23.8%	1	4.5%	
	LT	2	9.5%	0	0.0%	
Sex	Fem.	4	19%	7	30.4%	0.494
	Male	17	81%	16	69.6%	

level (C) presents a frequency of 95.5% in the traditional teaching method. However, the gamification-mediated algorithm teaching method presented 23.8% of transitory intermediary scientific logical knowledge levels (ET) and 9.5% of transitory scientific logical knowledge levels (LT). Therefore, the gamification-mediated algorithm teaching method was proven the most efficient, stimulating the students' intermediary and transitory reasoning.

4.4. Performance between the Students (Gender)

Through analyses, we identified significant differences between both methods regarding the students' gender (p-value = 0.009). Table 4 presents the performance comparison of the tests between the students' genders.

Table 4. Performance comparison between the students' genders

Variable		N	Mean	E.P.	1Q	2Q	3Q	p-value
Sex	Fem.	11	1.91	0.37	1	2	3	0.009
	Male	33	3.58	0.37	2	3	4	

Through Table 4, we notice that there significant differences between the students' genders regarding the test. Therefore, we notice that the male students have a significantly greater mean than the female students (Figure 4).

Analyzing the occurrence of significant differences regarding the genders between both methods, we notice also that the male students got greater means than the female students in both teaching methods, traditional and gamification-mediated (Table 5).

Table 5. Performance comparison between the students' genders

Variable		Method	Mean	p-value
Sex	Fem.	Gamification	1.77	0.007
		Traditional	1.61	
	Male	Gamification	3.2	0.005
		Traditional	2.8	

However, evaluating just the students' gain in each method, we notice that the gamification-mediated algorithm teaching method, in both genders, got significantly greater means in comparison with the students of the traditional teaching method. Therefore, we noticed that there was a positive and significant correlation (r = 0.34 and p-value = 0.025) between the variables: right answers, score, age, and sex.

Therefore, we notice through the analyses that gamification-mediated methods point to gains in the scientific logical reasoning regardless of the students' gender, stimulating greater intermediary and transitory levels.

5. Conclusions

The paper measured the cognitive development of computer higher education students under the perspective of scientific reasoning in algorithm teaching-learning processes between the traditional and the gamification-mediated teaching method.

Therefore, we used the LCTSR as our measuring tool to evaluate the cognitive capability between both student groups. The test verified through the issues of variable control, and hypothetical-deductive, proportional, and probabilistic reasoning, the indication of the scientific knowledge levels acquired by the students during the algorithm teaching-learning processes.

Through the analyses, we noticed the occurrence of significant differences between both methods. Therefore, we noticed that the students that used the gamification-mediated algorithm teaching method presented greater scientific logical knowledge levels in comparison with those that used the traditional teaching method.

Therefore, through the gamification-mediated algorithm teaching method, a significant number of students reached the transitory intermediary (ET) and transitory (LT) scientific knowledge levels, with greater rates of right answers in the LCTSR. We also noticed that both genders gave more right answers using the gamification-mediated algorithm teaching method.

Therefore, the gamification inclusion in algorithm teaching-learning processes proved itself efficient to stimulate the computer higher education students' cognitive development. Through these processes, the students developed scientific reasoning cognitive skills in the concrete, intermediary, transitory, formal and post-formal levels, contributing, therefore, as a significant improvement indicator in algorithm teaching-learning processes. That implies in the mental and intellectual development of who learns.

References

Acar, Ö. (2014). Scientific reasoning, conceptual knowledge, & achievement differences between prospective science teachers having a consistent misconception and those hav-

ing a scientific conception in an argumentation-based guided inquiry course. *Learning and Individual Differences*, 30:148–154.

Acar, Ö. and Patton, B. R. (2016). Examination of learning equity among prospective science teachers who are concrete, formal and postformal reasoners after an argumentation-based inquiry course. *Australian Journal of Teacher Education*, 41(2):n2.

Acar, Ö., Türkmen, L., and Bilgin, A. (2015). Examination of gender differences on cognitive and motivational factors that influence 8th graders' science achievement in turkey. *Eurasia Journal of Mathematics, Science & Technology Education*, 11(5):1027–1040.

Agresti, A. and Kateri, M. (2011). *Categorical data analysis*. Springer.

Ding, L., Wei, X., and Liu, X. (2016). Variations in university students' scientific reasoning skills across majors, years, and types of institutions. *Research in Science Education*, 46(5):613–632.

Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*. John Wiley & Sons.

Jensen, J. L., Neeley, S., Hatch, J. B., and Piorczynski, T. (2017). Learning scientific reasoning skills may be key to retention in science, technology, engineering, and mathematics. *Journal of College Student Retention: Research, Theory & Practice*, 19(2):126–144.

Khandelwal, S., Sripada, S. K., and Reddy, Y. R. (2017). Impact of gamification on code review process: An experimental study. In *Proceedings of the 10th Innovations in Software Engineering Conference*, pages 122–126. ACM.

Knutas, A., van Roy, R., Hynninen, T., Granato, M., Kasurinen, J., and Ikonen, J. (2017). Profile-based algorithm for personalized gamification in computer-supported collaborative learning environments. *on CEUR in Proceedings of GHItaly*.

Lawson, A. (2000). Classroom test of scientific reasoning: Multiple choice version. *Author, Arizona State University. OpenURL.*

Lawson, A. E. (2004). The nature and development of scientific reasoning: A synthetic view. *International Journal of Science and Mathematics Education*, 2(3):307.

Lopes, B., Duarte, W., Nogueira, T., Lopes, R., and Ferreira, D. (2016). Método de ensino de programação mediada por simulação: Um estudo de caso no curso técnico integrado em informática. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 27, page 340.

Opitz, A., Heene, M., and Fischer, F. (2017). Measuring scientific reasoning–a review of test instruments. *Educational Research and Evaluation*, 23(3-4):78–101.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., and Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences*, 191:1479–1482.

Pessoni, V. V., Federson, F. M., and Vincenzi, A. M. R. (2015). Learning difficulties in computing courses: Cognitive processes assessment methods research and application.

In *Proceedings of the XI Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective*, volume 1, page 5.

Piaget, J., Inhelder, B., and Piaget, J. (2013). *The growth of logical thinking from childhood to adolescence: An essay on the construction of formal operational structures*, volume 84. Routledge.

Piraksa, C., Srisawasdi, N., and Koul, R. (2014). Effect of gender on student's scientific reasoning ability: A case study in thailand. *Procedia-Social and Behavioral Sciences*, 116:486–491.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11):60–67.

Seng, W. Y. and Yatim, M. H. M. (2014). Computer game as learning and teaching tool for object oriented programming in higher education institution. *Procedia-Social and Behavioral Sciences*, 123:215–224.

Topîrceanu, A. (2017). Gamified learning: A role-playing approach to increase student in-class motivation. *Procedia Computer Science*, 112:41–50.

Tsai, H.-C., Jou, M., Wang, J., and Huang, C.-C. (2016). An empirical study on the incorporation of app and progressive reasoning teaching materials for improving technical creativity amongst students in the subject of automatic control. *Computers in Human Behavior*.

Grupo de Estudo, Pesquisa e Extensão em Robótica e Automação Como Fator Motivacional Para Estudantes de Computação

Matheus E. Franco[1], Breno M. Barra[1], Rosana A. Moreira[1], Caio C. Dias[1]

[1]Departamento de Computação - IFSULDEMINAS- Campus Machado
CEP - 37.750-000 – MG – Brasil

matheus.franco@ifsuldeminas.edu.br, brenno_barra@hotmail.com,
rooh.moreiraa@gmail.com, caiodias910@gmail.com

Abstract. *In Computer courses the difficulties encountered by students and the high dropout rates are notorious. Thus, it is necessary to develop activities that privilege the contemporary scenario with the objective of motivating and externalizing the students' creativity. This paper describes the creation and implementation of a study, research and extension group in robotics and automation using the Arduino and the LEGO Mindstorms robotics kit. The results show that the proposal can be an interesting resource to develop in academics computational communities in view of the high agreement obtained.*

Resumo. *Nos cursos de computação são notórias as dificuldades encontradas pelos estudantes e as altas taxas de evasão. Assim, é necessário o desenvolvimento de atividades que privilegiem o cenário contemporâneo com o objetivo de motivar e externar a criatividade dos estudantes. Este trabalho descreve a criação e implantação de um grupo de estudo, pesquisa e extensão em robótica e automação utilizando a plataforma Arduino e o Kit de robótica LEGO Mindstorms. Os resultados obtidos demonstram que a proposta pode ser um recurso interessante a se desenvolver nas comunidades acadêmicas de computação tendo em vista a alta concordância obtida.*

1. Introdução

A partir dos anos 2000 vivenciamos uma expansão considerável na oferta de cursos de Computação no Brasil, seja de nível médio ou superior [INEP 2014]. Apesar deste crescimento, o índice de evasão em cursos superiores da área chega a ultrapassar 80% [Hoed 2017], sendo que dentre os diferentes fatores que influenciam a evasão, encontra-se a desmotivação dos alunos em razão do estilo de ensino utilizado e a aprendizagem esperada [Csikszentmihalyi and Wong 2014].

Este cenário possui grande relevância, pois estudos encomendados pela Cisco apontam um déficit superior a 195 mil profissionais na área de computação somente no Brasil em 2015 [Cisco 2016]. Desta maneira, considerando a evolução da robótica, automação e das tecnologias de informação e comunicação, o contexto educacional do ensino de computação exige transformações visando o desenvolvimento de novos métodos que privilegiem o cenário do mundo contemporâneo. Torna-se necessário

desenvolver atividades que motivem e instiguem os alunos a aprender novas habilidades, assimilar novos conceitos, avaliar novas situações, lidar com o inesperado.

A realização de diferentes atividades é necessária, pois o processo de aprendizagem não é vivenciado por todos os estudantes da mesma maneira, sendo que cada aluno pode desenvolver um diferente estilo de aprendizagem enfatizando algumas habilidades sobre as outras [Felder and Silverman 1988]. Uma vez que os alunos possuem diferentes estilos de aprendizagem, é interessante desenvolver métodos e atividades com características diferenciadas utilizando variados formatos além dos tradicionais expositivistas ministrados em sala de aula.

Neste contexto, a realização de atividades multidisciplinares que envolvam computação, automação e robótica é uma prática consolidada, sendo que trabalhos recentes relatam experiências nesta vertente [Correll et al. 2013; Monteiro et al. 2016; Oliveira Maia et al. 2014]. Estudos sugerem a robótica como alternativa para a melhor aprendizagem de diferentes disciplinas, pois a mesma tem se mostrado útil para auxiliar professores e estudantes no processo de ensino aprendizagem de conteúdos desde Computação até outras áreas como Matemática e Física [Pimentel et al. 2015; Rocha et al. 2013].

Segundo Lessa et al. (2015), os estudantes de computação possuem grande interesse pela robótica e automação, pois, estes conteúdos oferecem a possibilidade de materialização daquilo que, até então, era visível somente na tela do computador, permitindo que o aluno reflita, manuseie, construa e execute na prática conceitos abstratos. Assim, a robótica aplicada ao ensino se mostra como uma forma de implementar práticas construtivistas, pois ela permite que o aluno alcance o aprendizado através da busca e investigação [Ribeiro et al. 2011], se mostrando adequada a projetos de ensino, pesquisa e extensão.

O trabalho de Ryan e Deci (2000) citado por Silva, Tedesco e Melo (2014), descreve que motivação representa uma tendência natural para buscar novos desafios e exercitar as próprias capacidades se envolvendo em uma atividade por ser interessante, envolvente ou geradora de satisfação. Nessa perspectiva, projetos que envolvam robótica e automação possuem os elementos necessários para explorar a motivação dos estudantes.

Considerando o exposto, este trabalho descreve a criação e implantação de um grupo de estudo, pesquisa e extensão em robótica e automação como fator motivacional para estudantes de computação utilizando a plataforma de prototipagem de hardware Arduino e o Kit de robótica educacional LEGO Mindstorms. A área definida para a criação do grupo justifica-se pelo fato da robótica e automação possibilitarem que estudantes interajam com o hardware para o qual desenvolvem o software, sendo a lógica inerente na montagem e programação dos robôs e sistemas de automação, envolvendo problemas do mundo real como o da internet das coisas, estimulando assim o aprendizado.

O grupo realiza atividades caracterizadas primordialmente como extensão universitária que consiste em um processo educativo, cultural e científico que articula o ensino e a pesquisa de forma indissociável e viabiliza a relação transformadora entre Universidade e Sociedade [BRASIL 2011]. Assim, a proposta vai de encontro com

discussões importantes no contexto da indissociabilidade entre ensino-pesquisa-extensão, caracterizados como norteadores das universidades brasileiras.

2. Arduino

Diante dos objetivos do grupo de estudo, pesquisa e extensão, utilizou-se a placa de prototipagem Arduino. Segundo seu cofundador, o Arduino é uma plataforma de prototipagem de hardware aberta com base em uma placa simples de entrada/saída e um ambiente de desenvolvimento baseado nas linguagens C/C++. O Arduino pode ser utilizado para desenvolver objetos interativos independentes, ou conectado a softwares de computador. As placas podem ser montadas manualmente, ou compradas pré-montadas [Banzi 2011].

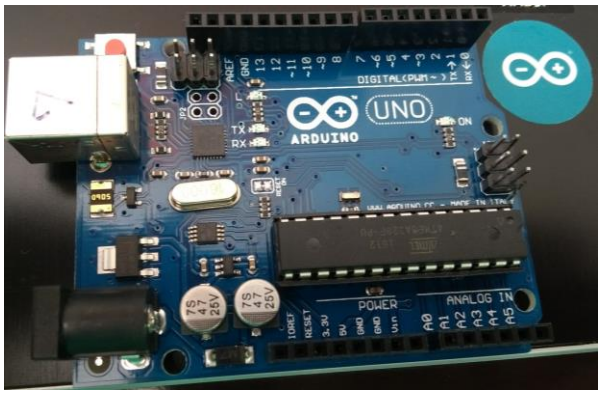


Figura 1. Placa Arduino Uno utilizada pelo grupo

A plataforma de hardware Arduíno é de fácil utilização, permitindo a interação com o usuário através de diferentes componentes eletrônicos como sensores, atuadores e leds. Sensores e atuadores utilizados na plataforma do Arduino aproximam as ideias de algoritmos que utilizam estruturas de repetição e/ou de decisão para resolver problemas específicos da realidade, sendo assim um meio de melhorar a capacidade de percepção dos alunos que determinados passos podem resolver problemas cotidianos, sendo utilizados em projetos de diferentes áreas do conhecimento [Zanetti and Oliveira 2015].

3. LEGO Mindstorms

Para inserção de conceitos de robótica, em nosso grupo foi utilizado o kit LEGO Mindstorms NXT 2.0, um kit voltado para a área educacional, o qual permite criar e programar robôs, podendo realizar tarefas simples e complexas. O kit utilizado possui diversas peças para montagens de robôs, como itens estruturais, servo motores, sensores e o brick NXT. O brick NXT é um microprocessador que atua como cérebro do robô. A programação é feita em computadores, pelo uso da linguagem em blocos NXC, já inclusa no kit. Estes aspectos possibilitam ao estudante a montagem de diversos modelos, estimulando a criatividade na resolução de problemas [Baum 2013].

Figura 2. Modelo robótico montado pelo grupo com o Kit LEGO Mindstorms

O Kit de robótica LEGO Mindstorms possui uma grande popularidade, sendo utilizado em diferentes aplicações, não somente na área robótica, mas também para o ensino de conteúdos como computação, automação, física, matemática, entre outros [Aguiar et al. 2015; Maia et al. 2008; Williams 2003].

4. Trabalhos Relacionados

Já foram realizados diversos trabalhos que envolvam robótica e automação utilizando a plataforma Arduino e o Kit LEGO Mindstorms em sala de aula, porém relatos relacionando estes tópicos a projetos de extensão são limitados. Nesta sessão são apresentados trabalhos que descrevem as aplicações acadêmicas do kit robótico da LEGO e da plataforma Arduino.

No trabalho de Aguiar et al. (2015), os autores utilizaram do Kit LEGO Mindstorms com vistas a tornar o ensino da programação algo mais simplificado e interessante. Para tanto, os autores realizaram um trabalho com alunos de nível fundamental. Após familiarização com os materiais e softwares, foram ministradas aulas contendo introdução à lógica de programação, utilização de fluxogramas, explicação do kit e atividades de desenvolvimento. Ao concluir as aulas, propuseram um desafio, em que o robô deveria percorrer um trajeto com o uso de sensores de cor e distância, trabalhando todo o conteúdo estudado. O estudo concluiu que a realização do curso foi proveitosa para os alunos do ensino básico. Também se averiguou que o conhecimento dos alunos se nivelou baseado em testes aplicados antes e depois do curso, onde o desvio padrão entre as notas diminuiu. Os autores ainda relataram que em curto prazo, a lógica de programação e a robótica podem auxiliar esses alunos em avaliações que exijam pensamento lógico e abstrato.

Williams (2003) afirma que estudantes tendem a assimilar melhor o que lhes é ensinado quando a informação é entregue de acordo com seu estilo de aprendizagem. Estudantes ativos e sensitivos, ao contrário dos refletivos e intuitivos, aprendem melhor quando precisam utilizar suas mãos e olhos no processo de aprendizagem. Na busca por atender ambas as partes, o autor introduziu a alunos do curso de Engenharia em Computação o ensino da linguagem de programação C e sistemas embarcados através do Kit LEGO Mindstorms. O objetivo da disciplina foi fazer com que os alunos programassem, em equipes, dois robôs para realizar uma simples tarefa de forma cooperativa. Com este estudo, concluiu-se que os alunos responderam favoravelmente

ao uso do Kit LEGO Mindstorms no ensino da linguagem C, apontando que 60% deles recomendariam o uso da ferramenta em usos futuros.

Em trabalho realizado na Universidade de Miami no curso de Sistemas Embarcados [Jamieson 2010], os alunos deveriam escolher entre Arduino, microprocessadores PIC ou placas de prototipagem DE2 FPGA. Mais de 90% dos alunos escolheram a plataforma Arduino. Os projetos desenvolvidos foram variados, onde fizeram uso de FPGA, Xbox Kinect e interface com um computador. O objetivo principal foi fazer com que os estudantes compreendessem sistemas embarcados e seus vários tópicos e, posteriormente, pudessem projetar estes sistemas. O trabalho chegou à conclusão que a plataforma Arduino pode ser utilizada para expor os estudantes diferentes tópicos. O autor ainda aponta que os estudantes puderam escolher a plataforma com qual desejaram trabalhar, o que resultou em uma maior preferência pelo Arduino.

De acordo com Cavalcante et al. (2014) através da tecnologia pode-se instigar o educando a aprender de modo mais eficaz. Em pesquisa desenvolvida pelos autores no Instituto Federal de Educação, Ciência e Tecnologia da Bahia, alunos de graduação em Engenharia Elétrica utilizaram a plataforma Arduino na disciplina de Probabilidade e Estatística. A pesquisa apontou muitas vantagens, tais quais: visualização, plotagem e interpretação de gráficos; realização de cálculos como média, desvio padrão, máximos e mínimos em tempo real; maior ganho de tempo. A pesquisa concluiu que, introduzir e aliar tecnologia com educação é uma ideia factível e de boa aceitação, especialmente tendo em vista seu custo financeiro acessível.

5. Metodologia

O grupo de estudo, pesquisa e extensão em Robótica e Automação teve suas atividades iniciadas em maio de 2016 com a participação de oito alunos e ao final do no de 2017 possuía dezenove participantes desenvolvendo atividades caracterizadas em sua maioria como extensão, sendo oito estudantes de nível técnico e onze estudantes de nível superior.

Para implantação do grupo, utilizou-se um conjunto de 4 (quatro) kits de robótica educacional LEGO Mindstorms NXT 2.0 e 20 (vinte) placas de prototipagem de hardware Arduino em conjunto com sensores, atuadores e componentes eletrônicos para elaboração dos projetos. O grupo é composto por alunos dos cursos técnico em Informática integrado ao ensino médio, bacharelado em Sistemas de Informação e Licenciatura em Computação, o qual se reúne semanalmente contemplando uma carga horária mensal em torno de 16 (dezesseis) horas.

Dentre as atividades desenvolvidas pelo grupo pode-se citar: (a) realização de cursos sobre o tema para alunos de escolas publicas e da própria instituição, (b) construção de protótipos para diferentes áreas como automação residencial e agrícola baseado em internet das coisas, (c) criação de modelos de aprendizagem para programação de computadores utilizando o Kit LEGO Mindstorms e a plataforma Arduino, (d) consultoria a outros setores da instituição para automação de processos e captura de dados através de sensores.

Nosso grupo se caracteriza pelo desenvolvimento de atividades que privilegiem a interação aluno-aluno, seja no desenvolvimento de projetos ou enquanto os mesmos

ministram cursos para a comunidade. Este método de trabalho é importante, pois atividades colaborativas são um dos preditores de maior peso na intenção dos alunos de computação em continuar seu curso superior [Barker et al. 2009].

Para avaliação da proposta, realizou-se um levantamento qualitativo com os alunos participantes do grupo utilizando-se a metodologia de avaliação de objetos de aprendizagem descrita por Savi et al. (2010) a qual é baseada na taxonomia de Bloom para avaliação da aprendizagem [Bloom et al. 1984] e no modelo de Keller para avaliação da motivação [Keller 2009]. Este instrumento permite auxiliar a avaliação de objetos de aprendizagem e aquisição do conhecimento através de um questionário para mensurar itens como (a) atenção, (b) compreensão, (c) divertimento, (d) desafio e (e) motivação com objetivo de obter dados relacionados ao incremento da motivação e aprendizado dos alunos participantes do grupo.

Assim, tendo em vista o objetivo de mensurar o envolvimento dos estudantes e seus sentimentos de confiança e motivação acerca dos conteúdos da área de computação, automação e robótica, foi elaborado um questionário contendo 15 questões baseadas na escala de Likert de 1 a 5 (1 - para Discordo Totalmente; 5 - para Concordo Totalmente), e uma questão do tipo verdadeira ou falsa. O questionário aplicado e a divisão nas dimensões de análise dos resultados estão acessíveis pela url https://goo.gl/pvvs7D.

6. Resultados e Discussões

Na Figura 3 são apresentados alguns trabalhos desenvolvidos pelos integrantes do grupo. Na url https://goo.gl/photos/L3DXHGMM8JVfGrpJ7 são apresentadas algumas fotos de atividades desenvolvidas.

Figura 3. Trabalhos desenvolvidos pelo grupo

Ao final de cada semestre letivo os estudantes participantes recebem um certificado o qual podem contabilizar como atividades complementares. Tendo isto em vista, os mesmos foram questionados se participam do grupo por necessidade de algum

certificado. Como se pode observar na Figura 4, apenas 5% (correspondente a um aluno) respondeu que participa por este motivo, o que indica que os mesmos participam do grupo por buscar novos desafios e para exercitar as próprias capacidades, algo descrito por Ryan e Deci (2000) e observado na alta concordância com as dimensões de desafio e compreensão observados na Figura 5.

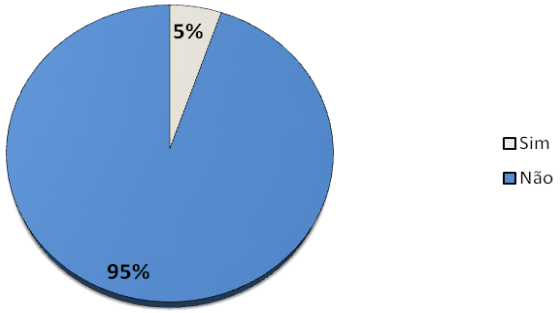


Figura 4. Gráfico de concordância com a afirmação "Participo do grupo por alguma imposição ou necessidade de certificados"

O gráfico apresentado na Figura 5 demonstra a distribuição da intensidade de concordância atribuída pelos estudantes a cada uma das dimensões avaliadas.

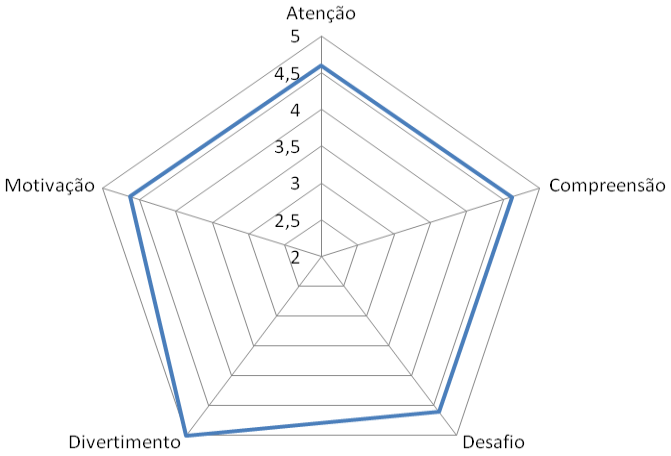


Figura 5. Gráfico radar das dimensões avaliadas pelos estudantes participantes do grupo

A partir dos resultados obtidos, observou-se uma alta concordância dos estudantes com as questões apresentadas, destacando-se a concordância máxima para as questões da dimensão de divertimento, o que vai de encontro com o trabalho de Somyürek (2015) o qual descreve em sua pesquisa que a utilização de kits robóticos oferece oportunidades para aprofundar a compreensão dos alunos de vários conceitos com exploração prática resultando em diversão e prazer.

Ademais, cabe-se destaque a uma questão avaliada na dimensão de motivação dos estudantes tendo em vista os objetivos deste trabalho. Para a questão "*Participar deste grupo me motiva a permanecer na área de computação /automação/robótica*",

houve um nível de concordância máxima por parte de 74% dos estudantes, outros 21% apresentaram uma concordância de intensidade 4 (quatro), enquanto apenas um estudante (5%) não apresentou concordância com a afirmação (Figura 6). O cenário obtido corrobora com Barker et al. (2009) que descreve em seu trabalho que o desenvolvimento de atividades práticas que privilegiam a interação aluno-aluno são preditores na motivação dos mesmos em continuar um curso na área de computação.

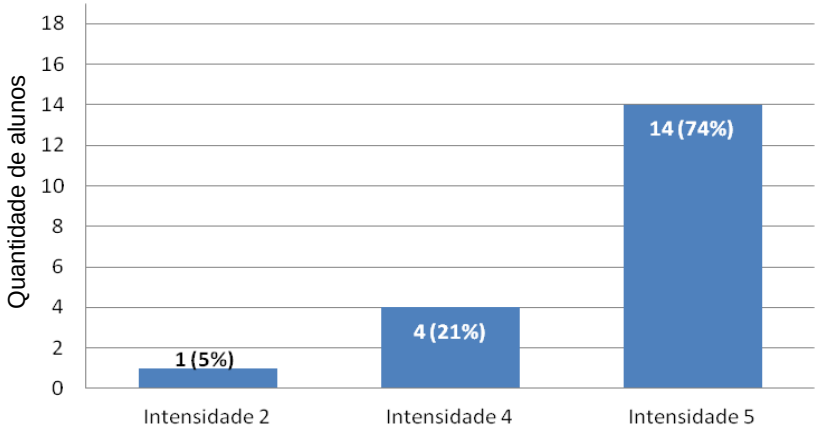


Figura 6. Gráfico de concordância com a afirmação "Participar deste grupo me motiva a permanecer na área de computação /automação/robótica"

7. Considerações Finais

Diante dos resultados obtidos e atividades desenvolvidas, constatou-se que a abordagem proposta tem, de fato, capacidade de potencializar a motivação dos estudantes de computação, não somente no ensino superior, mas também no ensino técnico integrado. O fator motivacional e de divertimento manifestados pelos alunos foram preponderantes dentre os aspectos que confirmam a capacidade da proposta. Além de se sentirem motivados, os educandos possuem a oportunidade de externar sua criatividade com a plataforma de hardware Arduino e o kit robótico da LEGO, seja em trabalhos propostos, cursos ministrados ou em projetos independentes.

Nossa avaliação sobre o grupo de estudo, pesquisa e extensão ainda é limitada tendo em vista o caráter experimental da proposta e a população pequena analisada, porém os dados obtidos demonstram que este pode ser um recurso interessante a se desenvolver nas comunidades acadêmicas de computação. Com a inclusão de inovações tecnológicas, como conteúdos relacionados à internet das coisas, o estudante procura com maior empenho o conhecimento e se interessa em participar mais daquilo que lhe foi proposto, seja por curiosidade, por necessidade em sua área profissional ou, puramente, para sua diversão.

Agradecimentos

Ao Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – IFSULDEMINAS pelo apoio concedido.

Referências

Aguiar, Y. Q., Maciel, B. K., Matos, S. D. G., Soares, L. B. and Oliveira, V. M. (2015). Introdução à Robótica e Estímulo à Lógica de Programação no Ensino Básico Utilizando o Kit Educativo LEGO® Mindstorms. In *Anais dos Workshops do CBIE*.

Banzi, M. (2011). Primeiros passos com o Arduino. *São Paulo: Novatec*, p. p1.

Barker, L. J., McDowell, C. and Kalahar, K. (2009). Exploring factors that influence computer science introductory course students to persist in the major. In *ACM SIGCSE Bulletin*. ACM.

Baum, D. (2013). *Dave Baum's Definitive Guide to Lego Mindstorms*. APress.

Bloom, B. S., Krathwohl, D. R. and Masia, B. B. (1984). *Bloom taxonomy of educational objectives*. Pearson Education.

BRASIL (2011). Plano Nacional de Extensão Universitária.

Cavalcante, M. M., Silva, J. L. de S., Viana, E. C. and Dantas, J. R. (2014). A plataforma Arduino para fins didáticos: estudo de caso com recolhimento de dados a partir do PLX-DAQ. In *XXXIV Congresso da Sociedade Brasileira de Computação (CSBC 2014), Brasília, DF*.

Cisco (2016). The Network Skills in Latin America - White Paper.

Correll, N., Wing, R. and Coleman, D. (2013). A one-year introductory robotics curriculum for computer science upperclassmen. *IEEE Transactions on Education*, v. 56, n. 1, p. 54–60.

Csikszentmihalyi, M. and Wong, M. M. (2014). Motivation and academic achievement: The effects of personality traits and the quality of experience. *Applications of flow in human development and education*. Springer. p. 437–465.

Felder, R. M. and Silverman, L. K. (1988). Learning and teaching styles in engineering education. *Engineering education*, v. 78, n. 7, p. 674–681.

Hoed, R. M. (2017). Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de computação - Dissertação (mestrado) - UNB.

INEP (2014). Censo da Educação Superior - Resumo Técnico.

Jamieson, P. (2010). Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Proc. FECS*, p. 289–294.

Keller, J. M. (2009). *Motivational design for learning and performance: The ARCS model approach*. Springer Science & Business Media.

Lessa, V., Forigo, F., Teixeira, A. and Licks, G. P. (2015). Programação de Computadores e Robótica Educativa na Escola: tendências evidenciadas nas produções do Workshop de Informática na Escola. In *Anais do Workshop de Informática na Escola*. CBIE.

Maia, L. D. O., Da Silva, V. J., Rosa, R. E. V. de S., De Lucena Junior, V. F. and De Queiroz Neto, J. P. (2008). A Robótica como Ambiente de Programaç ao Utilizando o Kit Lego Mindstorms. In *Simposio Brasileiro de Informática na Educação - SBIE*.

Monteiro, D., Bremgartner, V., Lima, H. and Salgado, N. (2016). Uma Experiência do

Uso Do Hardware Livre Arduino no Ensino De Programação De Computadores. In *Anais do Workshop de Informática na Escola*. CBIE.

Oliveira Maia, R., Da Silva, F. A., Pazoti, M. A., De Almeida, L. L. and Pereira, D. R. (2014). DESENVOLVIMENTO DE UM DISPOSITIVO PARA APOIO AO ENSINO DE COMPUTAÇÃO E ROBÓTICA. *Colloquium Exactarum*, v. 6, n. 2.

Pimentel, C., Sampaio, F. and Revoredo, K. (2015). Mecatrônica educacional apoiando o aprendizado de conceitos de física e matemática: Um estudo de caso. In *Anais do Workshop de Informática na Escola*. . CBIE.

Ribeiro, P. C., Martins, C. B. and Bernardini, F. C. (2011). A Robótica como Ferramenta de Apoio ao Ensino de Disciplinas de Programação em Cursos de Computação e Engenharia. In *Anais do Workshop de Informática na Escola*. CBIE.

Rocha, F. S., Maranghello, G. F. and Lucchese, M. M. (2013). Acelerômetro eletrônico e a placa Arduino para ensino de física em tempo real. *Caderno Brasileiro de Ensino de Física*, v. 31, n. 1, p. 98–123.

Ryan, R. M. and Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, v. 25, n. 1, p. 54–67.

Savi, R., Von Wangenheim, C. G., Ulbricht, V. and Vanzin, T. (2010). Proposta de um modelo de avaliação de jogos educacionais. *RENOTE*, v. 8, n. 3.

Silva, T. S. C., Tedesco, P. C. and Melo, J. C. B. (2014). A importância da motivação dos estudantes e o uso de técnicas de engajamento para apoiar a escolha de jogos no ensino de programação. In *Simpósio Brasileiro de Informática na Educação*.

Somyürek, S. (2015). An effective educational tool: construction kits for fun and meaningful learning. *International Journal of Technology and Design Education*, v. 25, n. 1, p. 25–41.

Williams, A. B. (2003). The qualitative impact of using LEGO MINDSTORMS robots to teach computer engineering. *IEEE Transactions on Education*, v. 46, n. 1, p. 206.

Zanetti, H. and Oliveira, C. (2015). Práticas de ensino de Programação de Computadores com Robótica Pedagógica e aplicação de Pensamento Computacional. In *Congresso Brasileiro de Informática na Educação*.

Produção de Jogos Digitais Educativos por Alunos do Ensino Superior: Um Relato da Experiência

Kleber Tavares Fernandes[1,2], Eduardo Henrique da Silva Aranha[1], Márcia Jacyntha Nunes Rodrigues Lucena[1]

[1]Programa de Pós-Graduação em Sistemas e Computação – PPgSC
Universidade Federal do Rio Grande do Norte – UFRN
Caixa Postal 1524 –Lagoa Nova – CEP 59072-970 – Natal/RN – Brasil

[2]Departamento de Ciências Exatas e Tecnologia da Informação
Universidade Federal Rural do Semiárido - UFERSA
CEP 59515-000 - Campus Angicos/RN - Brasil

Kleber76@gmail.com, eduardoaranha@dimap.ufrn.br, marciaj@dimap.ufrn.br

Abstract. *The development of a game is a complex activity that involves several phases and professionals from different areas. Some computer science courses have benefited by using digital game development practices associated with theories presented in the classroom environment. In this context, this article reports an experience of the practical training of computing students in a university using as a teaching-learning strategy the development of digital games directed for educational.*

Resumo. *O desenvolvimento de um jogo é uma atividade complexa que envolve diversas etapas e profissionais de diferentes áreas. Algumas disciplinas dos cursos de computação têm se beneficiado quando adotam práticas de desenvolvimento de jogos digitais associadas às teorias apresentadas em sala de aula. Neste contexto, este artigo relata uma experiência da formação prática de alunos da área de computação de uma Universidade Brasileira utilizando como estratégia de ensino aprendizagem a produção de jogos digitais com fins educacionais.*

1. Introdução

A maioria dos projetos pedagógicos dos cursos de computação visa a formação de profissionais que sejam capazes de analisar problemas diversos e propor soluções computacionais eficientes, rápidas e viáveis economicamente. Estes projetos valorizam a participação dos alunos em experiências práticas associadas aos conteúdos das disciplinas e contribuindo para uma formação mais próxima do que a indústria exige.

Nesse sentido as disciplinas de programação, engenharia de software, entre outras, têm buscado relacionar os conteúdos teóricos, vistos em sala de aula, com momentos práticos nos laboratórios de informática, criando oportunidade para os alunos experimentarem a prática no desenvolvimento de sistemas computacionais. Contudo, a formação oferecida constitui-se insuficiente em função da reduzida carga horária

destinada às experiências práticas e a falta de diálogo entre a academia e a indústria [Begosso et al 2011].

Diante disso, algumas pesquisas tem buscado propor momentos extracurriculares para que os alunos possam vivenciar a prática em desenvolvimento de sistemas computacionais. Esses momentos têm despertado nos alunos uma curiosidade e motivação pela área constituindo-se, muitas vezes, como um diferencial no seu desempenho e na sua formação profissional [Paes et al 2013], [Torezani et al 201] e [Nascimento et al 2010].

Neste contexto, disciplinas de programação têm se beneficiado pelo interesse dos alunos por jogos digitais. Eles se sentem cada vez mais atraídos por disciplinas com teor prático que utilizam o desenvolvimento de jogos como recurso para apresentar os conteúdos [Prensky 2012]. Apesar desta atratividade por parte dos alunos, o desenvolvimento de jogos em disciplinas práticas de computação não é uma atividade simples, pois envolve diversas etapas e profissionais de várias áreas, trazendo à tona a complexidade que contém projetos reais.

Além disso, a grande variedade dos gêneros e estilos de jogos, como também a definição dos seus requisitos tornam os projetos mais desafiadores, muitas vezes sendo necessário o envolvimento de outras disciplinas, tornando o projeto uma atividade interdisciplinar.

Portanto, segundo Prensky (2012), a estratégia do uso de jogos na formação prática de alunos dos cursos de computação tem-se mostrado como um elemento motivacional, interdisciplinar bastante envolvente. Têm buscado resolver os problemas de formação desses alunos tentando aproximar a indústria da academia, bem como proporcionam oportunidades para realização de atividades interdisciplinares durante o curso, constituindo-se como momentos de significativa aprendizagem.

Este artigo relata uma experiência da formação prática de alunos de cursos superiores da área de computação de uma Universidade Brasileira, utilizando como estratégia de ensino-aprendizagem a produção de jogos digitais com fins educacionais. Está organizado da seguinte maneira: a seção 2 descreve a fundamentação teórica utilizada neste trabalho; a seção 3 apresenta a estratégia de ensino aprendizagem através do desenvolvimento de jogos; a seção 4 relata a experiência da aplicação da estratégia com um grupo de universitários, bem como os resultados alcançados; e, finalmente, a seção 5 apresenta as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

O Brasil tem-se destacado no mercado Internacional como um dos produtores de software em potencial em razão da disponibilidade de mão de obra e da produção a baixo custo. Porém, a formação insuficiente de profissionais qualificados na área de TIC, como programadores, pode-se caracterizar como fator de entrave no desenvolvimento do País [Vivaqua, 2009].

Para evitar que isso aconteça, os cursos de computação das Universidades Brasileiras têm desempenhado um importante papel na formação de profissionais qualificados para atuar na indústria de softwares.

Preparar os alunos desses cursos para o mercado de trabalho é um desafio. A falta de diálogo entre a academia e a indústria, como também a falta de recursos das

Universidades, têm dificultado a oferta de atividades que criam oportunidades de formação mais próximas do perfil exigido pela indústria de software. Além disso, a carga horária das disciplinas de caráter prático dos cursos de computação parece ser insuficiente comprometendo a formação dos alunos. Segundo Begosso et al (2011), isso faz com que os alunos concluam seus cursos sem a maturidade que a indústria exige.

A saída adotada por algumas Instituições de Ensino Superior para minimizar esse problema tem sido a oferta de momentos de formação prática em desenvolvimento de softwares.

O trabalho de Barth et al (2011) apresenta um programa de formação ofertado semestralmente aos alunos do curso de Análise e Desenvolvimento de Software. Nesse programa, os alunos participam do desenvolvimento de um software sob demanda. Como resultado é perceptível a melhoria no desempenho dos alunos nas disciplinas, a formação de um portfólio pessoal, a prática em linguagens de programação e melhor compreensão sobre requisitos.

A pesquisa de Begosso et al (2011) descreve o projeto de implantação do Programa de Residência em Software em um ambiente de graduação, permitindo aos estudantes a participação no desenvolvimento de projetos de software para empresas reais. O trabalho constatou um aumento da maturidade dos alunos e a aproximação da academia com o mercado de trabalho.

Já Way (2005) relata uma experiência de alunos do curso de Engenharia de Software na participação de um programa de desenvolvimento de softwares. O programa proporcionou a vivência no desenvolvimento de um projeto de sistemas.

O trabalho de Sampaio et al (2005) apresenta um programa de residência de software denominado Programa de Teste de Software. Os alunos eram treinados nas diversas áreas do desenvolvimento de sistemas, porém o foco principal era a disciplina de teste de software.

A diferença da experiência relatada neste artigo para as iniciativas apresentadas acima está na estratégia da produção de jogos digitais como motivação para a prática de desenvolvimento de softwares. Também a constituição de uma equipe multidisciplinar formada por alunos e profissionais que atuaram em diversos papéis, assim como realizado na indústria de jogos.

Atualmente, pesquisas demonstram uma série de benefícios da inclusão do desenvolvimento de jogos digitais no processo ensino-aprendizagem de computação. Apontam possibilidades de tornarem-se um rico instrumento para a construção do conhecimento em diversas disciplinas. Fazer seu próprio jogo possibilita ao aluno alguns benefícios: podem ser ferramentas eficientes, pois eles divertem enquanto programam e testam os jogos, facilitam a aprendizagem e aumentam a capacidade de retenção do que é ensinado, exercitando as funções mentais e intelectuais do aluno [Sá et al 2007] e [Tarouco et al 2004].

Além disso, com o aumento da demanda por jogos cada vez mais sofisticados e a possibilidade de usá-los na educação surge a necessidade do envolvimento de outros profissionais, tais como educadores, designers, editores de áudio, entre outros, para tornar os jogos mais atrativos, divertidos e coerentes com as questões pedagógicas. O envolvimento dos alunos de computação com esses profissionais constitui-se como momentos de interação bastante ricos e com enorme potencial de aprendizado.

3. Estratégia de Ensino-Aprendizagem Através da Produção de Jogos

Os conteúdos vistos nas disciplinas dos cursos de computação de grande parte das instituições de ensino superior não têm acompanhado a evolução que essa área tem passado, nem tão pouco as demandas e os conhecimentos exigidos pela indústria. Uma das estratégias de aproximar a indústria e a academia é a oferta de programas de treinamento em desenvolvimento de software. Tais programas têm o objetivo principal de possibilitar aos alunos a capacitação nas tecnologias utilizadas pela indústria de forma prática. Os alunos participantes do programa, orientados pelos professores, vivenciaram experiências práticas em projetos de desenvolvimento de softwares, em todas as suas fases, simulando na academia um ambiente da indústria. Dessa maneira, o programa contribui para a formação e inserção desses alunos no mercado de trabalho.

Nesse sentido, a estratégia adotada pela Universidade, onde foi realizada a experiência relatada neste trabalho, foi a oferta de um programa de treinamento em desenvolvimento de jogos. Inicialmente os alunos foram selecionados para compor equipes de desenvolvimento que, sob orientação de um professor, vivenciaram experiências práticas na produção de jogos demandados pela indústria. A primeira etapa do programa foi constituída de um treinamento em desenvolvimento de jogos que contempla ferramentas, metodologias, análise e gestão de projetos. A segunda fase foi o desenvolvimento de um jogo educativo. Cada equipe teve a oportunidade de participar do programa que durou aproximadamente um semestre, podendo se estender aos semestres seguintes, até a conclusão do curso.

A equipe de produção foi constituída por alunos dos cursos de computação que foram divididos de acordo com os seguintes perfis: designers gráficos, programadores, sonoplastas, game designers e gerente de projeto. Os perfis e respectivas responsabilidades são descritos a seguir. O programador é responsável pela codificação (programação) dos jogos em uma linguagem de programação específica. Enquanto o designer gráfico é responsável pela concepção e produção da arte gráfica dos jogos, incluindo os cenários, personagens e demais elementos gráficos do jogo. Já o sonoplasta é responsável pela concepção e produção da trilha sonora dos jogos, incluindo as músicas e os sons vinculados aos elementos dos jogos. O game designer é responsável pelo projeto de criação do jogo, inclui a definição da história, roteiro, regras, animação dos personagens, planejamento da interface, além de garantir a interatividade e entretenimento do jogo. O gerente do projeto é responsável pelo gerenciamento do projeto e acompanhamento das fases do desenvolvimento dos jogos e finalmente o testador e/ou os avaliadores são responsáveis pelos testes e validação dos jogos e pode ser um pedagogo, coordenador, game designer, programador, professor e/ou aluno.

Os professores envolvidos no programa identificaram as expectativas e necessidades de desenvolvimento integral dos alunos e articularam oportunidades educativas capazes de atendê-las. Acompanharam e orientaram o desenvolvimento dos jogos. Articularam o conteúdo teórico das disciplinas com a prática da indústria.

O projeto contou também com uma pedagoga responsável pelo levantamento dos requisitos pedagógicos do jogo. Incluiu a análise do material didático, fonte de inspiração do jogo, inicialização e acompanhamento do cartão do jogo (considerado o documento de requisitos para o projeto) e a avaliação do produto final para prováveis adaptações ou melhorias.

O ambiente de desenvolvimento dos jogos foi formado pelas seguintes ferramentas: [1]*Construct* 2, que permite a criação dos jogos em 2 dimensões no formato HTML5; [2]Ilustrator, utilizado para produção dos elementos gráficos do jogo; [3]Audacity, utilizado na produção dos áudios; e [4]*Trello*, que permite o gerenciamento das tarefas e do fluxo de produção. Estas ferramentas foram escolhidas pela facilidade de uso e vasto material didático disponível na literatura e internet.

O documento de requisitos para a elaboração dos jogos denomina-se nesta experiência de "Cartão do Jogo". Nele contém a identificação do material didático utilizado como ponto de partida, público alvo, objetivos, conteúdos e habilidades a serem trabalhadas, as variações de dificuldade, formas de avaliação da aprendizagem, a proposta do jogo, itens do game designer, projeto gráfico, projeto do áudio, requisitos de programação e arquitetura, regras e créditos do jogo.

3.1. Visão Geral do Processo de Desenvolvimento

Para a produção de jogos é necessário utilizar um processo de desenvolvimento de software que se adapte à realidade do projeto. O processo utilizado no programa de treinamento foi elaborado especificamente para essa finalidade através da colaboração da equipe de alunos e professores envolvidos. Foram realizadas diversas reuniões usando a técnica de *brainstorming* para definição das fases e atividades do processo, bem como a atuação da equipe. A Figura 1 apresenta o processo de desenvolvimento de jogos usado nesta experiência de aprendizado.

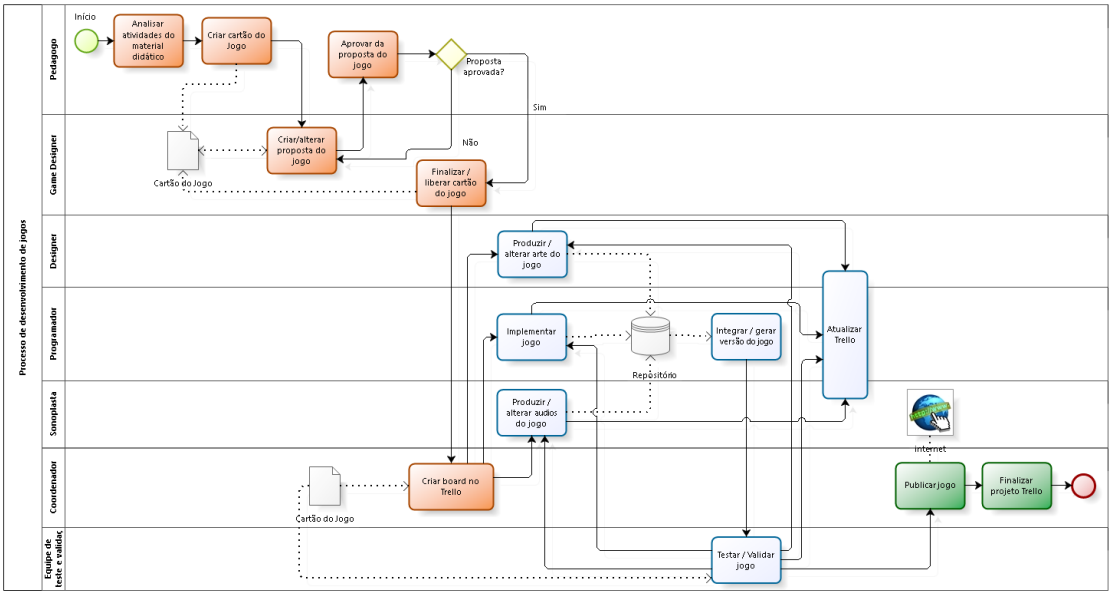


Figura 1. Processo de Desenvolvimento de Jogos

A Figura 1 apresenta as fases e atividades divididas em raias que representam os perfis dos membros da equipe envolvida no projeto. As atividades relacionadas à pré-produção estão na cor laranja, as de produção estão na cor azul e as da pós-produção

[1] https://www.scirra.com/construct2
[2] https://www.adobe.com/
[3] http://www.audacityteam.org/
[4] https://trello.com/

estão na cor verde. Algumas atividades não são necessariamente sequenciais para dar maior agilidade ao desenvolvimento do jogo.

O processo inicia na fase de pré-produção com a análise das atividades do material didático pela pedagoga. Essas atividades representam a demanda inicial, ou o que se espera que o aluno (jogador) aprenda em determinado ano ou série. Elas norteiam ou inspiraram a definição dos objetivos, habilidades ou conteúdos a serem abordados nos jogos. Vários livros didáticos da mesma série foram consultados, para ampliação das referências.

Após a análise, a pedagoga cria um cartão com os requisitos iniciais do jogo. A partir desse cartão é possível iniciar as atividades do game designer que cria a proposta de desenvolvimento do jogo. A proposta complementa do cartão do jogo é então analisada podendo ser aprovada ou não pela pedagoga, de acordo com o atendimento dos requisitos levantados. Sendo aprovada segue para finalização e liberação do cartão estando pronto para iniciar a fase de produção do jogo. Caso contrário, retorna para o game designer realizar as alterações necessárias e ser novamente avaliado.

A fase de produção é iniciada com 3 processos executados em paralelo que são: produção da arte gráfica do jogo, produção dos áudios do jogo e a implementação do jogo. A produção da arte gráfica consiste em elaborar todos os elementos gráficos do jogo. A produção dos áudios consiste na elaboração da trilha sonora e todos os sons que irão compor o jogo. A tarefa de implementar o jogo consiste na sua codificação em uma linguagem de programação.

Os produtos finais dessas 3 atividades são armazenados em um repositório do projeto. Assim, é possível integrar os elementos gráficos e sons ao código fonte e gerar a primeira versão executável do jogo.

Ainda na fase de produção são realizados os testes e validação do jogo. A partir dos testes são verificados problemas na mecânica do jogo, erros de programação, problemas de design gráfico e no áudio do jogo. Outra verificação feita é se os requisitos descritos no cartão do jogo foram atendidos. Não havendo nenhum problema detectado pelos testes o jogo é validado. Caso contrário, é encaminhado para a realização das correções necessárias. O registro dos erros é realizado no Trello. Essa tarefa é realizada por uma equipe constituída pela pedagoga, gerente do projeto, game designer, programador, professores e alunos.

À medida que o jogo vai sendo desenvolvido é possível atualizar as tarefas que foram cadastradas no Trello. A pedagoga também interage com a equipe fornecendo orientações no decorrer do processo. Dessa forma, o gerente do projeto tem condições de acompanhar a produtividade da equipe, os problemas relatados e suas soluções. A fase final do processo é a pós-produção que possui como tarefas principais a publicação do jogo numa plataforma computacional ou site na Internet e a finalização do projeto no Trello.

4. Relato da Experiência da Produção de Jogos

O ponto de partida para iniciar este trabalho foi o desenvolvimento de jogos digitais educativos a partir do material didático do componente curricular de matemática, das séries iniciais do ensino fundamental, por alunos do Ensino Superior. Tivemos como demandante uma escola pública da rede municipal de ensino que reuniu sua equipe de

professores e coordenadores para definir os requisitos necessários para elaboração dos jogos.

O componente curricular da matemática foi escolhido em razão do baixo rendimento dos alunos nas avaliações realizadas pela escola. Ela espera que a aplicação dos jogos desenvolvidos nas aulas de matemática possa contribuir para a aprendizagem dos alunos e melhoria dos índices avaliativos.

Os jogos produzidos seguiram o processo de desenvolvimento de jogos educativos proposto pelo programa de treinamento. Dessa forma, pretendeu-se garantir uma produção padronizada, em curto prazo de tempo e que pudesse atender efetivamente aos requisitos do projeto.

A equipe foi formada por 17 alunos do curso superior de computação de uma Universidade Brasileira sendo dividida em 07 grupos de produção. Cada grupo foi constituído por alunos com o perfil de desenvolvedor, design gráfico, game designer e sonoplasta. O programa contou com um profissional da pedagogia que prestava o suporte pedagógico aos grupos. Todos eram acompanhados por um aluno que fazia o papel de gerente de projetos.

A produção dos jogos foi realizada em ciclos produtivos. Cada um deles com duração média de 4 semanas de trabalho. Cada grupo era responsável por desenvolver 1 jogo por ciclo, cumprindo todas as etapas de produção já descritas. A experiência foi realizada num período de 3 ciclos produtivos. A equipe e os ciclos produtivos foram organizados e gerenciados pelo gerente de projetos utilizando uma ferramenta de gestão de tarefas (Trello). A cada semana eram realizadas reuniões para monitorar o progresso da produção dos jogos, semelhante as sprints da metodologia [5]Scrum.

Foram realizadas reuniões avaliativas onde a equipe avaliava a eficiência do processo e a qualidade dos jogos produzidos ao final de cada ciclo realizado. Para iniciar o ciclo seguinte, era necessário fechar o ciclo anterior, fazendo os ajustes ou alterações percebidas nos testes e avaliação dos jogos.

4.1. Avaliação dos Resultados Alcançados

Esta subseção apresenta os resultados alcançados, a avaliação dos jogos produzidos e do processo utilizado. Essa atividade foi realizada através da análise dos questionários e entrevistas respondidas pelas equipes participantes do projeto. Ao final de cada ciclo de desenvolvimento foi aplicado um questionário e ao final do projeto foram realizadas entrevistas com os grupos de produção.

4.1.1. Considerações a Respeito dos Jogos Desenvolvidos

A partir da análise dos jogos produzidos, é possível afirmar que as equipes estiveram voltadas para o atendimento dos objetivos, conteúdos e habilidades estabelecidos inicialmente no cartão do jogo. Na etapa de concepção do tema a ser abordado, personagens, cenários, procedimentos envolvidos no jogo, título, houve um verdadeiro desdobramento para se idealizar um produto criativo, que atendesse à faixa etária e aos objetivos pretendidos. Verificamos interfaces bem apresentadas, propostas criativas de jogo, com temas e títulos adequados aos interesses dos alunos (jogadores). Portando, observamos que a fase de pré-produção do processo proposto está clara e possui tarefas

[5] http://www.desenvolvimentoagil.com.br/scrum/

compatíveis para o desenvolvimento de jogos dessa natureza.

Ao verificar o conjunto dos jogos, percebemos a variedade dos conteúdos que foi proposto, o que possibilita a abordagem de diversos assuntos do programa das séries iniciais do ensino fundamental por meio dos jogos.

Porém, como análise do resultado da fase de produção, constatamos que 3 jogos não conseguiram corresponder ao produto idealizado, no que se refere ao propósito educacional de desenvolver habilidades e favorecer a aprendizagem. Apesar do empenho da equipe na elaboração do cartão do jogo, a atividade de implementação, em 4 jogos, foi comprometida pela limitação de conhecimento avançado da ferramenta utilizada e do tempo destinado para essa atividade. Esses jogos requereram ações mais complexas que exigiam mais tempo de implementação.

Outro ponto que merece destaque, em 5 jogos, é a baixa interação entre o jogo e o jogador, sobretudo no início, quando deveria fornecer informações sobre as regras, desafios e os controles do jogo. Percebemos também a necessidade de melhoria no feedback para atender ao propósito de comunicação e para motivar o jogador a continuar jogando.

4.1.2. Considerações a Respeito do Processo Utilizado

O resultado das entrevistas realizadas com a equipe leva a perceber que o processo utilizado nesta experiência favorece a produção completa dos jogos, indicando que a sequência de atividades foi adequada. Aponta-se que o processo permite uma comunicação efetiva entre os seus participantes.

O processo contempla a possibilidade de retorno à atividade anterior para realização de ajustes ainda durante o desenvolvimento do jogo. Isso foi apontado positivamente pelos entrevistados. As atividades produzir/alterar arte do jogo, implementar jogo e produzir/alterar áudios do jogo podem ser executadas em paralelo o que permite agilizar a fase de produção. Porém, a tarefa de integrar/gerar versão do jogo pode ser comprometida por falhas na entrega dos artefatos que a antecede.

Fazendo uma análise do processo por fases, foi possível tecer ainda algumas considerações. Por exemplo, o cumprimento adequado da fase de pré-produção pelos profissionais envolvidos. Todas as atividades foram cumpridas satisfatoriamente. As atividades da produção também foram desenvolvidas de acordo com o previsto, com ressalva para a última atividade – testar e validar jogo; A equipe de teste não foi constituída por todos os membros idealizados. Os jogos produzidos não foram testados por professores e alunos. Este fator influenciou de forma negativa a finalização dos jogos, considerando que tais atores são o público alvo de toda a produção; 11 jogos não retornaram para a validação da pedagoga, a quem caberia uma análise mais cuidadosa quanto ao atendimento dos requisitos e uma intervenção/orientação para aprimoramento. Sem tal análise, não foi possível fazer o aperfeiçoamento mais apurado, que ficou centrado nas percepções das demais pessoas da equipe, as quais não tinham formação pedagógica. As deficiências nos testes comprometeram o êxito da fase de pós-produção, tendo em vista que sem a finalização adequada, os jogos não podiam ser publicados e disponibilizados para o uso.

4.1.3. Considerações a Respeito da Estratégia de Formação dos Alunos

A realização do programa de treinamento em desenvolvimento de jogos constituiu-se como uma experiência formativa para todos os envolvidos: alunos e professores.

Os depoimentos relatados nas entrevistas realizadas com os alunos acerca da sua participação e aprendizagem durante a experiência mostra que foram proporcionadas condições muito aproximadas da realidade com a qual poderão se deparar na indústria. Isso porque, estiveram envolvidos nesse programa: atendimento à demanda de clientes, necessidade de alinhamento de procedimentos adotados, necessidade de clareza e definição dos papéis de todos os membros da equipe, necessidade de tomar decisões em grupo, avaliar os resultados, retomar o processo e validar os jogos. Para os alunos, foi possível aprender na prática os conteúdos envolvidos em cada item do programa.

Para os professores, a estratégia adotada proporcionou uma formação mais consistente aos alunos, por possibilitar a associação entre teoria e prática, além da aquisição de expertise no desenvolvimento de jogos. A atuação em diversos perfis também foi importante, pois possibilitou compartilhar ideias e buscar saberes de áreas distintas em prol de um objetivo em comum: a criação de jogos para uma clientela definida. Tal vivência, certamente colaborou para ampliar a compreensão dos conceitos trabalhados nas disciplinas. Isso ficou evidenciado no resultado final da avaliação da aprendizagem das disciplinas dos alunos envolvidos no programa, em relação aos demais. Por essa razão, entende-se que essa experiência também trouxe contribuições para os professores, que puderam dar sentido às suas aulas e acompanhar, na prática, a atuação dos seus alunos na aplicação dos conceitos aprendidos.

5. Conclusões

Diversos autores ressaltam a importância do desenvolvimento de jogos como estratégia de ensino aprendizagem por possibilitar ao aluno a oportunidade de participar da construção do seu conhecimento [Sutherland 2014] [Savi et al 2014]. Porém, nossos estudos apontam a ausência de processos sistemáticos para desenvolver jogos digitais educativos que atendam tanto às necessidades técnicas e pedagógicas. Verifica-se a necessidade de aprofundamento do diálogo entre os profissionais de educação e de tecnologia da informação, para garantir a concepção de produtos que tanto atendam às demandas educacionais, como também promovam a inserção dos alunos no “universo tecnológico”.

A experiência aqui relatada constitui-se como modelo de consolidação da formação de estudantes dos cursos de computação. Como principal benefício proporcionado pelo programa de treinamento em desenvolvimento de softwares destaca-se o desenvolvimento em curto prazo de jogos digitais educativos com padrão de mercado e geração de experiência formativa do aluno, que inserido em um contexto que simula o ambiente profissional, irá também desenvolver relações interpessoais e valores profissionais.

Pode-se citar como resultado positivo o desenvolvimento de jogos educativos com intenções pedagógicas claras que podem ser utilizados pela escola. É possível considerar esse trabalho uma experiência piloto válida e que os resultados alcançados servem como subsidio para a realização de novos experimentos.

Como futuros trabalhos temos: aprimorar o programa levando-se em

consideração os critérios avaliados nessa experiência, de forma que se consiga superar as dificuldades que foram aqui observadas. Além disso, pretendemos melhorar o processo de desenvolvimento de jogos, validando-o através de outras experiências.

Referências

Begosso, L. R., Begosso, L. C., Poletto, A., da Cunha, D. S., and de Lima, F. C. (2011) "Programa de residência em software". In XIX Workshop de Educação em Informática, Natal, Brasil.

Fabricio J. Barth, Leo Burd, Mauricio Pimentel (2011) "Escritório de projetos: simulando o ambiente de projetos de software em cursos de tecnologia". Faculdade de Tecnologia Bandeirantes – BandTec , Sao Paulo, SP, Brasil.

Gros, Begona (2008) "The impact of digital games in education" First Monday, v.8.

Lara, I. C. M (2014) "Jogando com a Matemática de 5ª a 8ª série". São Paulo: Rêspel.

Nascimento, M., Mendonça, A., Guerrero, D., and de Figueiredo, J. (2010). Teaching programming for high school students: A distance education experience. In Frontiers in Education Conference (FIE), 2010 IEEE, pages F1J–1–F1J–6.

Paes, R., Malaquias, R., Guimaraes, M., and Almeira, H. (2013). "Ferramenta para a avaliação de aprendizado de alunos em programação de computadores". In Anais dos Workshops do CBIE 2013, Dourados, MS.

Prensky,M.(2012). "Aprendizagem Baseada em Jogos Digitais". São Paulo. Editora Senac, 1ª edição.

Sá, E.J.V, Teixeira, J.S.F, and Fernandes, C.T (2007) "Design de atividades de aprendizagem que usam Jogos como princípio para Cooperação". In: Anais do XVIII Simpósio Brasileiro de Informática na Educação (SBIE), São Paulo - SP, Brasil.

Sampaio, A. et. al. (2005) "Software Test Program: a Software Residency Experience" In International Conference on Software Engineering. Proceedings of 27th ICSE.

Savi, Rafael and Ulbricht, Vania (2008) "Jogos digitais educacionais: benefícios e desafios". CINTED – UFRGS.

Sutherland, Jeff. (2014) "Scrum - A Arte de Faze o Dobro de Trabalho na Metade do Tempo". 1ª ed. Leya Brasil, São Paulo.

Tarouco, L. M. R, Roland, L. C, Fabre, M. C. J. M, and Konrath, M. L. P. (2004) "Jogos educacionais". In: Novas Tecnologias na Educação - RENOTE, v.2, n.1.

Torezani, C., Chagas, L., and Tavares, O. (2013). "Newprog - um ambiente online para crianças aprenderem programação de computadores. In XXV Workshop de Informática na Escola (WIE 13), Campinas, SP.

Vivacqua, F. R. (2009) "Fábricas de Software e a Academia: Análise da Formação Acadêmica em Informática no Município do Rio de Janeiro". Dissertação de Mestrado da FGV - Escola Brasileira de Administração Pública e de Empresas. Rio de Janeiro. Disponível em http://virtualbib.fgv.br/dspace/handle/10438/3703

Way, T. P. (2005) "A Company-based Framework for a Software Engineering Course". In Technical Symposium on Computer Science Education. Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education. St. Louis, USA.

Validação de Modelos ER

Cody Malnor[1,2],André Chateaubriand[1], Obede Carvalho[1], Ricardo Terra[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Lavras (UFLA)

[2]Department of Mathematics and Computer Science
Northern Michigan University (NMU)

cmalnor@nmu.edu, {andrechateau,obedejc}@computacao.ufla.br,
terra@dcc.ufla.br

Abstract. *The Entity-Relationship (ER) model is widely used for teaching conceptual data modeling. During the learning process, many of the mistakes made by students are not related to the logic of the model, but to its construction. In this paper, we propose and formalize an ER model validation integrated to an academic modeling tool. This extension checks connections as soon as they are added to the model and – if invalid – not only notifies the student but also suggests the correct connections. As a contribution, we seek to make the learning process faster and more interactive for the student, in addition to reducing the teacher's correction effort.*

Resumo. *O modelo Entidade-Relacionamento (ER) é largamente adotado no ensino de modelagem de dados conceitual. Durante o processo de aprendizagem, muitos dos erros cometidos pelos alunos não estão relacionados à lógica do modelo, mas sim às suas construções. Diante disso, este artigo propõe e formaliza uma validação de modelos ER integrada à uma ferramenta acadêmica de modelagem. Esse módulo de validação é capaz de verificar conexões logo que são adicionadas ao modelo e – caso inválidas – não só notificar o aluno como sugerir as conexões corretas. Como contribuição, busca-se tornar o processo de aprendizado mais rápido e interativo para o aluno, além de reduzir o esforço de correção pelo professor.*

1. Introdução

O modelo Entidade-Relacionamento (ER) é uma das abordagens de modelagem de dados conceitual mais adotadas no ensino de disciplinas de banco de dados [2, 16]. Esse modelo é baseado em uma percepção de um mundo real construída por uma coleção de objetos básicos, chamados entidades, e os relacionamentos entre esses objetos. Existem uma ampla gama de ferramentas que auxiliam a criação de modelos ER, tais como `Dia`[1], `TerraER`[2], `EERCASE`[3] e `MySQLWorkbench`[4].

No modelo ER, obviamente, não é permitido qualquer tipo de conexão entre quaisquer dois de seus elementos. Isso implica em alunos – durante o processo de aprendizagem – cometendo uma série de erros não relacionados à lógica do modelo, mas sim, às

[1]http://www.dia-installer.de
[2]http://www.terraer.com.br
[3]https://www.sites.google.com/a/cin.ufpe.br/eercase/apresentacao
[4]https://www.mysql.com/products/workbench/

suas construções. Mais importante, esses erros podem demorar para serem identificados como também corrigidos. Este artigo, portanto, reivindica que alunos devem focar na tarefa de modelagem conceitual, sem preocupações com formalizações do modelo.

Diante disso, este artigo propõe uma validação de modelos ER integrada à uma ferramenta acadêmica de modelagem, denominada `TerraER`. Esse módulo de validação é capaz de verificar conexões logo que são adicionadas ao modelo e – caso inválidas – não só notificar o aluno como sugerir as conexões válidas por meio de uma tela de recomendação que aponta quais as possíveis conexões entre os elementos e ainda quais os possíveis elementos que podem ter tal conexão estabelecida.

Como contribuição, a validação proposta neste artigo visa tornar o processo de aprendizado mais rápido e interativo para o aluno, além de reduzir o esforço de correção pelo professor. Isso se justifica, primeiramente, pelo fato de o módulo de validação permitir o erro do aluno e imediatamente o notificar; cenário o qual instiga o aprendizado e evita a recorrência do mesmo erro. Em segundo lugar, o aluno pode se preocupar exclusivamente na tarefa de modelagem conceitual, sem preocupações com formalizações do modelo. Da mesma forma, professores, no processo de correção, podem ter a atenção voltada exclusivamente à modelagem, o que fomenta maior eficiência do processo educacional.

O artigo está dividido como a seguir. A Seção 2 introduz os conceitos fundamentais à compreensão deste estudo. A Seção 3 formaliza as conexões válidas do modelo ER, descreve o projeto e implementação da validação integrada à ferramenta `TerraER` e conduz uma avaliação sob a perspectiva educacional. Por fim, a Seção 4 conclui o estudo.

2. *Background*

Esta seção apresenta conceitos fundamentais ao entendimento deste estudo, tais como modelo ER, modelo ER estendido (EER) e a ferramenta de modelagem conceitual de alto nível `TerraER`.

2.1. Modelo ER

Segundo Garcia-Molina [7], no modelo Entidade-Relacionamento, o esquema do banco de dados é representado graficamente como um diagrama usando três principais tipos de elementos: entidades, atributos e relacionamentos.

Entidade: Uma entidade é um objeto abstrato de algum tipo, e uma coleção de entidades similares forma um conjunto de entidades [7]. Chen [2] classifica essas entidades como entidades fortes e entidades fracas. Uma entidade fraca é uma entidade cuja existência depende de alguma forma de outra entidade, de tal forma que sua existência não se justifica sem que essa outra entidade (denominada entidade proprietária) exista. Como exemplo, a Figura 1 apresenta a entidade `DEPENDENTE` na qual depende sua existência da entidade `FUNCIONARIO`, pois um dependente demanda vínculo com o funcionário e não se justifica sem ele. Ao contrário, uma entidade forte[5] é uma entidade que não é fraca, ou seja, a entidade `FUNCIONARIO` da Figura 1 não depende de nenhuma outra para existir.

[5]Alguns autores utilizam o termo "entidade regular" ao invés de "entidade forte" [3], porém neste trabalho é adotado o termo "entidade forte".

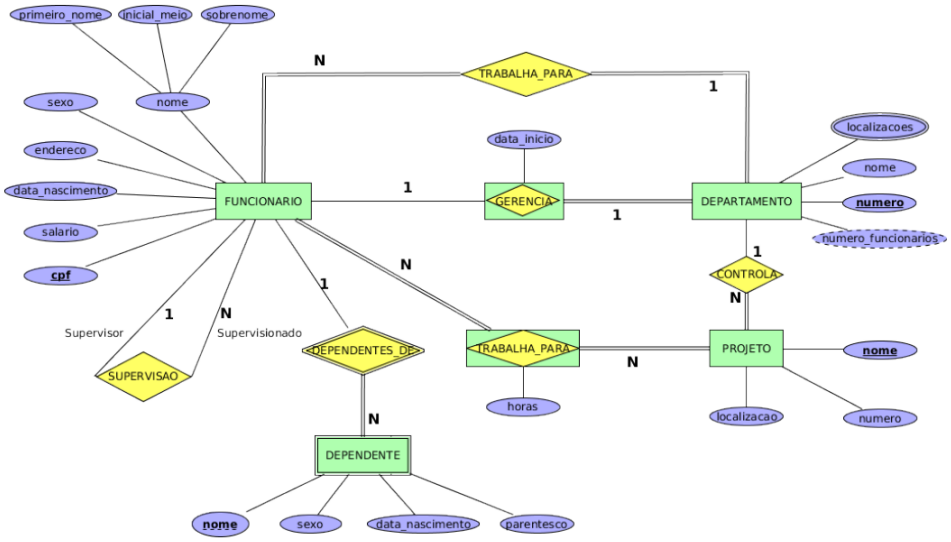


Figura 1. Diagrama ER do banco de dados de uma empresa

Atributos: Uma entidade é representada por um conjunto de atributos. Os atributos são propriedades descritivas que expressam características em comum de diferentes entidades em um conjunto de entidades. A designação de um atributo, para um conjunto de entidades, expressa que o banco de dados armazena informações semelhantes concernentes a cada entidade no conjunto de entidades; entretanto, cada entidade pode ter seu próprio valor para cada atributo [1]. Além disso, segundo Date [3], os atributos podem ser: (i) simples ou compostos; (ii) chaves; (iii) univalorados ou multivalorados; e (iv) básicos ou derivados. Em um modelo conceitual de alto nível, um atributo chave expressa uma restrição de unicidade, ou seja, cada entidade no conjunto de entidades contém valor distinto referente a esse atributo.

A Figura 1 ilustra cada um desses atributos. Por exemplo, a entidade `FUNCIONARIO` possui `salario` como atributo simples; `nome` como atributo composto, pois nome é composto de `primeiro_nome`, `inicial_meio` e `sobrenome`; e `cpf` como atributo chave. A entidade `DEPARTAMENTO` tem `localizacoes` como atributo multivalorado, pois um departamento pode-se encontrar em diferentes lugares, e `numero_funcionarios` como atributo derivado, ou seja, pode-se obter a quantidade de empregados através do somatório de todos os funcionários da entidade `FUNCIONARIO`.

Relacionamentos: Uma entidade pode se relacionar com outras entidades. Relacionamentos são conexões entre duas ou mais entidades [7]. As entidades envolvidas em determinado relacionamento são ditas participantes desse relacionamento. O número de participantes em determinado relacionamento é chamado grau desse relacionamento.

Seja R um tipo de relacionamento que envolve o tipo de entidade E como participante. Se toda instância de E participa de pelo menos uma instância de R, então a participação de E em R é considerada total; do contrário ela é considerada parcial. Relacionamentos em diagramas ER podem ser modelados de diversas formas: um para um; um para muitos; ou muitos para muitos. Na Figura 1, entidades do tipo `DEPARTAMENTO`

podem se relacionar com entidades do tipo PROJETO da seguinte forma: um departamento controla pelo menos um projeto. Assim, esse exemplo expõe a existência de um relacionamento binário um para muitos (1:N) com restrição de participação total no lado N.

O Modelo ER proposto por Peter Chen em 1976 é uma forma simples de representar problemas do mundo real de forma bem próxima da descrição semântica dos dados e as relações entre eles. O Modelo ER é composto por elementos chamados de entidades, atributos e relacionamentos.

2.2. Modelo ER Estendido

Desde o final da década de 1970, projetistas de aplicações de banco de dados têm tentado projetar esquemas de banco de dados mais precisos, que reflitam as propriedades de dados e restrições com mais precisão [5]. Diante desse fato e do sucesso do paradigma orientado a objetos, em 1986, Teorey et al. [16] propuseram uma extensão ao modelo ER acrescentando os conceitos de relacionamentos de classe/subclasse, herança de tipo, especialização/generalização (também chamadas hierarquias) e suas restrições.

A Figura 2 exemplifica o conceito de generalização e especialização. A generalização se refere ao processo de definição de um tipo de entidade generalizado com base nos tipos de entidades dados. Nessa figura, pode-se também observar a especialização ao se modelar os tipos de entidades partindo da superclasse (tipo de entidade FUNCIONARIO) e seguindo para a especificação do conjunto de subclasses (SECRETARIA, TECNICO e ENGENHEIRO).

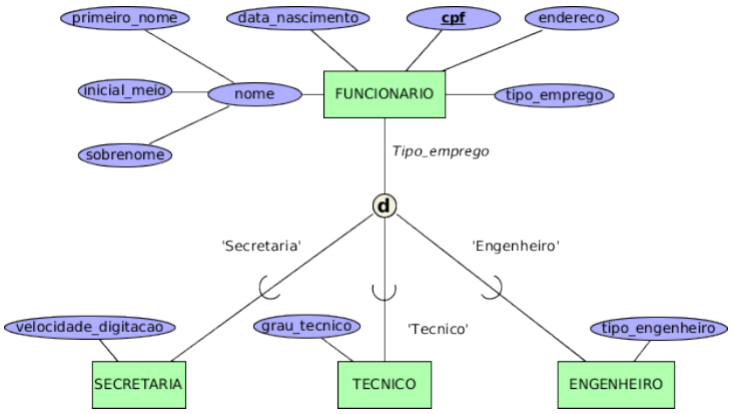


Figura 2. Diagrama EER de parte do banco de dados de uma empresa

Na Figura 2, por exemplo, uma entidade do tipo de entidade FUNCIONARIO pode ter seu emprego classificado como do tipo SECRETARIA, TECNICO ou ENGENHEIRO, o que caracteriza o tipo de entidade FUNCIONARIO como superclasse ou supertipo e, de forma análoga, os tipos de entidades SECRETARIA, TECNICO e ENGENHEIRO como subclasses ou subtipos. Vale ressaltar ainda que as entidades membros de uma subclasse (SECRETARIA, TECNICO e ENGENHEIRO) herdam todos os atributos do tipo de entidade da superclasse FUNCIONARIO).

Duas outras restrições podem se aplicar a uma especialização. A primeira é a restrição de disjunção, que especifica que as subclasses da especialização devem ser dis-

juntas. Isso significa que cada entidade pode ser membro de, no máximo, uma das subclasses da especialização. Se nas subclasses não for imposta a restrição de disjunção, seus conjuntos de entidades podem ser sobrepostos, isto é, cada entidade pode ser membro de mais de uma subclasse da especialização. A Figura 2 apresenta um exemplo de especialização com restrição de disjunção, na qual o tipo do trabalho do `FUNCIONARIO` pode ser somente um dos tipos `SECRETARIA`, `TECNICO`, ou `ENGENHEIRO`.

A segunda restrição sobre a especialização é chamada de restrição de completude, que pode classificada como total ou parcial. Uma restrição de especialização total especifica que toda entidade da superclasse precisa ter como membro ao menos uma subclasse na especialização. Já a especialização parcial permite que uma entidade da superclasse não tenha como membro qualquer uma das subclasses [5]. Portanto, podemos classificar as restrições de especialização como quatro possíveis: (i) disjunção, total; (ii) disjunção, parcial; (iii) sobreposição, total; e (iv) sobreposição, parcial. A Figura 2 exemplifica o caso de especialização com restrição de disjunção e restrição de completude parcial (caso ii supracitado), onde o tipo do trabalho do `FUNCIONARIO` pode ser somente um dos tipos `SECRETARIA`, `TECNICO`, ou `ENGENHEIRO`, além de permitir não ter como membro nenhum dos três, já que a restrição de completude é parcial.

2.3. TerraER

O `TerraER` é uma ferramenta de código aberto, voltada ao meio acadêmico, mais especificamente no auxílio ao aprendizado de disciplinas de modelagem conceitual de banco de dados [12, 13]. `TerraER` permite criar modelos conceituais de alto nível mais condizentes ao que os professores lecionam na disciplina de banco de dados. Isso pode ser constatado através da Figura 3, na qual a barra de ferramentas de objetos possui atalhos para criação de elementos do diagrama ER na notação de Peter Chen e EER, adotada por Elmasri e Navathe [5].

Devido à facilidade na criação de diagramas que utilizam modelo conceitual de alto nível e pelo fato de a ferramenta `TerraER` já ser adotada em mais de 30 Instituições de Ensino Superior (IES), essa foi escolhida para implementar o módulo de validação de Modelos ER, de forma a auxiliar professores e alunos no projeto e correção de atividades e tornar o processo de aprendizado mais rápido e interativo para o aluno.

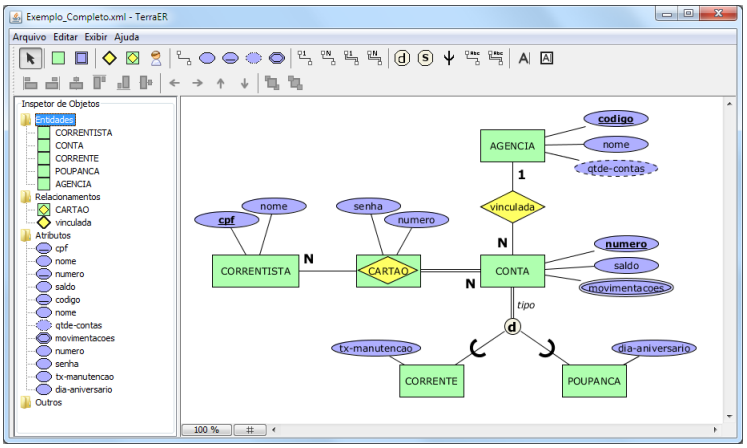


Figura 3. Modelo ER estendido na ferramenta TerraER

3. Validação de Modelos ER

Esta seção descreve a solução proposta para validação de Modelos ER. As Seções 3.1 e 3.2 formalizam cada elemento ER e as suas construções válidas, respectivamente. A Seção 3.3 descreve a integração da validação proposta na ferramenta `TerraER`. Por fim, a Seção 3.4 avalia a solução proposta sob a perspectiva educacional.

3.1. Elementos

Os elementos se dividem em duas categorias: *figuras* e *conexões*. Por um lado, as figuras são partes do modelo que existem por si só, como entidades, atributos, relacionamentos e o tipo de herança. Por outro lado, as conexões são elementos que necessitam de pelo menos duas figuras para existir, como conexões de atributos, conexões de participação e cardinalidade dos relacionamentos e conexões relacionadas à herança. A Tabela 1 ilustra os elementos ER usados na validação e suas categorias.

Tabela 1. Elementos ER

Figuras		*Conexões*	
Entidade	Entidade Fraca	Conexão de Atributo	Opcional ‘1 para’
Entidade Relacionamento	Relacionamento	Obrigatória ‘1 para’	Opcional ‘n para’
Relacionamento Fraco	Atributo	Obrigatória ‘n para’	Geral opcional
Atributo Chave	Atributo Chave Parcial	Geral obrigatória ‘n para’	Generalização
Atributo Derivado	Atributo Multivalorado		
Disjunção	Sobreposição		
União			

3.2. Formalização das Conexões Válidas

Claramente, alunos não podem realizar qualquer tipo de conexão entre quaisquer duas figuras. A Figura 4 ilustra um grafo onde os vértices são figuras ou grupo de figuras e as arestas são rotuladas com as conexões válidas entre elas.

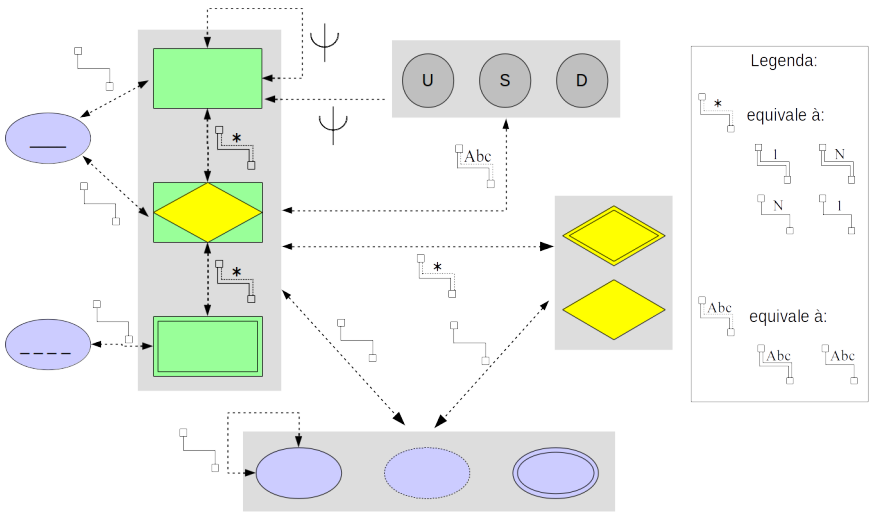


Figura 4. Conexões válidas

Conforme supramencionado, figuras foram devidamente agrupadas para facilitar a visualização. Um grupo de figuras é representado por um retângulo cinza em volta

das figuras e significa que todas elas compartilham as mesmas conexões válidas. Por exemplo, a conexão de atributo pode ser estabelecida entre (i) qualquer elemento do grupo composto por entidade, entidade fraca e entidade relacionamento e (ii) qualquer elemento do grupo composto por atributo, atributo derivado e atributo multivalorado.

No entanto, um elemento pertencente a um grupo pode ter conexões válidas que não são compartilhadas com os demais elementos do grupo. Por exemplo, a entidade e entidade relacionamento permitem conexão com atributo chave, porém entidade fraca – mesmo pertencendo ao mesmo grupo de figuras – não permite.

De forma similar, conexões também foram agrupadas de forma a facilitar a visualização em quatro grupos principais:

- Conexão de atributo *(grupo unitário)*;
- Conexão de relação *(grupo composto pela conexão opcional e obrigatória '1 para', e conexão opcional e obrigatória 'N para')*;
- Conexão geral *(grupo composto pela conexão geral opcional e obrigatória)*; e
- Conexão de generalização *(grupo unitário)*.

Por exemplo, o grupo composto por entidade, entidade fraca e entidade relacionamento pode estabelecer qualquer conexão de relação (conexão opcional ou obrigatória '1 para', ou conexão opcional ou obrigatória 'N para') com o grupo composto por relacionamento e relacionamento fraco.

3.3. Projeto e Implementação

Durante a modelagem na ferramenta `TerraER`, o aluno seleciona as figuras (ver lado esquerdo da Tabela 1) desejadas e as posiciona no modelo. Consequentemente, o aluno as conecta usando qualquer conexões entre figuras (ver lado direito da Tabela 1). O módulo de validação proposto atua exatamente no momento em que o aluno conecta duas figuras. Basicamente, o método `validateLineConnection` da classe `ModelValidation`[6] é o responsável por verificar se os elementos podem ser conectados.

Caso a conexão não seja permitida entre tais figuras, `TerraER` destacará a conexão na cor vermelha no intuito de alertar o aluno sobre o erro encontrado. Mais importante, o aluno não só é notificado do problema como pode solicitar sugestão de como corrigi-lo. Uma tela de recomendações, como a ilustrada na Figura 5, aponta quais as possíveis conexões entre as tais duas figuras e também quais as possíveis figuras que podem ser conectadas usando tal conexão.

Primeiramente, é exibida uma lista de conexões válidas entre os dois elementos envolvidos no erro encontrado. Por exemplo, conforme ilustrado na Figura 5, uma entidade está sendo erroneamente conectada à uma outra entidade por meio de uma conexão de relação. Nesse caso, é exibido ao aluno que a única conexão válida entre duas entidades é a conexão de generalização.

[6]Essa classe está publicamente disponível em: https://github.com/rterrabh/terraer/blob/master/terraer_project/src/main/java/org/jhotdraw/draw/ModelValidation.java

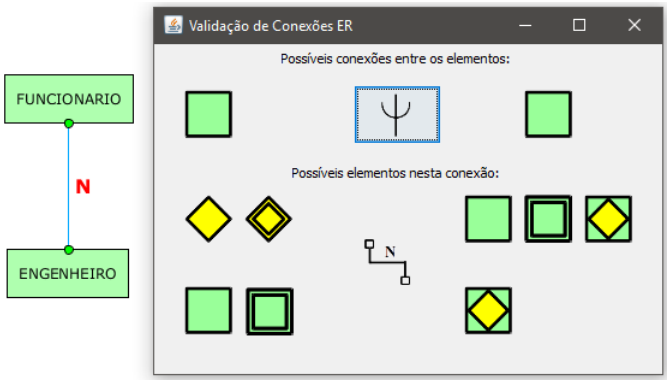


Figura 5. Conexão inválida em vermelho e tela de recomendações no TerraER

Segundamente, é exibida uma lista de elementos que podem ser conectados usando a conexão em que há o erro (conexão de relação, nesse exemplo). Por exemplo, conforme ilustrado na Figura 5, a conexão de relação pode ocorrer (i) relacionamento ou relacionamento fraco com entidade, entidade fraca ou entidade relacionamento, e (ii) entidade ou entidade forte com entidade relacionamento.

Como trabalho futuro, pretende-se integrar às recomendações explicações de uso de cada um dos elementos ER. O objetivo é trazer informações que podem – de forma complementar – contribuir para o aprendizado do aluno.

3.4. Avaliação sob a Perspectiva Educacional

Esta seção avalia a solução proposta sob a perspectiva educacional. Três elementos positivos puderam ser observados: (i) a liberdade do aluno de errar e notificá-lo sobre o erro, (ii) a facilidade de correção pelo professor e (iii) a influência da estratégia de tentativa-e-erro.

Permitir o aluno errar e fornecer feedback: Sabe-se que sinapses são geradas ao se cometer erros [10]. Mesmo que não se tenha ciência do erro cometido, sinais ERN (Negatividade Relacionada ao Erro) são gerados [9]. Porém, quando se tem ciência do erro por meio de algum *feedback*, sinais ERP (Positividade Relacionada ao Erro) são gerados pouco depois (200 a 500 milissegundos) [6, 11].

Estudos afirmam que a presença desse sinal de ERP está relacionada ao aprendizado pós-erro, levando a um melhor desempenho e precisão em atividades futuras [14, 8]. De acordo com a teoria supracitada, a presença do ERP está relacionada tanto ao cometimento de erro quanto ao *feedback* pós-erro. Logo, é seguro afirmar que permitir o erro do aluno e notificá-lo potencialmente melhora o seu aprendizado na tarefa realizada.

O fato de o *feedback* proporcionar maior presença do ERP [6, 11] explica ser amplamente utilizado na educação e no processo de aprendizado, especialmente o *feedback* corretivo [4, 15]. Tipicamente envolve o estudante recebendo um *feedback* formal ou informal sobre a execução de uma ou mais tarefas. A solução proposta envolve, no momento de cada nova conexão construção do modelo, uma verificação dessa construção. Caso houver algum erro de construção, o aluno é notificado através de um *feedback* corretivo. Essa notificação pode ser considerada um *feedback* imediato, isso por trazer à ciência do aluno o erro no momento em que este é cometido, o que pode

aprimorar ainda mais o aprendizado [4]. Ao permitir o erro do aluno e, imediatamente em seguida, informá-lo do mesmo, é instigado o aprendizado a fim de evitar tal erro em atividades futuras.

Menor preocupação com detalhes do modelo: Um dos objetivos do módulo de validação proposto é otimizar o processo educacional. Notificar e corrigir esses erros de construção facilitam a construção do modelo como um todo, uma vez que permite que o aluno foque na tarefa de modelagem conceitual, sem preocupações com formalizações do modelo. Isso ocorre sem sacrificar o aprendizado das formalizações do modelo, uma vez que o sistema de *feedback* provê exatamente isso. Ainda, essa facilitação também é transferida ao professor que poderá corrigir atividades focado exclusivamente na modelagem conceitual. Enfim, o fato de não ter que verificar o modelo fomenta uma maior eficiência no processo de educação.

Tentativa-e-erro: O módulo de validação pode acabar induzindo alunos a abusar da tentativa-e-erro. Sob a perspectiva educacional, não é interessante que o aluno tente diversas conexões até que alguma seja válida. O módulo de validação proposto ajuda a minimizar esse problema, uma vez que permite ao aluno (mas o alerta de) estabelecer conexões inválidas. Ao alertar que são inválidas, o módulo de validação apresenta as conexões válidas. A observação da lista de conexões válidas logo após o erro instiga o aluno a refletir sobre a correta de acordo com a modelagem do problema. Mas é importante ressaltar que o problema – mesmo minimizado – pode ainda ocorrer já que o aluno pode simplesmente estabelecer conexões inválidas e aceitar as sugestões sem prévia análise.

4. Considerações Finais

Devido à sua simplicidade em representar dados e relações, o modelo Entidade-Relacionamento (ER) é largamente adotado no ensino de modelagem de dados conceitual. No entanto, durante o processo de aprendizagem, muitos dos erros cometidos pelos alunos não estão relacionados à lógica do modelo, mas sim, às suas construções. Este artigo reivindica que alunos devem focar na tarefa de modelagem conceitual, sem preocupações com formalizações do modelo.

Diante disso, este artigo propôs uma validação de modelos ER integrada à uma ferramenta acadêmica de modelagem, denominada `TerraER`. Esse módulo de validação é capaz de verificar conexões logo que são adicionadas ao modelo e – caso inválidas – não só notificar o aluno como sugerir as conexões válidas por meio de uma tela de recomendação que aponta quais as possíveis conexões entre os elementos e ainda quais os possíveis elementos que podem ter tal conexão estabelecida.

Como contribuição, a o módulo de validação proposto neste artigo visa tornar o processo de aprendizado mais rápido e interativo para o aluno, além de reduzir o esforço de correção pelo professor. Isso se justifica, primeiramente, pelo fato de o módulo de validação permitir o erro do aluno e imediatamente o notificar; cenário o qual instiga o aprendizado e evita a recorrência do mesmo erro. Em segundo lugar, o aluno pode se preocupar exclusivamente na tarefa de modelagem conceitual, sem preocupações com formalizações do modelo. Da mesma forma, professores, no processo de correção, podem ter a atenção voltada exclusivamente à modelagem, o que fomenta maior eficiência do processo educacional.

Como trabalho futuro, pretende-se auditar o módulo de validação para que o professor possa ter acesso aos erros cometidos pelos alunos e a frequência dos mesmos. Isso viabilizará a compreensão das dificuldades reais do aluno e consequentemente trará benefícios ao processo educacional. Pretende-se também avaliar o módulo de validação proposto em relação à sua efetividade e conduzir uma comparação com outros trabalhos.

Agradecimentos: Este trabalho foi apoiado pela FAPEMIG, CAPES e CNPq.

Referências

[1] S. A. Korth H. F. and S. Sudarshan. *Sistema de Banco de Dados*. Elsevier, 5th edition, 2006.

[2] P. P.-S. Chen. The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[3] C. J. Date. *Introdução a Sistemas de Banco de Dados*. Elsevier, 8th edition, 2004.

[4] R. Ellis. Corrective feedback and teacher development. *L2 Journal*, 1(1):3–18, 2009.

[5] R. Elmasri and S. B. Navathe. *Sistemas de Banco de Dados*. Pearson Addison Wesley, 6th edition, 2011.

[6] M. Falkenstein, J. Hoormann, S. Christ, and J. Hohnsbein. ERP components on reaction errors and their functional significance: a tutorial. *Biological Psychology*, 51(2):87–107, 2000.

[7] H. Garcia-Molina. *Database Systems: The Complete Book*. Pearson Prentice Hall, 2nd edition, 2009.

[8] R. Hester, N. Barre, J. B. Mattingley, J. J. Foxe, and H. Garavan. Avoiding another mistake: error and posterror neural activity associated with adaptive posterror behavior change. *Cognitive, Affective, & Behavioral Neuroscience*, 7(4):317–326, 2007.

[9] S. J. Luck. An introduction to the event-related potential technique. cognitive neuroscience. *The Quarterly Review of Biology*, 81(2):201–202, 2006.

[10] J. Moser, H. Schroder, C. Heeter, T. Moran, and Y.-H. Lee. Mind your errors. *Psychological Science*, 22(12):1484–9, 2011.

[11] S. Nieuwenhuis, K. R. Ridderinkhof, J. Blom, G. P. Band, and A. Kok. Error-related brain potentials are differentially related to awareness of response errors: Evidence from an antisaccade task. *Psychophysiology*, 38(5):752–760, 2001.

[12] H. Rocha and R. Terra. Uma ferramenta voltada ao ensino do modelo de entidade-relacionamento. *VI Escola Regional de Banco de Dados (ERBD)*, pages 1–4, 2010.

[13] H. Rocha and R. Terra. TerraER - an academic tool for er modeling. *Methods and Tools*, 1(3):38–41, 2013.

[14] H. S. Schroder, M. E. Fisher, Y. Lin, S. L. Lo, J. H. Danovitch, and J. S. Moser. Neural evidence for enhanced attention to mistakes among school-aged children with a growth mindset. *Developmental Cognitive Neuroscience*, 24(1):42–50, 2017.

[15] G. Steuer, G. Rosentritt-Brunn, and M. Dresel. Dealing with errors in mathematics classrooms: Structure and relevance of perceived error climate. *Contemporary Educational Psychology*, 38(3):196–210, 2013.

[16] T. J. Teorey, D. Yang, and J. P. Fry. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18(2):197–222, 1986.

Identificação Automática de Estilos de Aprendizagem: Uma Revisão Sistemática da Literatura

Edilaine Santiago de Oliveira[1], **Gilvandenys Leites Sales**[1], **Pryscilla de Sousa Pereira**[1], **Ramires do Nascimento Moreira**[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará -
Programa de Pós-Graduação em Ciência da Computação (PPGCC)
Av. Treze de Maio, 2081 – Benfica – 60.040-215 – Fortaleza – CE – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Ceará -
Departamento de Engenharia de Telecomunicações
Av. Treze de Maio, 2081 – Benfica – 60.040-215 – Fortaleza – CE – Brasil

{edilaine.santiago9,pry.spereira,denyssales,ramires.nas}@gmail.com

Abstract. *The spread and evolution of technology have revolutionized the way education is approached. The classification of learning styles within Virtual Learning Environments (VLE's) can help in the use of new teaching strategies based on the student profile, besides contributing to the personalization of these virtual environments. This research consists of a Systematic Review of Literature (SRL) of relevant works in the last four years that adopt automatic approaches, such as data mining and machine learning, in order to classify student's learning styles. In all, 12 papers were selected, considered significant for this research, for a more careful analysis. The results show that different techniques are applied, as well as different models of learning styles available in the literature.*

Resumo. *A disseminação e a evolução da tecnologia têm revolucionado a maneira de como a educação é abordada. A classificação dos estilos de aprendizagem dentro de Ambientes Virtuais de Aprendizagem (AVA's) pode auxiliar no uso de novas estratégias de ensino baseadas no perfil do aluno, além de contribuir na personalização desses ambientes virtuais. Esta pesquisa consiste em uma Revisão Sistemática da Literatura (RSL) de trabalhos relevantes nos últimos quatro anos que adotam abordagens automáticas, como mineração de dados e aprendizado de máquina, para classificação dos estilos de aprendizagem dos alunos. Ao todo foram selecionados 12 artigos, considerados significativos para esta pesquisa, para realização de uma análise mais criteriosa. Os resultados mostram que diferentes técnicas são aplicadas, assim como diferentes modelos de estilos de aprendizagem disponíveis na literatura.*

1. Introdução

A disseminação e a evolução da tecnologia, que está cada vez mais presente em nosso meio, revolucionam a maneira como a educação é abordada. Essa revolução mudou a forma de aprender, tornando os indivíduos mais ativos na aprendizagem, buscando mais interatividade e participação [Macedo 2010]. A modalidade de ensino de Educação a

Distância (EaD) é uma prova de como a tecnologia afeta na educação, visto que a partir desta é possível atingir um maior número de pessoas interconectadas por meio da internet. Gradativamente as instituições de ensino têm adotado o uso de cursos a distância, é o que mostra a pesquisa elaborada pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira [INEP 2016].

Estes cursos utilizam como principal recurso os Ambientes Virtuais de Aprendizagem (AVA's), também conhecidos como *Learning Management System* (LMS). Estes ambientes podem ser compreendidos como espaços pedagógicos colaborativos dispostos na Internet para a formação de comunidades de aprendizes [Sales 2010]. Seja dentro ou fora desses ambientes uma difícil tarefa encontrada pelo professor é conseguir adaptar os métodos de aprendizagem para diferentes tipos de alunos [Felder and Brent 2005]. A falta de encontros presenciais provavelmente irá dificultar ainda mais a percepção de alguns aspectos comportamentais dos alunos. Outro obstáculo relacionado a estes ambientes virtuais é que eles tendem a ser centrados no curso, ao invés de serem focados no aluno [Graf and List 2005].

Esta abordagem pode não favorecer no desempenho dos alunos, onde cada um tem seus pontos fortes e fracos, interesses, níveis de motivação e abordagens diferentes para estudar [Henze et al. 2004]. A diversidade de tipos de personalidades, assim como os impactos dos estados cognitivos na compreensão e resolução de desafios, mostram a importância do processo de aprendizagem estar alinhado com os traços da personalidade de cada aluno [Farias et al. 2014].

Assim, a identificação dos estilos de aprendizagem dentro de um AVA pode auxiliar na construção de sistemas personalizados que mesclam os benefícios dos sistemas de aprendizagem com uma abordagem adaptada para cada estilo. Essas plataformas de *e-learning* personalizadas buscam melhorar a satisfação e a motivação dos alunos ao longo do curso [Halawa et al. 2015a].

Portanto, considerando a importância do tema proposto, decidiu-se realizar uma Revisão Sistemática da Literatura (RSL) a fim de identificar as estratégias utilizadas para classificação automática de perfis de aprendizagem, assim como detectar os modelos de estilos mais comuns. O objetivo é avaliar e interpretar os estudos relevantes nessa área e, consequentemente, levantar conteúdo que auxilie equipes responsáveis por novas modelagens de sistemas adaptáveis com base no perfil de aprendizagem do aluno.

O restante deste trabalho está organizado da seguinte maneira: na seção 2 abordamos a metodologia desta pesquisa, explicando todas as etapas seguidas no protocolo de revisão. Na seção 3 apresentamos os resultados obtidos após os critérios de inclusão e exclusão, assim como a síntese e análise dos artigos selecionados. Por fim, na seção 4 apresentamos as conclusões levantadas por intermédio desta pesquisa.

2. Metodologia

Para atingir o objetivo desta pesquisa, foram realizadas as seguintes etapas (Figura 1), conforme indicado por [Kitchenham 2004].

2.1. Planejamento

Inicialmente foram definidas duas questões de pesquisa que esta RSL tem como objetivo responder:

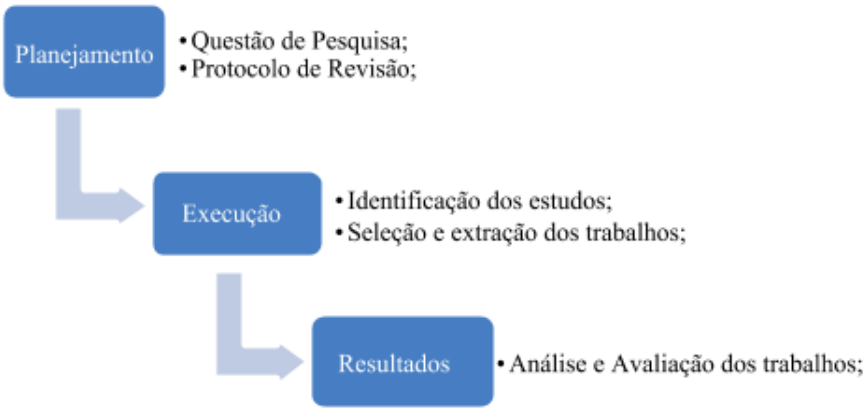


Figura 1. Etapas da metodologia

- QP1: Quais modelos de estilos de aprendizagem da literatura são mais utilizados nos estudos realizados?
- QP2: Quais são os diferentes métodos usados para identificar estilos de aprendizagem de forma automática?

O protocolo de revisão foi elaborado de forma a responder às questões de pesquisas definidas. No mês de março de 2018 foram consultadas quatro bases de arquivos científicos para a etapa de seleção dos trabalhos: *Scopus*, *Association for Computing Machinery Digital Library* (ACM DL), *Web of Science* e *IEEEXplorer*.

Estas bases foram selecionadas devido aos mecanismos de buscas online que possuem, possibilitando o uso de filtros por ano e tipo de publicação, além de abranger pesquisas relacionadas ao tema da computação. As *strings* de busca utilizadas nestas bases (Tabela 1) foram elaboradas a partir de palavras-chaves como: *data mining methods, data mining techniques, machine learning, learning styles*, assim como seus sinônimos.

Tabela 1. *Strings* de busca usadas nos repositórios

Repositório	String de Busca
Scopus	TITLE-ABS-KEY (("learning styles" OR "learning style") AND (“Data mining techniques” OR “Data mining methods” OR “data mining” OR “machine learning”))
ACM	Searched for +("learning style") +(“ Data mining techniques” “Data mining methods” “data mining” “machine learning”)
Web of Science	TS=(("learning style*") AND (“Data mining technique*” OR “Data mining method*” OR “data mining” OR “machine learning”))
IEEEXplorer	(((((("learning styles") AND "Data mining techniques") OR “Data mining methods”) OR “data mining”) OR “machine learning”)

Para que a pesquisa seja mais específica e criteriosa, somente foram aceitos trabalhos que atendessem aos critérios de inclusão e exclusão definidos a seguir (Tabela 2). Nesta revisão desejamos explorar artigos mais recentes para identificar quais técnicas de

classificação e quais modelos de estilos de aprendizagem são mais utilizados atualmente, logo é importante ressaltar que apenas foram consideradas publicações realizadas nos últimos quatro anos. Ao todo foram selecionados cinco critérios de inclusão e quatro de exclusão, utilizados na etapa de seleção.

Tabela 2. Critérios de Inclusão e Exclusão

Critério	ID	Descrição
Inclusão	I-1	Trabalhos na área de Ciências da Computação
	I-2	Trabalhos escritos em inglês
	I-3	Trabalhos publicados entre 2014 e março de 2018
	I-4	Artigos revisados por pares
	I-5	Utiliza abordagens automáticas para identificar os estilos de aprendizagem
Exclusão	E-1	Trabalhos duplicados
	E-2	Trabalhos que não contém palavras-chaves no título ou resumo
	E-3	Número de páginas menor que quatro
	E-4	Trabalhos não disponíveis integralmente na web e não acessíveis de forma gratuita

2.2. Execução

A etapa de execução realiza a identificação, seleção e extração dos trabalhos. Durante o processo de identificação, as *strings* de busca foram aplicadas nas bases de artigos científicos e o conjunto inicial foi de 566 artigos, distribuídos nos quatro repositórios (Figura 2).

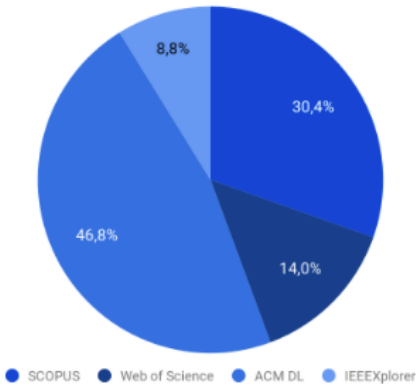


Figura 2. Artigos selecionados por repositório

A maior parte dos artigos (77,2%) foi encontrada nas bases de dados da ACM DL e *Scopus*, onde quase metade (46,8%) dos artigos vieram da ACM DL. Os outros 22,8% dos trabalhos foram encontrados nos repositórios do *IEEEXplorer* e *Web of Science*, onde somente 8,8% vieram do *IEEEXplorer*.

Na etapa de seleção ocorre a aplicação dos critérios de inclusão e exclusão. A fim de encontrar quais artigos utilizam abordagens automáticas para identificar os estilos

de aprendizagem, realizamos a leitura do título e resumo dos noventa artigos finais, onde apenas doze trabalhos foram escolhidos para uma análise completa (Figura 3). Estes trabalhos são analisados de forma mais detalhada na próxima seção a fim de responder às questões de pesquisa elaboradas.

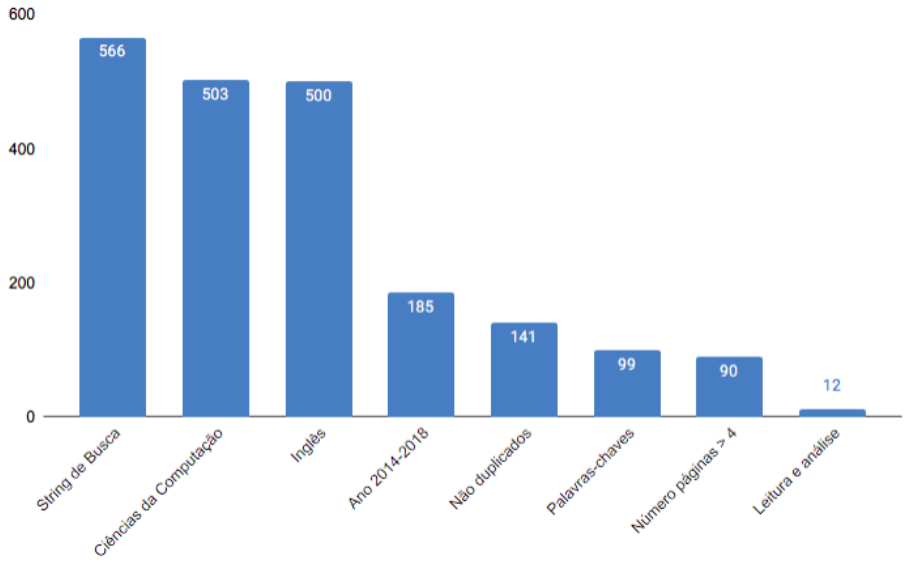


Figura 3. Aplicação dos critérios de inclusão e exclusão

Com a aplicação do primeiro critério de inclusão (somente artigos da área da computação), 63 artigos foram excluídos. Dos 503 artigos que ficaram, somente 3 não são da língua inglesa. Na aplicação do critério de inclusão 3 (trabalhos publicados entre 2014 e março de 2018), 185 artigos foram extraídos para aplicação dos outros critérios.

Após retirar os trabalhos duplicados, trabalhos que não contém palavras-chaves no resumo ou título e trabalhos com número de páginas menor que quatro, foram selecionados 12 artigos para a etapa de análise e avaliação. Nesta etapa levamos em consideração também a qualidade de cada trabalho, assim como a qualidade da conferência/periódico onde os mesmos foram publicados.

Estes trabalhos foram selecionados para serem lidos por completo, de modo a avaliar se realmente descrevem a forma, com alto nível de detalhe, de como é feita a classificação dos estilos de aprendizagem nos ambientes virtuais de aprendizagem.

3. Discussão e Análise dos Resultados

Esta seção apresenta os resultados produzidos pela extração de dados dos estudos primários de acordo com o processo descrito na Seção 2. Para fornecer uma maneira mais simples de visualizar os resultados, estes foram separados em dois grupos: informações gerais e respostas das questões de pesquisa. Em primeiro lugar, apresentamos uma visão geral dos estudos relevantes, em seguida as respostas às questões de pesquisa são apresentadas.

3.1. Informações Gerais dos Resultados

Os documentos filtrados foram lidos e, em seguida, recuperamos muitas outras informações. Depois disso, armazenamos o título, ano de publicação, local de publicação

(periódico ou conferência), nome e localização do autor. Inicialmente consideramos algumas informações gerais, como, distribuição de artigos por ano (Tabela 3).

Tabela 3. Distribuição de artigos por ano

Ano	Artigo	Publicação
2014	[Efrati et al. 2014]	Conferência
	[Mohamed et al. 2014]	Conferência
2015	[Halawa et al. 2015a]	Conferência
	[Hassan and Hegazy 2015]	Conferência
	[Paireekreng and Prexawanprasut 2015]	Conferência
	[Sweta and Lal 2015]	Conferência
2016	[Adel et al. 2016]	Conferência
	[Hung et al. 2016]	Periódico
	[Liyanage et al. 2016]	Periódico
	[Popescu et al. 2016]	Conferência
2017	[Binh and Duy 2017]	Conferência
	[Zhong et al. 2017]	Periódico

Os seguintes padrões foram observados no resultado: o número de publicações varia de 2 a 4 artigos; 2015 e 2016 são os anos mais produtivos com 4 publicações cada. A maioria dos estudos (75%) foi publicada em anais de congressos, enquanto os trabalhos [Hung et al. 2016, Liyanage et al. 2016, Zhong et al. 2017], foram publicados em periódicos.

Já as informações sobre em quais países foram realizadas as pesquisas filtradas, são exibidas na Tabela 4. Uma tendência que se observa nos resultados é que, com exceção do Egito, todos os trabalhos são de países diferentes, o que pode apontar para existência de particularidades locais para o tema de perfis de aprendizagem.

Tabela 4. Países das pesquisas selecionadas

Localização Autor	Artigo
China	[Zhong et al. 2017]
Itália	[Efrati et al. 2014]
Egito	[Hassan and Hegazy 2015], [Halawa et al. 2015a]
Tailândia	[Paireekreng and Prexawanprasut 2015]
Malásia	[Mohamed et al. 2014]
Japão	[Liyanage et al. 2016]
Taiwan	[Hung et al. 2016]
Romênia e Estados Unidos	[Popescu et al. 2016]
Vietnã	[Binh and Duy 2017]
Reino Unido	[Adel et al. 2016]
Índia	[Sweta and Lal 2015]

3.2. Respostas para as Questões de Pesquisa

Nesta seção nós apresentamos os resultados agrupados para responder as duas questões de pesquisas levantadas no início deste documento. O processo foi iniciado com a leitura dos doze artigos selecionados. Inicialmente foi feita a identificação dos objetivos e das metodologias de cada um dos artigos, selecionando informações relevantes para responder cada uma das questões.

QP1: Quais modelos de estilos de aprendizagem da literatura são mais utilizados nos estudos realizados?

Diversos tipos de modelos de estilos de aprendizagem foram abordados nos trabalhos selecionados (Tabela 5). O modelo de Felder e Silverman foi o mais utilizado nas pesquisas, totalizando 66% dos trabalhos. Comparando-se o número de citações deste modelo com os demais, através do *Google Scholar*, também nota-se uma maior tendência em seu uso.

Tabela 5. Modelos de Estilos de Aprendizagem

Modelo	Artigo
Felder-Silverman	[Adel et al. 2016], [Binh and Duy 2017], [Hung et al. 2016], [Liyanage et al. 2016], [Mohamed et al. 2014], [Paireekreng and Prexawanprasut 2015], [Popescu et al. 2016],[Sweta and Lal 2015]
Grasha-Riechmann	[Efrati et al. 2014]
Honey-Mumford	[Zhong et al. 2017]
Myers-Briggs	[Halawa et al. 2015a]
VARK	[Hassan and Hegazy 2015]

No modelo de Felder e Silverman (FSLSM) os estilos de aprendizagem são definidos como as preferências dos alunos na forma de perceber, captar, organizar, processar e compreender a informação [Felder and Silverman 1988]. Neste modelo os alunos são categorizados em quatro dimensões e cada dimensão está dividida em dois estilos (ativo/reflexivo, sensorial/intuitivo, visual/verbal e sequencial/global).

De acordo com o modelo FSLSM, alunos ativos preferem trabalhos em equipe, experimentações e discussões. Já o tipo reflexivo prefere trabalhos individuais, conceito e teorias. Na dimensão de percepção, o estilo sensorial prefere obter informações por meio de dados experimentais e sentidos (vendo, ouvindo, tocando), enquanto que o perfil intuitivo prefere abstrações e definições.

Com relação à forma que os indivíduos captam as informações no modelo FSLSM, perfis visuais preferem captar a informação por meio de imagens, gráficos, vídeos, enquanto perfis verbais preferem textos e áudios. Por último temos a dimensão de organização, que indica como o indivíduo prefere progredir, podendo ser em sequência, onde tendem a progredir de forma lógica ou globalmente, onde preferem ter a visão geral do todo, aprendendo em saltos e com mais liberdade. O uso frequente desse modelo se da ao fato dele ter sido baseado em extensa experimentação, que validou as dimensões em uma população de estudantes de engenharia [Dorça 2012].

Dos oito trabalhos que abordam o modelo de Felder e Silverman, seis desses trabalhos [Paireekreng and Prexawanprasut 2015, Liyanage et al. 2016, Hung et al. 2016, Popescu et al. 2016, Binh and Duy 2017, Adel et al. 2016], usam o questionário *Index Learning System* (ILS), elaborado por Felder e Soloman [Felder and Soloman 1997] para identificar os estilos de aprendizagem dos alunos. Este questionário contém 44 questões objetivas, onde para cada dimensão existem 11 questões equivalentes, cada uma com duas opções de escolha. Os autores fazem uso das respostas do questionário para comparar com a classificação feita pelas abordagens automáticas que utilizam os registros das interações dos alunos com o ambiente virtual.

A pesquisa de [Efrati et al. 2014] sugere o uso do modelo Grasha-Riechmann, aplicado como referência para classificar os estilos. Este modelo possui apenas três dimensões: intra-subjetividade/intersubjetividade, competitivo/colaborativo, independente/dependente, onde cada uma possui uma escala de valores, podendo ser do tipo baixo, médio ou alto. Grasha e Riechmann revisaram e avaliaram os estilos de aprendizagem de estudantes universitários por meio de uma perspectiva social, a fim de identificar diferentes abordagens no ambiente da sala de aula [Grasha 1972].

Outro modelo utilizado foi o de Honey e Mumford, no trabalho de [Zhong et al. 2017]. Este modelo possui quatro estilos: reflexivo, teórico, pragmático e ativo. Ele é derivado diretamente da teoria de [Kolb 1984]. Já o trabalho de [Halawa et al. 2015b] aborda o modelo de Myers-Briggs, que utiliza o *Myers-Briggs Type Indicator* (MBTI), onde este avalia tipos de personalidade e preferências por meio de quatro aspectos da personalidade: extroversão/introversão, detecção/intuição, pensamento/sentimento, julgando/percebendo. A combinação dessas diferentes preferências resulta na possibilidade de 16 diferentes tipos de estilos de aprendizagem, assim como no modelo de Felder e Silverman.

Por último temos o modelo de VARK, acrônimo para *Visual, Auditory, Read/write, e Kinesthetic*, utilizado no trabalho de [Hassan and Hegazy 2015]. Esse modelo também utiliza um questionário, onde este é usado para categorizar quatro modalidades de preferências sensoriais: visual, auditivo, ler/escrever e cinestésica. Nesta última modalidade o aluno adquire conhecimento através de experiência simulada ou real. Logo, nesta classificação, o aluno pode ter preferência em um único tipo de perfil ou ser do tipo multimodal, onde terá preferência em mais de um estilo, totalizando até 15 possibilidades de estilos diferentes.

QP2: Quais são os diferentes métodos usados para identificar estilos de aprendizagem de forma automática?

Inicialmente identificamos e apresentamos a distribuição dos tipos de técnicas usadas nos trabalhos referentes à identificação dos estilos de aprendizagem (Figura 4). A maioria das pesquisas (71,4%) implementa a técnica de classificação. Esse método é um tipo de aprendizado de máquina supervisionado [Duda et al. 2000].

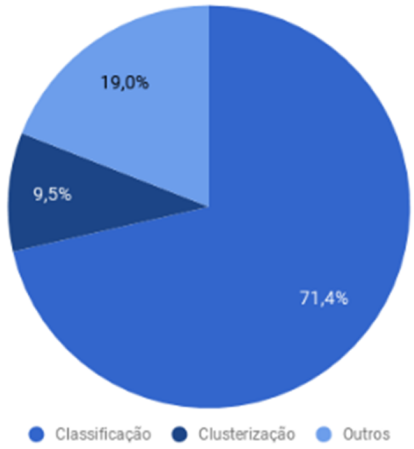


Figura 4. Técnicas usadas para identificação dos estilos de aprendizagem

Ao todo, vinte algoritmos foram utilizados nesses trabalhos (Tabela 6). De todos esses, os algoritmos de Redes Bayesianas, Árvores de Decisão, *Naive Bayes* e Redes Neurais Artificiais (RNA), foram os mais referenciados nas pesquisas, onde as Redes Bayesianas foram aplicadas em cinco trabalhos e os demais em quatro. Apesar dessas técnicas não serem tão recentes, ainda são aplicáveis na classificação do perfil dos estudantes na atualidade.

Já os outros algoritmos foram usados em um ou dois trabalhos apenas. Dos doze artigos analisados, cinco deles [Efrati et al. 2014, Popescu et al. 2016, Hung et al. 2016, Binh and Duy 2017, Sweta and Lal 2015], utilizaram somente um algoritmo, enquanto os demais utilizaram no mínimo dois [Zhong et al. 2017] e no máximo dez [Halawa et al. 2015b].

Tabela 6. Algoritmos utilizados

Algoritmo	Artigo	Acurácia
Bayesian Network	[Hassan and Hegazy 2015], [Liyanage et al. 2016], [Halawa et al. 2015a], [Adel et al. 2016], [Sweta and Lal 2015]	89,9%, 68,1%, 88,5%, 83,3%, -
Bijective Soft Set	[Mohamed et al. 2014]	98,4%
Decision Table	[Halawa et al. 2015a]	85,4%
Decision Tree/J48 Graft	[Paireekreng and Prexawanprasut 2015], [Liyanage et al. 2016], [Halawa et al. 2015a], [Adel et al. 2016]	62,7%, 79,1%, 92,1%, 83,3%
Discriminant Function Analysis (DFA)	[Popescu et al. 2016]	62,6%
Expectation Maximization (EM)	[Efrati et al. 2014]	-
FURIA	[Mohamed et al. 2014]	97,2%
Hyper Pipes	[Hassan and Hegazy 2015]	92,3%
JRIP	[Halawa et al. 2015a], [Hassan and Hegazy 2015]	63,5%, 90,3%
K-means	[Zhong et al. 2017], [Hung et al. 2016]	56,5%, -
KNN/IBK	[Halawa et al. 2015a], [Adel et al. 2016], [Hassan and Hegazy 2015]	90,6%, 62,9%, 97%
KStar	[Halawa et al. 2015a]	48,9%
Naive Bayes	[Halawa et al. 2015a], [Paireekreng and Prexawanprasut 2015], [Liyanage et al. 2016], [Adel et al. 2016]	68,2%, 42,5%, 65,2%, 79,6%
OneR	[Halawa et al. 2015a]	97,4%
RandomTree/Random Forests	[Liyanage et al. 2016], [Halawa et al. 2015a], [Adel et al. 2016]	78,1%, 93,2%, 81,6%
Rede Neural/MLP	[Zhong et al. 2017], [Paireekreng and Prexawanprasut 2015], [Binh and Duy 2017], [Adel et al. 2016]	100%, 88,3%, 80,6%, 69,9%
Rough Set	[Mohamed et al. 2014]	96,3%
Simple CART	[Adel et al. 2016]	83,3%
Simple Logistic	[Hassan and Hegazy 2015]	93,8%
SVM	[Zhong et al. 2017], [Paireekreng and Prexawanprasut 2015]	99,7%, 25,5%

Os algoritmos que obtiveram as melhores acurácias foram: RNA (100%), SVM (99,7%) [Zhong et al. 2017], Bijective Soft Set (98,4%) [Mohamed et al. 2014], OneR (97,4%) [Halawa et al. 2015a] e IBK (97%) [Hassan and Hegazy 2015].

Metade dos trabalhos analisados [Efrati et al. 2014, Hassan and Hegazy 2015, Liyanage et al. 2016, Halawa et al. 2015a, Adel et al. 2016, Sweta and Lal 2015], utilizou a ferramenta *Waikato Environment for Knowledge Analysis* (WEKA) para aplicação e análise dos algoritmos. Essa ferramenta possui uma coleção de algoritmos de aprendizado de máquina para mineração de dados, além de ser uma plataforma de código aberto baseada em Java desenvolvida na Universidade de Waikato, Nova Zelândia [Liyanage et al. 2016]. Desses trabalhos que fizeram uso da ferramenta WEKA, cinco deles usaram o ambiente virtual *Modular Object Oriented Developmental Learning Environment* (Moodle), totalizando 42% dos doze trabalhos pesquisados.

Já os dados mais utilizados para fazer a identificação dos estilos de aprendizagem, foram dados coletados a partir dos registros gerados pelas interações dos alunos dentro dos AVAs [Efrati et al. 2014, Zhong et al. 2017, Hassan and Hegazy 2015, Liyanage et al. 2016, Hung et al. 2016, Halawa et al. 2015a, Sweta and Lal 2015]. Dois trabalhos [Popescu et al. 2016, Adel et al. 2016] usaram análise de dados textuais, um utilizou somente dados do questionário ILS [Binh and Duy 2017], outro usou dados pessoais [Paireekreng and Prexawanprasut 2015] e o último usou dados simulados das interações no AVA [Mohamed et al. 2014].

Abaixo apresentamos o resumo de todas as características extraídas dos trabalhos (tabela 7) .

Tabela 7. Síntese dos trabalhos selecionados

Autor	Modelo	Algoritmo	Dados Utilizados	Qtde. Alunos	Ambiente Virtual	Ferramenta
[Adel et al. 2016]	Felder-Silverman	Naive Bayes, Bayes Network, J48, Simple CART, Random Forest, IBK e RBF	ILS e Registros do sistema de tutoria inteligente	75	Oscar	WEKA
[Binh and Duy 2017]	Felder-Silverman	MLP	ILS	316	-	Phyton
[Efrati et al. 2014]	Grasha-Riechmann	Expectation Maximization (EM)	Registros do ambiente virtual	-	Moodle	WEKA
[Halawa et al. 2015a]	Myers-Briggs	NaiveBayes, BayesNet, Kstar, Random forest, J48, OneR, JRIP, KNN/IBK, RandomTree e Decision Table.	Registros do ambiente virtual e Redes Sociais	240	Moodle	WEKA
[Hassan and Hegazy 2015]	VARK	Simple Logistic, IBK, Bayes Net, J48 Graft, Hyper Pipes, JRIP	Registros do ambiente virtual e Redes Sociais	131	Moodle	WEKA
[Hung et al. 2016]	Felder-Silverman	K-means	ILS e Registros do ambiente virtual (jogo)	67	-	-
[Liyanage et al. 2016]	Felder-Silverman	J48, Bayesian Network, Naive Bayes e Random Forests.	ILS e Registros do ambiente virtual	80	Moodle	WEKA
[Mohamed et al. 2014]	Felder-Silverman	Bijective Soft Set, FURIA e Rough Set	Dados simulados de registros do ambiente virtual	-	-	-
[Paireekreng and Prexawanprasut 2015]	Felder-Silverman	Decision Tree, Naive Bayes, RNA e SVM	ILS e Dados pessoais dos alunos	400	-	-
[Popescu et al. 2016]	Felder-Silverman	Discriminant Function Analysis (DFA)	ILS e Análise de texto	66	eMUSE	Phyton
[Sweta and Lal 2015]	Felder-Silverman	Bayesian Network	Registros do ambiente virtual	40	Moodle	WEKA
[Zhong et al. 2017]	Honey e Mumford	K-means, SVM, Rede Neural Artificial	ILS e Registros do ambiente virtual	183	MOOC	-

4. Conclusão

Neste artigo apresentamos uma revisão sistemática da literatura para analisar os trabalhos que usam diferentes abordagens automáticas na identificação dos estilos de aprendizagem de alunos. Procuramos trabalhos de 2014 a 2018 e definimos questões de pesquisa juntamente com um protocolo de revisão para orientar nossa busca feita em quatro bases de artigos. Pretende-se com esse estudo ampliar o estado da arte, contribuindo por evidenciar trabalhos de pesquisa voltados para identificação desses estilos.

As principais conclusões são de que a maioria dos trabalhos são de conferências, o modelo Felder e Silverman é o modelo mais utilizado, as técnicas mais utilizadas são do tipo classificação, onde as mais referenciadas foram: RNA, Árvores de decisão, Redes Bayesianas e Naive Bayes. Outro ponto que merece destaque nos resultados encontrados é a predominância de dois tipos de coleta de dados: as coletas baseadas em questionários e as coletas baseadas em registros do ambiente virtual de aprendizagem.

Com isso, além das técnicas/ferramentas utilizadas presentes nos estudos selecionados, esta pesquisa mostrou os avanços no que se refere ao uso de técnicas de mineração de dados educacionais para identificação dos perfis de aprendizagem. Espera-se a partir desta revisão estimular a condução de mais pesquisas nessa área, de modo a potencializar o uso de novos algoritmos de classificação, além de estimular o desenvolvimento de ambientes virtuais personalizados com base nesses perfis.

A partir desta RSL, concluímos que identificação dos estilos de aprendizagem em AVA é uma linha de pesquisa que tem despertado o interesse da comunidade acadêmica e explorado uma diversidade de tecnologias.

Agradecimentos

Este pesquisa é apoiada pelo PPGCC/IFCE e pela Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico - FUNCAP.

Referências

Adel, N., Latham, A., and Crockett, K. A. (2016). Towards socially intelligent automated tutors: Predicting learning style dimensions from conversational dialogue. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 315–320.

Binh, H. T. and Duy, B. T. (2017). Predicting students' performance based on learning style by using artificial neural networks. In *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, pages 48–53.

Dorça, F. A. (2012). *Uma abordagem estocástica baseada em aprendizagem por reforço para modelagem automática e dinâmica de estilos de aprendizagem de estudantes em sistemas adaptativos e inteligentes para educação a distância*. Tese de doutorado, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2Nd Edition)*. Wiley-Interscience.

Efrati, V., Limongelli, C., and Sciarrone, F. (2014). A data mining approach to the analysis of students' learning styles in an e-learning community: A case study. In Stephanidis, C. and Antona, M., editors, *Universal Access in Human-Computer Interaction. Universal Access to Information and Knowledge*, pages 289–300, Cham. Springer International Publishing.

Farias, A., Gomes, T., and Cabral, G. (2014). Uma proposta metodológica para o ensino de programação baseado na relação entre perfis cognitivos, padrões pedagógicos

e autoregulação dos estudantes. *XXXIV Congresso da Sociedade Brasileira de Computação, XXII Workshop sobre Educação em Computação*.

Felder, R. M. and Brent, R. (2005). Understanding student differences. *Journal of Engineering Education*, 94(1):57–72.

Felder, R. M. and Silverman, L. K. (1988). Learning and teaching styles in engineering education. *ENGINEERING EDUCATION*.

Felder, R. M. and Soloman, B. A. (1997). Index of learning styles questionnaire.

Graf, S. and List, B. (2005). An evaluation of open source e-learning platforms stressing adaptation issues. In *Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*, pages 163–165.

Grasha, A. F. (1972). Observations on relating teaching goals to student response styles and classroom methods. *American Psychologist*, 27(2):144–147.

Halawa, M. S., Shehab, M. E., and Hamed, E. M. R. (2015a). Predicting student personality based on a data-driven model from student behavior on lms and social networks. In *2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC)*, pages 294–299.

Halawa, M. S., Shehab, M. E., and Hamed, E. M. R. (2015b). Predicting student personality based on a data-driven model from student behavior on lms and social networks. In *2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC)*, pages 294–299.

Hassan, S. and Hegazy, A. E. F. (2015). A model recommends best machine learning algorithm to classify learners based on their interactivity with moodle. In *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*, pages 49–54.

Henze, N., Dolog, P., and Nejdl, W. (2004). Reasoning and ontologies for personalized elearning in the semantic web. *Journal of Educational Technology and Society*, 7(4):82–97.

Hung, Y. H., Chang, R. I., and Lin, C. F. (2016). Hybrid learning style identification and developing adaptive problem-solving learning activities. *Computers in Human Behavior*, 55:552 – 561.

INEP (2016). *Censo da Educação Superior 2016 - Notas Estatísticas.* Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.

Kitchenham, B. (2004). *Procedures for Undertaking Systematic Reviews*.

Kolb, D. (1984). *Experiential Learning: Experience As The Source Of Learning And Development*, volume 1.

Liyanage, M. P. P., Gunawardena, K. L., and Hirakawa, M. (2016). Detecting learning styles in learning management systems using data mining. *Journal of Information Processing*, 24(4):740–749.

Macedo, C. M. S. (2010). *Diretrizes para criação de objetos de aprendizagem acessíveis.* Tese de doutorado, Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil.

Mohamed, H., Ahmad, N. B. H., and Shamsuddin, S. M. H. (2014). Bijective soft set classification of student's learning styles. In *2014 8th. Malaysian Software Engineering Conference (MySEC)*, pages 289–294.

Paireekreng, W. and Prexawanprasut, T. (2015). An integrated model for learning style classification in university students using data mining techniques. In *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pages 1–5.

Popescu, E., Dascalu, M., Becheru, A., Crossley, S., and Trausan-Matu, S. (2016). Predicting student performance and differences in learning styles based on textual complexity indices applied on blog and microblog posts: A preliminary study. In *2016 IEEE 16th International Conference on Advanced Learning Technologies (ICALT)*, pages 184–188.

Sales, G. L. (2010). *Learning Vectors: Um Modelo de Avaliação da Aprendizagem em EaD Online Aplicando Métricas Não-Lineares*. Tese de doutorado, Universidade Federal do Ceará, Fortaleza, CE, Brasil.

Sweta, S. and Lal, K. (2015). Web usages mining in automatic detection of learning style in personalized e-learning system. In Ravi, V., Panigrahi, B. K., Das, S., and Suganthan, P. N., editors, *Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015)*, pages 353–363, Cham. Springer International Publishing.

Zhong, S., Li, Y., Liu, Y., and Wang, Z. (2017). A computational investigation of learning behaviors in moocs. *Computer Applications in Engineering Education*, 25(5):693–705.

Experiência de Uso de Caixas de Ovos no Apoio ao Ensino de Vetores e Matrizes

Rita C. G. Berardi[1], Silvia A. Bim[1], Regiane Macuch[2], Leticia Fleig Dal Forno[2]

[1] Departamento Acadêmico de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brasil

[2]Centro Universitário de Maringá (UNICESUMAR)- Bolsista Produtividade ICETI
Maringá – PR - Brasil

{ritaberardi,sabim}@utfpr.edu.br,
{leticia.forno,regiane.macuch}@unicesumar.edu.br

Abstract. *The programming discipline is embedded in several Engineering curricula and several are the challenges related to computer education mainly due to the high level of abstraction that this requires. In this context, different teaching methodologies are necessary. With the intention of sharing methodologies that facilitate the teaching-learning process of Vectors and Matrices, content with great difficulty, this article reports the experience of a teaching approach aided by egg boxes. The results of this experiment show that the students had a good perception about the approach, evaluating as a positive differential for learning.*

Resumo. *A disciplina de programação está inserida em diversos currículos de Engenharias e vários são os desafios relacionados ao ensino de computação principalmente pelo elevado nível de abstração que esta exige. Neste contexto, diferentes metodologias de ensino se fazem necessárias. Com a intenção de compartilhar metodologias que facilitam o processo de ensino-aprendizagem de Vetores e Matrizes, conteúdo com grande grau de dificuldade, este artigo relata a experiência de uma abordagem de ensino auxiliada por caixas de ovos. Os resultados dessa experiência mostram que os estudantes tiveram uma boa percepção sobre a abordagem, avaliando como um diferencial positivo para o aprendizado.*

1. Introdução

O ensino de Programação na educação básica e ensino médio tem sido um assunto bastante discutido recentemente. No ensino superior, na maioria das grades curriculares dos cursos de graduação em Engenharias, o ensino de disciplinas de Algoritmos e Programação é obrigatório segundo as Diretrizes Curriculares Nacionais dos cursos de Engenharia[1]. Todas essas iniciativas de inserir Programação nos diferentes níveis de ensino, e em diversas áreas do conhecimento impulsionam a discussão sobre as metodologias de ensino. Em vários trabalhos sobre aprendizagem de programação, é

[1] http://portal.mec.gov.br/cne/arquivos/pdf/CES1362.pdf

possível encontrar exemplos relacionados à dificuldade de lidar com ambientes e linguagens de programação [Junior e França 2017] [Castro et al. 2017]. Tais dificuldades não são encontradas apenas em turmas de Engenharias, onde os índices de evasão nas disciplinas correlatas à computação são altos, segundo INEP (Instituto Nacional de Estudos e Pesquisas Educacionais, Sinopses do ensino superior)[2]. Os últimos índices publicados pelas universidades quanto à evasão de estudantes nos cursos de computação contribuem com a discussão sobre os possíveis fatores que levam a esses índices. Em outubro de 2016, a USP, uma das mais conceituadas universidades do país, divulgou que mais de 20% dos (as) estudantes desistem de seus cursos, sendo os relacionados ao Instituto de Ciências e Matemática e Computação de São Carlos dentre os cursos de maior média de evasão chegando a 48%[3] . Alguns estudos indicam que são variados os motivos, passando por base matemática fraca, falta de conhecimento prévio sobre o real objetivo do curso, currículos longos, critérios impróprios de avaliação do desempenho discente, entre outros [Cavalcante e Embiruçu 2013]. Porém, outro fator também apontado pelos estudos indica a não adequação da metodologia docente aos estudantes e uma falta de integração do estudante com os ambientes sócio ambientais universitários levam a um consequente nível de comprometimento afetado [Medeiros e Melo 2011]. Giraffa e Mora (2013) revelam em seu estudo que estudantes que alegavam a metodologia dos professores como um fator de desistência, demonstraram em testes específicos ter pouca dificuldade com lógica e algoritmo.

Apesar de a metodologia tradicional ainda prevalecer, tendo o professor como mero transmissor de informação e de soluções de problemas, a busca por metodologias que facilitem o aprendizado tem crescido [da Silva et al. 2015], [Sousa et al. 2010]. Dentre essas metodologias, questiona-se sobre o uso ou não da própria tecnologia como ferramenta de ensino, pois alguns movimentos defendem o afastamento do computador no ensino de Computação como o "*Computer Science Unplugged*" (computação "desplugada") idealizado por Bell, Witten e Fellows (2011). Essa metodologia visa o uso de atividades lúdicas, onde a aprendizagem de princípios fundamentais de computação como algoritmos e estrutura de dados acontece sem a necessidade do estudante interagir com softwares. Desta forma, "todo método ou estratégia que promova o envolvimento e a participação ativa do aluno no processo de desenvolvimento do conhecimento contribui para formar ambientes ativos de aprendizagem" [Barbosa e Moura, 2013 p. 57]. Atividades mais lúdicas podem contribuir com uma aprendizagem mais adequada a estudantes mais dinâmicos, mais ativos, como a própria proposta da metodologia ativa tende a motivar, fazendo com que estudantes façam atividades que vão além de ouvir o professor e tomar notas [Bonwell e Eison, 1991], [Barbosa e Moura, 2013]. Várias estratégias de estudo sugerem a diversificação de atividades para promover a aprendizagem mais significativa, fazendo com que os estudantes se mantenham realizando algo e pensando sobre o que estão fazendo, estimulando a habilidade de argumentação [Bonwell e Eison, 1991]; promovendo a autonomia [Berbel, 2011] e desenvolvendo a aprendizagem significativa [Rogers, 1969], [Weibell, 2011], [Moreira, 2011]. Com base nessa reflexão, o presente

[2] www.inep.gov.br

[3] http://www.jornaldocampus.usp.br/index.php/2016/10/basic-2/

artigo busca compartilhar e analisar uma experiência no ensino de Vetores e Matrizes em uma disciplina de Computação em cursos de graduação em Engenharia Elétrica, Mecânica de Controle e Automação e Licenciatura em Física. A experiência diz respeito ao ensino específico de Vetores e Matrizes, utilizando caixas de ovos como material didático no apoio à redução da abstração lógica exigida, por ser um dos conteúdos em que os estudantes declaram maior dificuldade pelo alto nível de abstração que exige.

Na Seção 2 são apresentados trabalhos correlatos que também estão inspirados em metodologias ativas no ensino de computação. Na Seção 3 a experiência é compartilhada, incluindo a descrição do perfil de estudante e como foi utilizado em sala de aula. Na Seção 4 resultados e discussões são apresentados por meio das respostas dos estudantes a um questionário de avaliação da metodologia e na Seção 5, as conclusões sobre este trabalho são delineadas.

2. Trabalhos correlatos

O uso de recursos lúdicos e jogos (digitais ou não) vem ganhando cada vez mais espaço em diversas disciplinas dos cursos da área de Computação. Estes recursos são usados para oferecer uma experiência mais prática tanto em disciplinas mais teóricas, como os exemplos desta seção, quanto para disciplinas com conteúdo mais abstrato [Nunes e Parreira Júnior 2015] [Vahldick et al. 2015].

Uma estratégia para a disciplina de Gerenciamento de Projetos foi proposta por Schoeffel e Wazlawick (2016). A atividade vivencial: *Mão na Massa* foi criada como ferramenta complementar a outras abordagens para o ensino de Gerenciamento de Projetos com foco nos conceitos do PMBOK[4]. A atividade foi realizada em duas turmas distintas de um curso de especialização, com 35 estudantes no total. Os resultados da realização da atividade indicam que houve, segundo os próprios estudantes, uma significativa contribuição para a aprendizagem, nos três níveis de aprendizagem (conhecimento, entendimento e aplicação). Outro exemplo, também para a disciplina de Gerenciamento de Projetos, é o jogo MEGA GP [Tomisaki et al. 2016], que assim como o trabalho citado anteriormente, não utiliza recursos computacionais para realização da atividade. O MEGA GP visa à fixação de conceitos do PMBOK e do CMMI-DEV[5] de maneira lúdica e motivadora. Conforme os resultados do questionário de satisfação aplicado, os estudantes ficaram satisfeitos com o uso deste recurso didático. Além disto, para a maior parte do conteúdo houve um desempenho melhor para a turma que fez uso do MEGA GP. Figueiredo e Santos (2016) apresentam um jogo de cartas de estratégia para ensinar História da Computação em diversos níveis de ensino. O jogo é composto de 60 cartas divididas em três categorias: personalidades, instituições e eventos. Além disto, cada carta pertence a uma área de conhecimento: hardware, software, teoria computacional e matemática, computação e sociedade, gestão de dados. Embora o trabalho citado não descreva a utilização do jogo em sala de aula, os autores acreditam

[4] PMI. (2013). Project Management Body of Knowledge – PMBOK Newton Square. Pennsylvania: Project Management Institute. 5 ed.

[5] SEI. (2010). CMMI for Development, version 1.3. Carnegie Mellon University, Software Engineering Institute (SEI).

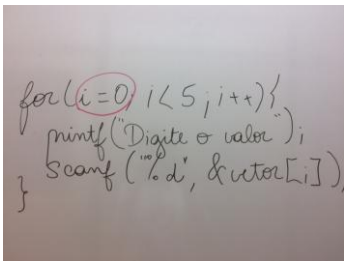
que sua dinâmica estratégica estimula o jogador na retenção dos conteúdos das cartas e a ludicidade do jogo incentiva o seu envolvimento com a disciplina.

3. Descrição da Experiência de uso das caixas de ovos em sala de aula

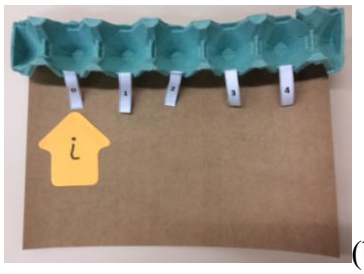
O ensino de vetores e matrizes com as caixas de ovos foi realizado em uma turma de Computação com estudantes que já haviam cursado esta disciplina, mas que haviam reprovado. Os estudantes são de cursos de graduação não relacionados à área de computação, como Licenciatura em Física, Engenharia Elétrica, Engenharia Mecânica e Engenharia de Controle e Automação. A turma era composta de 16 estudantes dos diversos períodos dos seus cursos, em média entre o 3º e 4º período. O material com as caixas de ovos foi utilizado primeiro no apoio ao ensino de Vetores. Os vetores foram representados em sequências horizontais de várias posições para os ovos. Para caracterizar o nome de cada posição do vetor, foram coladas pequenas tiras com os números, neste caso de 0 a 4. Para representar o conteúdo de cada posição do vetor foram utilizados pequenos quadrados com números inteiros escritos. Por fim, para o uso na dinâmica de execução do laço que percorre o vetor, foi utilizado um post-it no formato de seta, com o nome da variável `i` escrita.

A aula sobre vetores foi primeiramente ministrada com a metodologia expositiva com uso de quadro e slides. Após fazerem exercícios, a principal dificuldade relatada pelos estudantes foi com relação à visualização de como o laço `for` estava relacionado com as posições do vetor. Assim, na segunda aula, foi feita uma revisão da utilização do laço `for` com a estrutura de vetor. A docente escreveu no quadro o trecho de código referente ao `for` e reuniu a turma ao redor de uma mesa próxima ao quadro, com o vetor de caixa de ovo exposto. Após, seguiu as seguintes etapas:

1º etapa: Foi explicitado que o `vetor[5]`, declarado no código no quadro, refere-se ao vetor na mesa, contando a quantidade de posições. Essa é outra dificuldade expressa pelos estudantes na aula expositiva, a confusão entre a quantidade de posições e a contagem do laço `for`, por ser definido que na linguagem C, a primeira posição do vetor é obrigatoriamente o 0. O nome da posição impresso na caixa de ovos auxiliou nesta abstração.



(a)



(b)

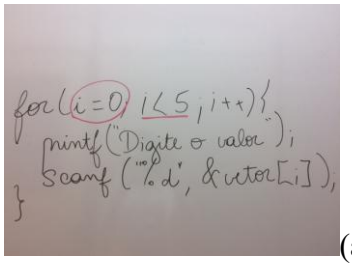
Figura 1. Atribuição do valor 0 à variável i

2º etapa: A docente então marca no quadro que a primeira instrução do `for` é executada `i=0`, a atribuição do valor zero à variável de controle `i` (Figura 1(a)). Neste momento, o post-it no formato de seta com o nome da variável de controle `i`, passa a apontar para a posição 0 do vetor (Figura 1(b)). Esta dinâmica explicita que o `i` está manipulando apenas a posição do vetor e que seu conteúdo do `vetor[0]` ainda não foi manipulado. É importante salientar que não é correto afirmar que a posição está

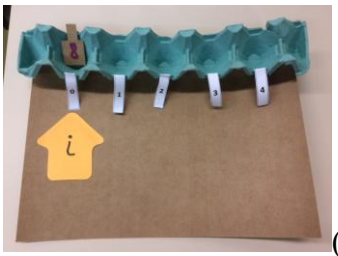
vazia, pois é sabido que a memória pode conter valores aleatórios enquanto não é diretamente manipulada por meio das variáveis. Mesmo assim, didaticamente, a posição parece vazia pela posição do "ovo" não conter nada ainda.

3º etapa: O próximo passo é salientar no código do quadro que a condição do laço será executada. Assim, a docente sublinha a condição `i<5` (Figura 2(a)) e pergunta aos estudantes se `i` (que neste momento possui o valor 0) é menor que 5, então os estudantes responderam que sim, e as instruções de dentro do laço serão executadas. Neste caso, a analogia da execução da instrução "`printf`" foi escrever um valor inteiro em um papel (no exemplo utilizaram o número 8) e a analogia da execução da instrução "`scanf`" foi colocar o papel dentro da posição do vetor (salientando o significado a instrução `&vetor[0]`). Essa atribuição é mostrada na Figura 2(b).

Esta dinâmica ajuda a diferenciar o valor da variável de controle do conteúdo da posição apontada pela variável de controle, e que o laço manipula o `i` e não os conteúdos.



(a)

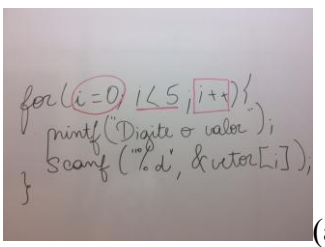


(b)

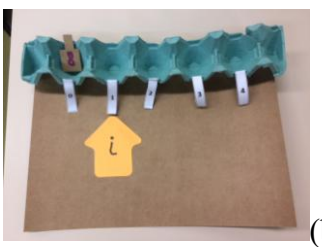
Figura 2. Atribuição do valor 9 à posição vetor [0]

4º etapa: A próxima etapa é salientar no código do quadro que o próximo passo a ser executado no laço é o incremento fazendo um retângulo em torno da instrução `i++` (Figura 3(a)). Com o incremento, neste momento a variável de controle `i` passa a possuir o valor 1, assim o incremento no vetor de ovos também acontece como mostra a Figura 3(b) e lembrando que a condição de `i<5` continua sendo verdadeira neste caso, então a execução das instruções dentro do laço será efetuada.

Neste momento a abstração do funcionamento da variável de controle `i` já está mais concreta e os estudantes já lidam com naturalidade com a analogia da instrução `printf` e `scanf`, em que eles escrevem novamente outro valor em um papel (neste exemplo está o inteiro 30) e colocam dentro da posição do `vetor[1]` (Figura 3(c)).



(a)



(b)

(c)

Figura 3. Execução do incremento da variável i e atribuição de valor

Após a 4ª etapa, a varredura e preenchimento das outras posições tornaram-se apenas a repetição desses passos, guiado pela execução do laço `for`. Após o vetor estar

completamente preenchido com os valores, foi feito um exercício semelhante para a leitura de todos os valores, bem como, exercícios que varrem o vetor de forma decrescente para que todas as oportunidades de manipulação de vetor ficassem mais concretas com o uso do vetor de caixa de ovo.

5º etapa: Após a docente fazer uso do quadro e do vetor em caixa de ovo, com todos os estudantes em grupo, foi proposta a seguinte atividade: os estudantes foram divididos em grupo e cada grupo recebeu um "kit" com os mesmos recursos utilizados pela docente contendo: um vetor de caixa de ovo, 5 papéis para escrever os valores do vetor e um post-it para o controle da variável `i`. Os estudantes deveriam utilizar o kit para resolver a lista de exercícios sobre vetores.

Em um primeiro momento, os estudantes ficaram um pouco resistentes para utilizar o kit, dizendo que não haveria necessidade porque já haviam entendido de forma concreta como manipular o vetor. Mesmo assim, foi possível observar que em alguns momentos, recorreram sim ao uso das caixas de ovos como pode ser visto na Figura 4.

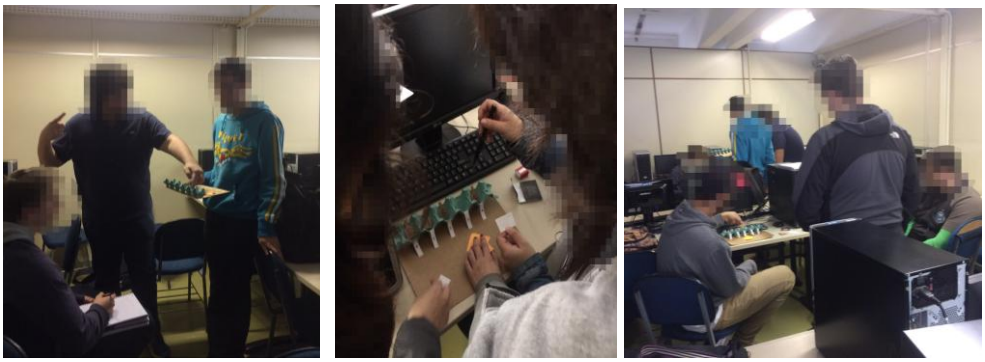


Figura 4. Estudantes utilizando os kits para resolver exercícios em grupo

Metodologia semelhante foi utilizada para o ensino de Matrizes, com a diferença que a primeira aula sobre Matrizes já foi utilizando o kit das caixas de ovos. Na Figura 5 pode ser visualizada a matriz representada com caixa de ovo. As caixas de ovos utilizadas são bastante grandes, e como a matriz para a dinâmica não precisa ser tão grande, foram cortadas matrizes quadradas de 3 linhas por 3 colunas. Dependendo da intencionalidade da aula, as matrizes podem ser não quadradas, sendo opção total do professor. Assim como no vetor, foram coladas tiras com os nomes das posições, nas matrizes as tiras possuíam um detalhe para demarcar o que estava sendo demarcado como linha (variável **i**) e como coluna (variável **j**). Para uma identificação rápida e visual, foram utilizadas linhas horizontais juntamente com os nomes das posições de linha (controlas pela variável `i`) e linhas verticais juntamente com os nomes das posições de coluna (controladas pela variável `j`). Também foram utilizados 2 post-it para a marcação das variáveis **i** e **j** na dinâmica. Primeiramente a docente trouxe uma caixa de ovos para representar uma matriz quadrada (Figura 5), 2 post-its para as variáveis de controle `i` e `j`, e papéis para colocar dentro das posições da matriz. A dinâmica também consistiu em escrever no quadro o trecho de código que percorre a matriz para atribuir valores e executar utilizando a caixa de ovos. Foram exploradas dinâmicas para preencher a matriz com valores, multiplicar matrizes e soma. Logo em seguida os grupos receberam kits para auxiliar a resolver os exercícios que envolviam

Matrizes. Após a execução das atividades com vetores e matrizes, os estudantes foram convidados a voluntariamente responder a um questionário de avaliação sobre sua experiência de aprendizado com as caixas de ovos. Os resultados são mostrados e discutidos a seguir.

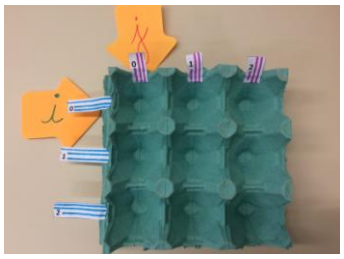


Figura 5. Matriz representada com caixa de ovo

4. Resultados e Discussões

Ao questionário sobre a experiência com a caixa de ovos no aprendizado de vetores, responderam 11 estudantes, visto que a resposta ao questionário era facultativa aos estudantes. O gráfico da Figura 6 mostra as respostas para a pergunta "Como você avalia a diferença que fez no seu aprendizado a explicação da docente sobre vetores com a caixa de ovos?". A maioria dos estudantes (64%) respondeu que ajudou muito por diminuir a abstração. Ainda assim, alguns estudantes responderam que fez pouca ou nenhuma diferença, cujo julgamento pode ser, em parte, devido à resistência que alguns estudantes demonstram quando são requisitados a utilizar uma ferramenta diferente do computador. Durante a proposta da atividade, alguns demonstraram resistência por parecer ser um material que lembrava ensino fundamental ou médio.

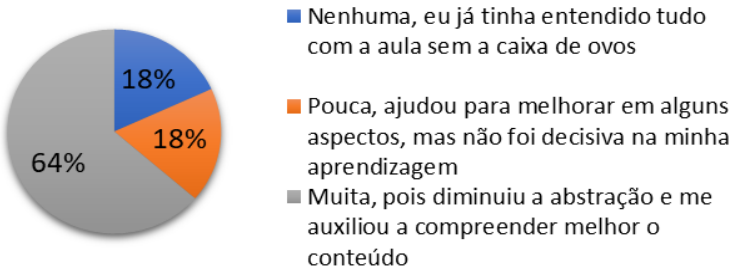


Figura 6. Respostas à pergunta sobre o aprendizado

O gráfico da Figura 7 mostra as respostas para a pergunta "Como você avalia a diferença que fez o vetor em caixa de ovos para o seu aprendizado sobre compreender a manipulação do incrementador dos índices (i)"? A maioria dos estudantes (73%) respondeu que o vetor em caixa de ovos ajudou muito no seu aprendizado por auxiliar a ver como o `i` funciona dentro do laço. Essas respostas dão fortes indícios de que a metodologia de fato auxilia na compreensão da manipulação do laço, pois mesmo quem respondeu que fez pouca diferença, ainda afirma que ajudou a melhorar a compreensão devido á manipulação concreta da variável `i` com o post-it.

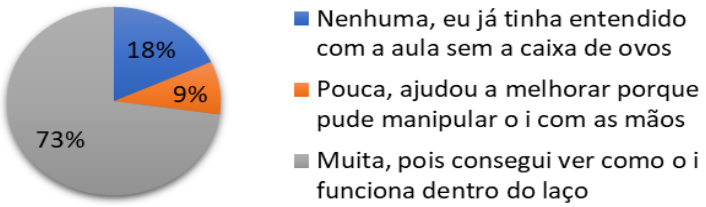


Figura 7. Respostas à pergunta sobre a variável de controle i

O gráfico da Figura 8 mostra as respostas para a pergunta "Como você avalia a diferença que fez no seu aprendizado a explicação da docente sobre compreender o que deve ficar dentro e fora do laço ao percorrer um vetor?". Observando todos os gráficos, este foi o que obteve maior porcentagem de resposta quanto à metodologia ter feito muita diferença (82% das respostas). Esse é um fator bastante positivo, por mostrar que o entendimento da estrutura do laço `for` não é trivial por exigir uma abstração de identificar em que momento exatamente cada uma das 3 instruções é executada. Com os destaques no código escrito no quadro, juntamente com a manipulação do vetor, auxilia a compreender. Quanto às outras respostas, novamente associamos à resistência com o uso do artefato.

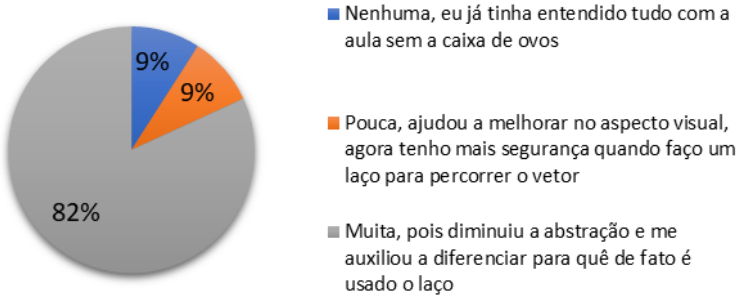


Figura 8. Respostas à pergunta sobre o laço que percorre o vetor

Os estudantes também foram convidados a voluntariamente avaliar seu aprendizado com as matrizes em caixas de ovos. Porém, nesta avaliação, apenas 4 estudantes responderam ao questionário. Para as perguntas "Como você avalia a diferença que fez no seu aprendizado a explicação da docente sobre matrizes com a caixa de ovos"? e "Como você avalia a diferença que fez no seu aprendizado a explicação com caixa de ovos para compreender o for aninhado?", as respostas refletiram 50% para "Muita pois diminuiu a abstração" e 50% para "Pouca, ajudou a melhorar no aspecto visual, assim adquiri mais segurança quando faço um laço aninhado para manipular matrizes" e nenhum aluno respondeu que não fez diferença alguma.

Para as matrizes salientamos as respostas à pergunta "Na sua opinião, você acredita que foi melhor aprender os conceitos primeiro com a caixa de ovos do que com slides apenas?", mostrado na Figura 9. Para esta pergunta, apesar de poucas respostas, a maioria (75%) dos estudantes afirma que aprender com os artefatos foi melhor do que apenas com slides, metodologia predominante em aulas expositivas.

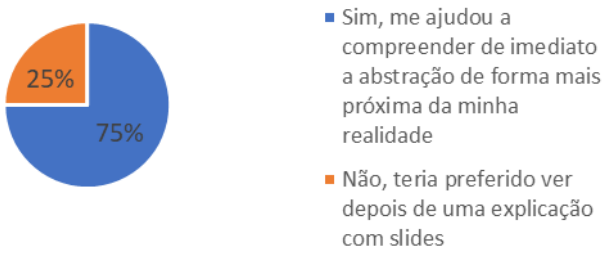


Figura 9. Resposta sobre o aprendizado de matrizes

Embora a aprendizagem significativa tenha ocorrido em aspectos como manipulação de laço, da variável de controle as relações disso tudo com os conteúdos das estruturas de dados, é interessante salientar a surpresa dos estudantes ao verem os kits de caixas de ovos bem como a resistência inicial em utilizarem os mesmos. Mesmo quando os estudantes confirmam que a caracterização visual das caixas de ovos em vetores e matrizes, juntamente com os códigos escritos no quadro fizeram com que o conteúdo passasse do nível abstrato para concreto, ainda assim, nota-se que a estratégia metodológica utilizada pela docente não foi de todo compreendida por eles como facilitadora da aprendizagem.

5. Conclusões

Historicamente, na disciplina de Computação em que a metodologia exposta neste artigo foi aplicada, o ensino de vetores e matrizes é um dos mais temidos conteúdos abordados. A partir dessa constatação, foi desenvolvida a metodologia que fez uso de um artefato sustentável construído artesanalmente com caixas de ovos, que de fato, resultou positivamente com relação à melhora na retenção de estudantes na disciplina. Apenas 1 aluno desta turma reprovou, e, apesar de não ser possível apenas com uma aplicação da metodologia afirmar que esse foi o fator determinante para o sucesso dos estudantes. Assim, considera-se que o método foi válido no sentido de concretizar conceitos abstratos para a aprendizagem significativa sobre algoritmos pelos estudantes.

Referências

Barbosa, E.F., Moura, D.G. (2013) Metodologias ativas de aprendizagem na educação profissional e tecnológica. Boletim Técnico SENAC. Rio de Janeiro, v.39, n.2, p.48-67, maio-ago.

Bell, T.; Witten, I.H., Fellows, M. (2011) Computer Science Unplugged: ensinando Ciência da Computação sem o uso do computador. Tradução coordenada por Luciano Porto Barreto. p.3- 45.

Berbel, N. (2011) As metodologias ativas e a promoção da autonomia dos estudantes. Semina: Ciências Sociais e Humanas, Londrina, v. 32, n. 1, p. 25-40

Bonwell, C. C., Eison, J.A. (1991) Active Learning. Creating Excitement in the Classroom. ASHE-ERIC Higher Education Report. 121 p.

Castro, R.M.C., Siqueira, S.W.M., Almeida, D.N.; Nascimento, F.C. (2017) Agility Scrum – Um jogo para Ensino da Metodologia SCRUM. 25 WEI – Workshop sobre Educação em Computação. São Paulo. p.2217-226

Cavalcante, F. P.L., Embiruçu, M. S. (2013) Aprendizado com base em problemas: Como entusiasmar os estudantes e reduzir a evasão nos cursos de graduação em engenharia. XLI Congresso Brasileiro de Ensino em Engenharia, v. 2013.

Da Silva, S.F., Barbosa, A.F., de Souza, A.A., da Silva, E.G., Silva, M.L., Neto, S.R.S., dos Santos, W.O. (2015) Relato de Experiência de Ensino de Computação no Ensino Fundamental em Estágio Supervisionado da Universidade de Pernambuco no Campus Garanhuns. In: XXIII Workshop sobre Educação em Computação.

Figueiredo, K. da S., Santos, J. C. O. (2016) Computasseia: Um Jogo para o ensino de História da Computação. In: XXIV Workshop sobre Educação em Informática (WEI 2016). p. 2026 – 2035.

Giraffa, Lucia, Mora, Michael. (2013) Evasão na Disciplina de Algoritmo e Programação: Um Estudo a partir dos Fatores Intervenientes na Perspectiva do Aluno. Conferência Latino-americana Sobre El Abandono En La Edcación Superior – III.

Junior, S.M.S., França, S.A. (2017) Programação para todos: Análise Comparativa de Ferramentas Utilizadas no Ensino de Programação. 25 WEI – Workshop sobre Educação em Computação. São Paulo.

Medeiros, S.M.O., Melo, J.C.B. (2011) Metodologias para o Ensino de Ciência da Computação na Educação Básica. XI Jornada de Ensino, Pesquisa e Extensão – JEPEX 2011 – UFRPE. Recife

Moreira, M.A. (2011) Aprendizagem significativa: a teoria e textos complementares. São Paulo: Livraria Editora da Física.

Nunes, I. F. e Parreira Júnior, P. A. (2015) RPG4Sorting - Um Jogo Educacional para Auxílio ao Ensino de Métodos de Ordenação, In: XXIII Workshop sobre Educação em Informática (WEI 2015).

Rogers, C. (1969) Freedom to learn: a view of what education might become. Columbus, OH: Charles E. Merrill Pub. Co. 358p.

Schoeffel, P., Wazlawick, R.S. (2016) Mão na Massa: Dinâmica Vivencial para Apoio ao Ensino de Gerenciamento de Projetos de Software. In: XXIV Workshop sobre Educação em Informática (WEI 2016) p. 2215 – 2224.

Sousa, R. V., Barreto, L. P., Andrade, A., Abdalla, D. (2010) Ensinando e aprendendo conceitos sobre ciência da computação sem o uso do computador: Computação Unplugged! Congresso Brasileiro de Informática na Educação. Volume 1, Nº1. p.5.

Tomisaki, S. M. M., Souza, A. D.; Seabra, R. D. (2016) MEGA GP: Aplicando a Gamificação no Ensino de Gerência de Projetos. In: XXIV Workshop sobre Educação em Informática (WEI 2016) p. 2225 – 2234.

Vahldick, A., Mendes, A. J., Marcelino, M. J., Hogenn, M., Schoeffel, P. (2015) "Testando a Diversão em um Jogo Sério para o Aprendizado Introdutório de Programação", In: XXIII Workshop sobre Educação em Informática (WEI 2015).

Weibell, C. J. (2011). Principles of learning: 7 principles to guide personalized, student-centered learning in the technology-enhanced, blended learning environment.

Percepção de estudantes sobre motivação e aprendizagem em Teoria da Computação com PBL

Luiz Otávio Ramos Gavaza[1], **Laís do Nascimento Salvador**[1], **David Moises Barreto dos Santos**[2]

[1]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Av. Adhemar de Barros, S/N, Ondina – 40.170-110 – Salvador – BA – Brasil

[2]Departamento de Ciências Exatas – Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, Novo Horizonte – 44.036-900 – Feira de Santana – BA – Brasil

{lgavaza,laisns}@ufba.br, davidmbs@uefs.br

Abstract. *The content of Computer Theory discipline has a high level of abstraction and complexity, and it is an important part of the formation of students in Computer Science courses. Studies has presented evidence that the traditional pedagogical approach has not been certified enough to keep students motivated and able to follow the contents of the subject of Computer Theory. Some constructivist pedagogical approaches have been experienced by educators and researchers in an attempt to improve computer students motivation and learning. This work uses the students' opinion survey to evaluate their perception about motivation and learning in a Computer Theory discipline using the constructivist approach Problem-Based Learning (PBL) in a computer class at the Federal University of Bahia. The results of the experiment showed that the students had good perceptions about the PBL approach, which confirms the potential for the use of PBL in the teaching and learning of Computer Theory discipline.*

Resumo. *O conteúdo de Teoria da Computação possui alto nível de abstração e complexidade, e é parte importante da formação dos estudantes dos cursos de Computação. Estudos levantam indícios de que a abordagem pedagógica tradicional não tem demonstrado ser suficiente para que os estudantes se mantenham motivados e consigam acompanhar os conteúdos da disciplina de Teoria da Computação. Algumas abordagens pedagógicas construtivistas têm sido experimentadas pelos educadores e pesquisadores na tentativa de melhorar a motivação e a aprendizagem dos estudantes de Computação. Este trabalho utiliza pesquisa de opiniões dos estudantes para avaliação da percepção deles sobre a motivação e aprendizagem em uma disciplina de Teoria da Computação com a utilização da abordagem construtivista Aprendizagem Baseada em Problemas (PBL), em uma turma de Computação na Universidade Federal da Bahia. Os resultados da experiência mostraram que os estudantes participantes tiveram boas percepções sobre a abordagem PBL, o que confirma o potencial para utilização de PBL no ensino e aprendizagem da disciplina de Teoria da Computação.*

1. Introdução

Ao mesmo tempo em que é crescente o número de estudantes universitários, é grande a evasão. Os altos índices de desistências nos cursos superiores são evidências da necessi-

dade de compreensão das variáveis motivacionais dos estudantes.

A motivação do estudante é um determinante crítico do nível e da qualidade da aprendizagem e do desempenho no contexto escolar. Desta forma, muitos trabalhos têm relacionado o desempenho acadêmico dos estudantes com a motivação [Zenorini et al. 2011, Rufini et al. 2011].

A formação dos estudantes de Computação exige que estes sejam capazes de construir um conhecimento em disciplinas que possuem um alto nível de abstração. A disciplina de Teoria da Computação é uma das disciplinas teóricas com alto nível de abstração nos cursos da área de Computação que exige bastante esforço por parte dos estudantes para que possam acompanhar o andamento, para assim, construírem conhecimento que permita compreender as bases de sustentação da área.

O que percebemos é que a maioria dos cursos da área de Computação utilizam, na maioria das vezes, abordagem de ensino e aprendizagem tradicional, com raras exceções onde são realizadas atividades práticas em laboratórios. Ainda assim, devemos destacar que apenas para algumas disciplinas existem atividades além das aulas expositivas.

A abordagem de ensino e aprendizagem tradicional não tem por objetivo a formação crítica de estudantes, uma habilidade desejável para os estudantes da área de Computação. Além disto, é necessário considerar que na revisão do currículo de Ciências da Computação da *Association for Computing Machinery* (ACM) em 2013, estão descritas características esperadas para os estudantes de graduação, como a compreensão técnica da ciência da computação, familiaridade com temas e princípios comuns, interação entre teoria e prática, visão sistêmica, habilidades para resoluções de problemas e experiências significativas em projetos, compromisso com a continuidade da aprendizagem, habilidades de comunicação e organização e consciência da aplicabilidade ampla da computação [Sahami et al. 2013].

Existem inúmeras abordagens pedagógicas que poderiam ser utilizas para desenvolvimento crítico dos estudantes e desenvolvimento da habilidades mencionadas na revisão do currículo de Ciências da Computação da ACM em 2013, mas entre as mais promissoras que serviriam as nossas necessidades, sobretudo na possibilidade de desenvolver habilidades de pensamento crítico, se pode destacar a abordagem baseada em problemas.

A abordagem de Aprendizagem Baseada em Problemas, em inglês *Problem Based Learning* (PBL), foi criada na Universidade McMaster nos anos de 1960 em cursos de medicina, onde é amplamente utilizada [Albanese 2010, Amos and White 1998].

A abordagem PBL pode ser definida como uma abordagem educacional construtivista ativa que usa problemas como contexto para que os estudantes adquiram conhecimentos sobre os conceitos. O foco de aprendizagem está nos estudantes, que são capacitados para que assumam a responsabilidade pela aprendizagem. Nessa abordagem os professores deixam o papel de repositório de conhecimento da abordagem de ensino e aprendizagem tradicional e assumem o papel de tutores atuando como facilitadores da construção do conhecimento [Dolmans et al. 2005, Albanese 2010, Amos and White 1998, Forsythe 2002].

A metodologia baseada em problemas deve capacitar os estudantes a realizar pesquisas, integrar teoria e prática e aplicar conhecimentos e habilidades para desenvolver

uma solução viável para um problema definido [Savery 2015].

No caso da abordagem baseada em problemas, entre as várias qualidades, podemos destacar que o processo possui como grande objetivo "ensinar a aprender" para que os estudantes possam assumir a responsabilidade pela aprendizagem. Na abordagem baseada em problemas os estudantes são capacitados para as investigações e a aprendizagem dos conceitos será consequência das investigações.

A motivação e qualidade da aprendizagem do estudante é fundamental para que possam construir um conhecimento mais sólido e amplo, investigar outras linhas de estudo e ter opiniões críticas sobre os conceitos. Este trabalho apresenta resultados para uma investigação sobre a percepção de motivação e de aprendizagem de estudantes da disciplina de Teoria da Computação na Universidade Federal da Bahia (UFBA).

Além desta seção introdutória, este trabalho está organizado da seguinte forma: a Seção 2 apresenta o contexto educacional; a Seção 3 para descrever a metodologia; a Seção 4 para apresentar os trabalhos correlatos; a Seção 5 para trazer uma discussão sobre alguns dos resultados obtidos com a experiência; por fim, a Seção 6 para trazer algumas das argumentações sobre consequências dos resultados exibidos na Seção anterior, as expectativas e as possibilidades de trabalhos na investigação proposta.

2. Contexto Educacional

Os cursos de Computação na UFBA utilizam majoritariamente a abordagem tradicional, onde algumas disciplinas específicas contam com carga horária específica para laboratórios práticos. Existem algumas poucas iniciativas de novas abordagens como é o caso deste estudo.

A disciplina em que este estudo foi realizado, apesar de introdutória, trabalha com diversos dos conceitos da Teoria da Computação, incluindo a Teoria da Complexidade e dos Compiladores.

Os estudantes estão entre o terceiro e o quinto semestre de um curso noturno em uma universidade pública, são majoritariamente do sexo masculino e possuem idade entre 19 e 44 anos.

A carga horária de 68 horas da disciplina no semestre foi dividida em: 26 horas para a realização de sessões tutoriais de PBL em que foi conduzida a abordagem; 32 horas para a realização de aulas expositivas em que o educador apresentou conceitos da disciplina; e 10 horas para a realização de avaliações tradicionais.

O conteúdo da disciplina, além dos problemas, foi trabalhado com aulas expositivas e construção de um projeto de desenvolvimento. A avaliação de conceito dos estudantes foi realizada com atribuição de notas para o projeto de desenvolvimento para os problemas e duas avaliações escritas individuais e sem consultas. No caso dos problemas, a nota considerou assiduidade e participação dos estudantes nas discussões das sessões tutoriais e o produto produzido como solução para o problema.

A condução da disciplina estimulou os estudantes a utilizarem um ambiente virtual de aprendizagem (AVA) institucional para continuidade das discussões sobre os problemas e conceitos em fóruns, além de repositório para armazenar os conteúdos produzidos.

3. Metodologia

No estudo foi utilizada uma metodologia de pesquisa descritiva na forma de pesquisa de opinião que contou com a participação de 50 estudantes que foram selecionados por oportunidade de estarem inscritos na disciplina em que este estudo foi realizado.

A experiência aconteceu em uma sala de aula tradicional equipada com um quadro branco e com quadros adicionais feitos com papel metro colados nas paredes da sala. Nas sessões tutorias os estudantes foram distribuídos em grupos de cinco até dez participantes. Os grupos formaram semicírculos de forma que todos foram capazes de enxergar o quadro adicional referente ao grupo ao qual foram designados e tiveram interação dentro do grupo facilitada.

A metodologia de pesquisa realizou aplicações de cinco problemas PBL no semestre 2017.1 que foram construídos baseados em sete princípios apresentados por Dolmans et al. (1997).

A abordagem PBL foi exemplificada para os estudantes utilizando um problema que foi construído especificamente com este propósito, o problema "A torta e o ladrão de pimenta". Podemos destacar que este problema trabalha os conceitos básicos de inferência lógica, assim, poderia também utilizado nesse contexto de ensino e aprendizagem. Uma cópia está disponível em `https://goo.gl/5DD83a`.

Os problemas inicialmente fornecem aos estudantes pequenos desafios e a cada novo problema tem um aumento gradual na complexidade em relação ao anterior, que é obtido removendo orientações específicas, assim, os problemas apresentam cada vez mais características dos problemas encontrados no mundo real [Fee and Holland-Minkley 2010].

Segue a lista dos problemas:

- Problema 1 - O código Morse – O problema menciona um grupo de estudantes programadores que estão naufragados em uma ilha deserta e precisam submeter um projeto de desenvolvimento. Ao serem resgatados devem discutir como utilizar um telégrafo para realizar a transmissão.
  O objetivo desse problema é trabalhar os conceitos de Linguagens Formais por meio de uma equivalência com o código Morse.
  Disponível em `https://goo.gl/M9r9oA`
- Problema 2 - A máquina de vender refrigerantes – O problema menciona uma empresa que deseja construir uma máquina de vender refrigerantes.
  O objetivo desse problema é trabalhar os conceitos de Autômatos Finitos e introduzir o conceito de não determinismo.
  Disponível em `https://goo.gl/gGg1Ch`
- Problema 3 - O controle de tráfego – O problema menciona a situação de buracos em uma estrada e solicita aos estudantes que realizem o balanceamento da proporção entre veículos leves e pesados.
  O objetivo desse problema é trabalhar os conceitos de Autômatos Finitos com Pilha.
  Disponível em `https://goo.gl/BcTJ7w`
- Problema 4 - O controle de tráfego – O problema é uma extensão do Problema 3. O objetivo é trabalhar os conceitos de Máquinas de Turing.
  Disponível em `ttps://goo.gl/2LXKku`

- Problema 5 - Uma função bastante curiosa – O problema relata sobre a capacidade dos compiladores em identificar erros e otimizar códigos e convida os estudantes a discutirem o quanto um compilador pode ser inteligente.
  O objetivo é trabalhar o Problema da Parada.
  Disponível em `https://goo.gl/rUyNov`

O conteúdo da Teoria da Complexidade, que está presente entre os conteúdos da disciplina em que aplicamos a metodologia, foi trabalhado com um problema em que os estudantes discutiram nos mesmos moldes da abordagem PBL, entretanto, ao invés de apresentarem um produto como solução, eles responderam uma lista de questões. Este problema foi utilizado no critério de avaliação dos estudantes. Uma cópia está disponível em `https://goo.gl/bewLVy`.

Para trabalhar os conceitos de Teoria dos Compiladores foi utilizado um projeto de implementação de uma calculadora pós-fixa. Uma cópia da especificação está disponível em `https://goo.gl/6zW9Wh`.

Para cada problema os estudantes foram convidados a responder no AVA um formulário de percepções com itens de caracterização de perfil, com afirmativas em escala Likert de cinco níveis e com um espaço aberto. Para evitar que dados da pesquisa modificassem a condução da metodologia, os dados foram processados apenas ao final de toda coleta. A assinatura do termo de consentimento livre e esclarecido pelos participantes foi digital, no AVA.

4. Trabalhos correlatos

Os educadores e pesquisadores têm buscado alternativas pedagógicas para motivação e aprendizagem dos estudantes em disciplinas de Computação e têm relatado as experiências, assim como é realizado neste trabalho.

A utilização mecânica de conceitos de Teoria da Computação por parte dos estudantes é a justificativa para a introdução de abordagens construtivistas no trabalho Chesnevar et al. (2004). Neste trabalho uma série de abordagens construtivistas foram combinadas e aplicadas por três semestres. A principal conclusão do trabalho é que os temas da disciplina de Teoria da Computação podem se tornar mais interessante se for utilizada uma combinação de diferentes abordagens. Entendemos que assim como em nosso trabalho, apesar de neste não está explicito, também foi realizada uma pesquisa de percepção com os estudantes, que também apresentou elevado índice de satisfação com abordagens construtivistas, mas a quantificação da satisfação não foi apresentada em resultados.

A utilização de uma ferramenta de aprendizagem é a proposta pedagógica em Vieira et al. (2003), onde é construída uma ferramenta denominada de *Language Emulator* onde os estudantes podem utilizar expressões regulares, gramáticas regulares e autômatos finitos. O principal destaque deste trabalho é que os estudantes podem obter respostas tempestivamente, assim, isto pode contribuir no processo de motivação, entretanto, apesar disto, e de ter obtido uma aprovação de 95% dos participantes, apenas alguns dos conceitos de Teoria da Computação estão contemplados pela ferramenta.

Os jogos educacionais são propostas comuns para abordagem pedagógica, e algumas são as propostas que utilizam desta alternativa para motivação e aprendizagem de

estudantes em disciplinas de Computação. Os aspectos lúdico e interativo dos jogos educacionais pode ser uma boa alternativa para auxiliar na aprendizagem e motivação dos estudantes [Silva et al. 2010].

O trabalho Leite et al. (2014) propõe a utilização de um jogo educacional com um ambiente para correção automática de exercícios de Teoria da Computação. Apesar de bem avaliada pelos estudantes, a ferramenta contempla apenas uma parte dos temas da disciplina de Teoria da Computação.

A utilização de um jogo educacional também é a proposta descrita em de Morais et al. (2011) como abordagem pedagógica para motivação dos estudantes e aprendizagem dos conteúdos da disciplina de Teoria da Computação. A ideia de os estudantes terem um aplicativo móvel permite que eles tenham acesso facilitado aos conceitos da disciplina de Teoria da Computação a qualquer momento, entretanto, neste trabalho o jogo apresentado parece uma lista de exercícios e a única história, sem diversos caminhos possíveis, pode não ser suficiente para motivação dos estudantes.

O curso de Engenharia da Computação na Universidade Estadual de Feira de Santana utiliza uma abordagem com PBL desde a criação do curso em 2003. A escolha pela utilização da abordagem baseadas em problemas ocorreu desde a elaboração do Projeto de curso. A flexibilidade é uma das principais características do curso, onde é permitido aos estudantes não só dentro das disciplinas escolherem por caminhos diversos para a solução dos problemas e construção do conhecimento, mas também na forma como este estudante vai integralizar o seu currículo para formação. Para aplicação da abordagem PBL, no curso da UEFS, são utilizadas infraestruturas específicas [dos Santos et al. 2007, Bittencourt and Figueiredo 2003]. O curso possui dez módulos curriculares em que são aplicados a metodologia PBL, entre eles podem ser citados *Estruturas de Dados* e *Engenharia de Software*. Apesar de consolidado o método PBL na UEFS, não há aplicação da metodologia PBL para a disciplina de Teoria da Computação [dos Santos and Burnham 2003].

5. Resultados

Apesar de ser uma disciplina situada no início do curso, entre os estudantes participantes deste estudo é grande a quantidade de estudantes que já abandonaram uma outra disciplina do curso. A disciplina deste estudo também possui historicamente um índice elevado de evasão. Para exemplificar sobre a evasão, utilizando uma abordagem tradicional com aulas expositivas, a disciplina teve 14 estudantes desistentes em um total de 28 estudantes matriculados no semestre 2014.1, e 10 estudantes desistentes em 24 estudantes matriculados no semestre 2015.1.

Como a participação nas pesquisas foi opcional, a partir deste ponto, o termo "participante" é utilizado para designar apenas o estudante que respondeu a pesquisa mencionada na discussão, não devendo ser confundido com o termo "estudante", que a partir deste ponto é utilizado para designar o participante da metodologia, independente de participação na pesquisa.

Os resultados apresentados neste trabalho utilizaram as respostas dos participantes nas afirmativas de percepção sobre motivação e aprendizagem. A apresentação dos resultados utiliza gráficos dispostos em barras, que foram construídas de forma a facilitar a leitura por nível de concordância para cada aplicação de problema. Cada uma das

barras representa uma aplicação, que está identificada conforme a numeração utilizada na Seção 3. A concordância pode ser lida por porcentagem de satisfação, em y_1 à esquerda, ou adicionalmente, por porcentagem de insatisfação em y_2 à direta.

Para as discussões e conclusões apresentadas neste trabalho será avaliada a concordância na percepção dos participantes. As percepções com algum nível de concordância positiva, isto é, o participante respondeu "concordo" ou "concordo parcialmente" para a afirmativa será considerada com avaliação favorável.

A Figura 1 exibe um gráfico com os resultados de percepção dos participantes para a motivação nas cinco aplicações de problemas deste trabalho. A informação é obtida com a extração de dados das respostas dos participantes para a afirmativa "O problema lhe deixou motivado para descobrir uma possível solução".

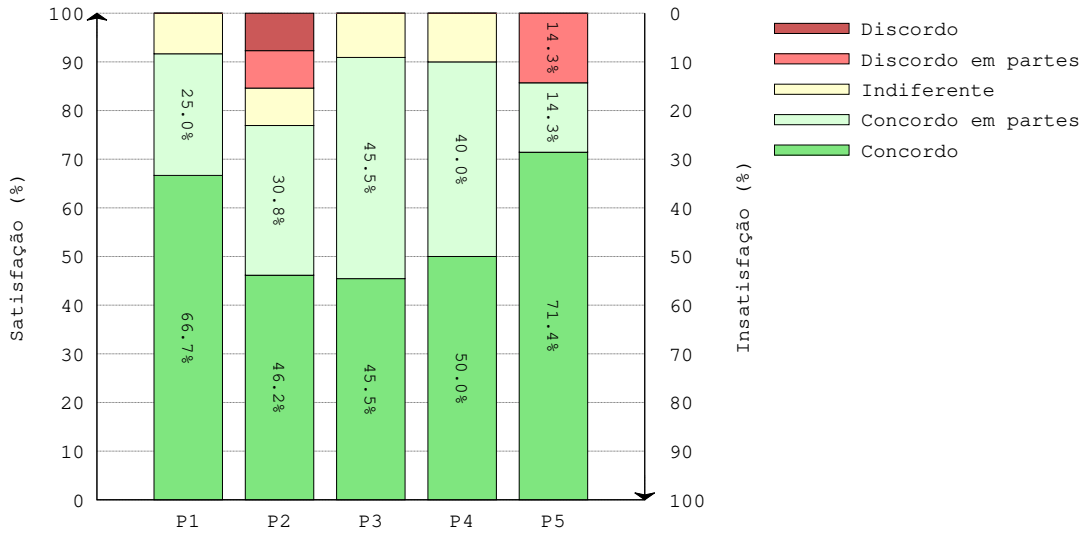


Figura 1. Percepções dos participantes sobre motivação

Em todas as aplicações dos problemas a percepção dos participantes para a motivação obteve bons resultados. A avaliação menos favorável, na aplicação do problema "A máquina de vender refrigerantes", obteve satisfação com relação à percepção de motivação por parte de $77,0\%$ dos participantes. Em três das cinco aplicações ao menos $90,0\%$ dos participantes se sentiram motivados.

A Figura 2 exibe um gráfico com os resultados de percepção dos participantes para a aprendizagem nas cinco aplicações deste trabalho. A informação é obtida com a extração de dados das respostas dos participantes para a afirmativa "Você acredita que cumpriu com os objetivos de aprendizagem do problema".

A percepção de aprendizagem também obteve bons resultados em todas as replicações. A avaliação menos favorável, na aplicação do problema "Uma função bastante curiosa", obteve satisfação com relação à percepção de aprendizagem por parte de $57,2\%$ dos participantes. Em três das cinco replicações ao menos $80,0\%$ dos participantes acreditaram ter cumprido com os objetivos de aprendizagem.

É possível perceber diferenças entre os resultados com relação à percepção de aprendizagem pelos participantes nas replicações. Uma alternativa é discutir essas

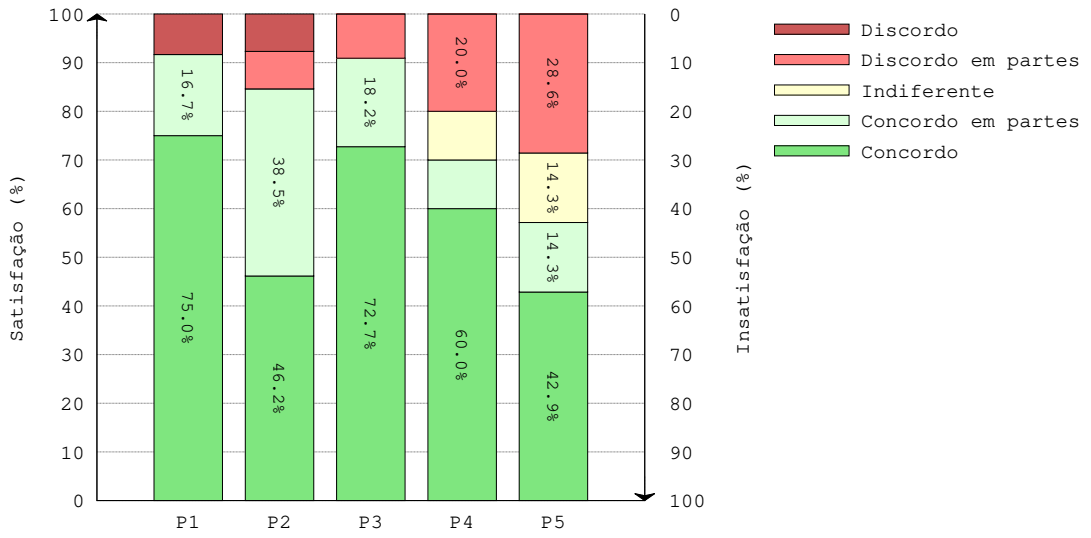


Figura 2. Percepções dos participantes sobre aprendizagem

diferenças do ponto de vista da crescente dificuldade e abstração dos conteúdos dos problemas. O processo de incremento da dificuldade ou desafio nos problemas pode não funcionar perfeitamente em uma sala de aula tradicional com todas as dificuldades que conhecemos, por este motivo, a transição entre os problemas pode não amortizar suficientemente o incremento de dificuldade e nível de abstração necessários para entendimento dos conceitos, assim, os estudantes podem ter sentido uma ruptura, como consequência, a percepção com relação à aprendizagem é apresentada com uma tendência de redução.

Os participantes inclusive mencionaram sobre as dificuldades com os conceitos da disciplina no espaço aberto da aplicação do problema “Uma função bastante curiosa”.

Os comentários em relação à metodologia PBL foram na maioria elogiosos. Foi mencionado que a metodologia ajuda no processo de aprendizagem da disciplina, além de ser interessante e motivadora.

A aprendizagem mais profunda dos conteúdos e a necessidade de mais esforço por parte dos estudantes também foram destacadas. Acreditamos que nesse contexto, o objetivo de fazer com que os estudantes assumam também a responsabilidade pela a sua aprendizagem fica explicitada.

6. Conclusão

Os bons resultados da experiência deste estudo podem ser utilizados como argumentação da possibilidade de utilização da abordagem baseada em problemas para motivação e aprendizagem de estudantes em disciplinas teóricas de Computação.

Nessa experiência percebemos que alguns estudantes possuem um perfil mais passivo de aprendizagem ou possuem outras obrigações que é utilizada por estes como justificativa para uma menor dedicação em buscar conhecimento, nesse contexto, traçar um perfil mais detalhado dos estudantes pode permitir a construção de problemas que estejam mais alinhados com estes perfis.

A motivação e absorção de conhecimento pelos estudantes deve ser avaliada com-

parativamente com outras abordagens, inclusive a abordagem pedagógica tradicional. Também é necessário uma avaliação sobre a absorção de conhecimento além da percepção dos estudantes, por exemplo, verificar se a utilização da abordagem baseada em problemas é capaz de melhorar o desempenho dos estudantes em avaliações tradicionais.

No que diz respeito a motivação dos estudantes, se faz necessário realizar futuramente uma análise que realize cruzamento com dados de evasão. A identificação dos principais motivos para evasão dos estudantes pode permitir ações efetivas na reversão da evasão, inclusive contribuir na construção de problemas mais efetivos para utilização de PBL como ferramenta para redução de evasão.

Estudar as correlações entre variáveis como percepção de motivação e percepção de aprendizagem também é um trabalho futuro. Nesse estudo eventualmente pode surgir um importante indicador para o educador acompanhar o envolvimento e desenvolvimento dos estudantes.

Além das contribuições diretas nos resultados, como na discussão do potencial da abordagem PBL para o ensino e aprendizagem de disciplinas teóricas de Computação, este trabalho também trouxe diversas questões de estudos futuros.

Agradecimentos

Os autores deste trabalho gostariam de agradecer aos revisores anônimos deste trabalho pelas contribuições que ajudaram a melhorar a versão final deste artigo.

Referências

Albanese, M. A. (2010). Problem-based learning. In *An introduction to medical teaching*, pages 41–53. Springer.

Amos, E. and White, M. J. (1998). Problem-based learning. *Nurse Educator*, 23(2):11–14.

Bittencourt, R. A. and Figueiredo, O. A. (2003). O currículo do curso de engenharia de computação da uefs: Flexibilização e integração curricular. In *XI Workshop sobre Educação em Computação–Anais do XXIII Congresso da Sociedade Brasileira de Computação–Anais do*, pages 171–182.

Chesnevar, C. I., González, M. P., and Maguitman, A. G. (2004). Didactic strategies for promoting significant learning in formal languages and automata theory. *ACM SIGCSE Bulletin*, 36(3):7–11.

de Morais, D. C. S., Alencar, A. D., and de Souza, R. (2011). Jogo baseado em m-learning e aprendizado tangencial para auxílio ao ensino de teoria da computação. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1.

Dolmans, D. H., De Grave, W., Wolfhagen, I. H., and Van Der Vleuten, C. P. (2005). Problem-based learning: Future challenges for educational practice and research. *Medical education*, 39(7):732–741.

Dolmans, D. H., Snellen-Balendong, H., and van der Vleuten, C. P. (1997). Seven principles of effective case design for a problem-based curriculum. *Medical Teacher*, 19(3):185–189.

dos Santos, D. M. B. and Burnham, T. F. (2003). O pensamento de paulo freire e pbl: primeiras aproximações e afastamentos. In *XI Workshop sobre Educação em Informática (WEI 2003), Campinas, SP*.

dos Santos, D. M. B., Pinto, G., Sena, C. P. P., Bertoni, F. C., and Bittencourt, R. A. (2007). Aplicação do método de aprendizagem baseada em problemas no curso de engenharia da computação da universidade estadual de feira de santana. In *Congresso Brasileiro de Educação em Engenharia-COBENGE*.

Fee, S. B. and Holland-Minkley, A. M. (2010). Teaching computer science through problems, not solutions. *Computer Science Education*, 20(2):129–144.

Forsythe, F. (2002). Problem-based learning. *The Handbook for Economics Lecturers: Teaching, Bristol: Economics LTSN, http://www. economicsnetwork. ac. uk/handbook*.

Leite, L. S., Sibaldo, M. A. A., Carvalho, T., and Souza, R. (2014). Montanha de chomsky: jogo tutor para auxílio no ensino de teoria da computação. In *XXII Workshop sobre Educação em Informática (WEI 2014), Brasília, DF*.

Rufini, S. É., Bzuneck, J. A., and de Oliveira, K. L. (2011). Estudo de validação de uma medida de avaliação da motivação para alunos do ensino fundamental. *Psico-USF*, 16(1):1–9.

Sahami, M., Roach, S., Cuadros-Vargas, E., and LeBlanc, R. (2013). Acm/ieee-cs computer science curriculum 2013: reviewing the ironman report. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 13–14. ACM.

Savery, J. R. (2015). Overview of problem-based learning: Definitions and distinctions. *Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows*, 9:5–15.

Silva, R. C., Binsfeld, R. L., Carelli, I. M., and Watanabe, R. (2010). Automata defense 2.0: reedição de um jogo educacional para apoio em linguagens formais e autômatos. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 1.

Vieira, L. F. M., Vieira, M. A. M., and Vieira, N. J. (2003). Language emulator, uma ferramenta de auxílio no ensino de teoria da computação. In *XIII Workshop sobre Educação em Computação–XXV Congresso da Sociedade Brasileira de Computação*.

Zenorini, R. d. P. C., Santos, A. A. A. d., and Monteiro, R. d. M. (2011). Motivação para aprender: relação com o desempenho de estudantes. *Paidéia*, 21(49):157–164.

Uso do Scratch na Introdução de Conceitos de Lógica de Programação: relato de experiência

Carina Machado de Farias[12], Anderson S. de Oliveira[1], Everton Dias de A. Silva[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Av. Centenário, 500, Nazaré - Jacobina - Bahia, 44700-000, Brasil

[2] Pesquisadora do Grupo de Pesquisa Automação, Eficiência Energética e Produção

carina.farias@ifba.edu.br, anderson.s.o@hotmail.com,
diaseverton@live.com

Abstract. *Learning programming is not a simple task for most of the students of the technical and superior courses in Informatics. This difficulty is reflected in the high rates of avoidance and disapproval in the introductory programming disciplines. Among the factors that influence the low performance of students and their difficulties in assimilating the contents of programming, the precarious logical-mathematical background of the students and the lack of previous experience in programming stand out. In order to promote a first contact with the students' programming, before the beginning of the classes, an extension project was carried out at the IFBA, Jacobina campus, where the students entering the Subsequent Information Technology Course participated in an introductory programming course, using the Scratch tool. This article reports on the effects of the project execution in relation to the failure rates in the Course of Programming of the said course in the second semester of 2017.*

Resumo. *Aprender programação não é uma tarefa considerada simples por grande parte dos estudantes dos cursos técnicos e superiores de Informática. Essa dificuldade é refletida nos altos índices de evasão e reprovação nas disciplinas introdutórias de programação. Dentre os fatores que influenciam o baixo rendimento dos alunos e suas dificuldades em assimilar os conteúdos de programação destacam-se o precário embasamento lógico-matemático dos estudantes e a falta de experiência prévia em programação. A fim de promover um primeiro contato com programação para os alunos, antes do início das aulas, foi realizado um projeto de extensão no IFBA, campus Jacobina, onde os alunos ingressantes no Curso Técnico Subsequente de Informática participaram de um curso introdutório de programação, utilizando a ferramenta Scratch. Este artigo relata os efeitos da execução do projeto em relação às taxas de reprovação na disciplina Lógica de Programação do referido curso no segundo semestre de 2017.*

1. Introdução

A disciplina Lógica de Programação, presente nas primeiras etapas dos cursos técnicos e superiores da área de Computação, constitui a base de formação dos estudantes, requerendo deles uma nova forma de pensar e habilidades que dificilmente são

desenvolvidas no ensino regular, levando os estudantes a terem grandes dificuldades e consequentemente apresentarem baixo rendimento durante o curso. (GUEDES, 2014)

Analisando os dados repassados pela Coordenação de Registros Escolares, do IFBA, campus Jacobina, ao longo dos anos de 2013 a 2016, percebe-se que grande parte dos alunos dos cursos técnicos de Informática apresenta dificuldades em assimilar as abstrações envolvidas nessa disciplina, levando a uma alta taxa de reprovação, como pode ser visto na Tabela 1.

Tabela 01: Aprovações e Reprovações na Disciplina Lógica de Programação

INGRESSO	INICIANTES	APROVADOS	REPROVADOS	TAXA DE REPROVAÇÃO
2013.2	42	18	24	57,14%
2014.2	42	15	27	64,28%
2015.2	37	03	34	91,89%
2016.2	50	26	24	48%

A origem do problema das reprovações na disciplina de Lógica de Programação diz respeito a alunos, professores e até mesmo às metodologias utilizadas, sendo possível destacar alguns fatores que contribuem para o cenário atual: precária base lógico-matemática dos alunos; falta de dedicação aos estudos; limitações do professor; material didático de apoio ineficiente; etc.

Considerando a importância dessa problemática e buscando alternativas para minimizar o número de reprovações, foi desenvolvido no IFBA, campus Jacobina, um projeto de extensão que propôs introduzir os conteúdos da lógica de programação, de maneira contextualizada, e fazendo uso de estratégias e metodologias alternativas ao ensino tradicional, para os alunos matriculados no 1º semestre do Curso Técnico Subsequente de Informática, visando promover um primeiro contato dos alunos com a área de programação antes do início das aulas, buscando, com isso, reduzir as dificuldades apresentadas pelos alunos no decorrer do semestre letivo. O curso fez uso da ferramenta Scratch, por ser uma ferramenta muito popular para o ensino de programação para quem não tem experiência prévia com a temática (SOUZA E CASTRO, 2016).

2. Trabalhos Relacionados

No decorrer dessa pesquisa foram encontrados alguns trabalhos que tratam de diferentes estratégias para tornar o processo de ensino-aprendizagem dos conteúdos de lógica de programação mais interessante, além da inclusão desses conteúdos na educação básica.

Em Friedrich et al. (2012), foi encontrada uma proposta de metodologia pedagógica para a inserção da lógica de programação para crianças de 7 a 10 anos de idade, fazendo uso do programa Logo e do projeto Lego Mindstorms.

Já Dias e Serrão (2014), relataram uma experiência do uso do Scratch com um grupo de alunos do primeiro ano do curso de Licenciatura da Computação. O trabalho

analisou os efeitos que essa ferramenta promoveu sobre o aprendizado dos conceitos mais elementares de programação de computadores no grupo observado.

Gomes et al. (2014), usaram o Scratch com um grupo de meninas do ensino médio com o intuito de aumentar o interesse das meninas pela área de computação.

Ferreira et al. (2016) também buscaram incentivar jovens e adultos através da oferta de um curso de iniciação à programação de computadores às comunidades de Salvador e região metropolitana. Nesse projeto foram utilizadas as ferramentas Scratch e Visualg.

Algumas experiências com o ensino fundamental também foram encontradas, como o trabalho de Silva et al. (2016) que relata a aplicação do Scratch no ensino de programação para alunos de 5º ano do ensino fundamental. Já Oliveira et al. (2014) e Farias et al. (2017), focaram nos alunos do 9º ano do ensino fundamental da escola pública.

Souza e Castro (2016) apresentam uma revisão sistemática da literatura referente ao uso do Scratch no ensino de programação para crianças. O trabalho incluiu um total de 12 artigos publicados no período entre 2007 e 2016 e apontou que o Scratch tem sido uma ferramenta popularmente utilizada no ensino de programação para crianças. De forma semelhante, Batista et al. (2016), faz um apanhado de relatos de experiência sobre o uso do Scratch no ensino de programação em diferentes níveis de escolaridades, desde as séries iniciais do ensino fundamental, passando pelo ensino médio até chegar no nível superior. O trabalho concluiu que o Scratch se apresenta como uma ferramenta interessante que pode ser adequada a diferentes faixas etárias e pode ser utilizada para ensinar instruções básicas de programação, favorecendo o raciocínio lógico e desenvolvendo habilidades para solução de problemas.

3. Ferramenta Scratch

Scratch[1] é um ambiente de programação desenvolvido pelo grupo de pesquisa Lifelong Kindergarten Group do Laboratório de Mídias do Massachusetts Institute of Technology (MIT), e tem por objetivo introduzir a programação de maneira fácil e rápida para aqueles que não possuem nenhum tipo de experiência, sendo apropriado para crianças a partir dos 8 anos de idade.

Em virtude do seu público alvo principal, a linguagem oferece um ambiente visual para programação, visando torná-la mais acessível que outras linguagens, disponibilizando uma interface que permite que programas sejam montados como blocos virtuais, arrastando-os para a área de trabalho, que é um plano de fundo estático, onde os elementos inseridos podem executar as suas ações. Para dar vida a seus projetos e definir o comportamento de cada um dos elementos presentes, os usuários associam a estes objetos sons, imagens e variáveis que podem ser manipulados pelos diferentes tipos de comandos fornecidos pela linguagem de programação do ambiente.

A linguagem Scratch disponibiliza comandos que permitem ao aprendiz trabalhar com conceitos computacionais importantes para iniciantes em programação, tais como, entrada e saída de dados, tipos de dados, variáveis, estruturas

[1] Disponível em https://scratch.mit.edu/

de controle, operadores e arrays. Além disso, também permite trabalhar com comandos que conferem a natureza multimídia inerente a esta linguagem. Scratch pode ser acessado por meio de qualquer navegador de internet e conta também com uma versão desktop que é disponibilizada gratuitamente para download (SCRATCH, 2018).

4. Condução do Projeto

O projeto apresentado neste trabalho dá continuidade a um projeto anterior, realizado em 2016, onde se levantou o debate sobre o ensino dos conteúdos de lógica de programação para alunos da educação básica, envolvendo pedagogos, professores e alunos do campus. O resultado foi a criação de dois cursos de Lógica de Programação: um fazendo uso da ferramenta Scratch e outro da ferramenta S4A (Scratch for Arduino). Naquele primeiro momento, todos os planos de aulas e materiais pedagógicos foram produzidos pela equipe do projeto, e os cursos foram realizados, envolvendo alunos do 9º ano do ensino fundamental da escola pública (FARIAS et al, 2017).

Nessa segunda etapa do projeto, com o material produzido na etapa anterior, foram realizadas as seguintes atividades:

- Revisão dos planos de aulas e do material pedagógico utilizados no curso de Lógica de Programação através do Scratch.
- Divulgação do curso para o público-alvo e inscrição dos interessados.
- Realização das aulas, análise dos resultados alcançados, pesquisa do veículo de publicação e escrita do artigo a ser publicado.

O curso foi realizado no período entre junho e julho de 2017 e teve 13 alunos inscritos, todos ingressantes no 1º semestre do curso Técnico Subsequente de Informática. As aulas aconteceram no laboratório de informática, contando com carga horária total de 30 horas, dividida em dois encontros semanais com duração de 3 horas, durante 5 semanas, totalizando 10 encontros. As aulas foram ministradas por dois alunos do Curso Superior de Licenciatura em Computação, orientados pela professora da disciplina de Lógica de Programação do campus.

A primeira aula foi realizada no pátio do campus, sem o apoio da ferramenta Scratch, e consistiu na realização de duas dinâmicas: “a fábrica de doces”, que simulava o fabrico de um doce, a partir da definição dos ingredientes necessários, das etapas a serem seguidas e do produto gerado; e a “Caça ao Tesouro”, onde de posse de um mapa ilustrado do campus, os alunos, agrupados em equipes, seguiam instruções que levavam a enigmas a serem solucionados a fim de encontrar pedaços de um algoritmo que deviam ser colocado em ordem. As dinâmicas tinham por objetivo explorar os conceitos de algoritmos, entrada, processamento e saída.

Na segunda aula, os alunos foram solicitados a responderem um formulário contendo perguntas como nome, idade, peso, altura, sexo, etc., sendo possível através das respostas explorar o conceito de tipos de dados. Em seguida foi realizada uma atividade denominada “Caixas-Variáveis”, onde os alunos recebiam um algoritmo e caixas representando as variáveis desse algoritmo. Os alunos deveriam nomear as caixas e atribuir valores às caixas de acordo com a execução do algoritmo.

Ainda na segunda aula os alunos foram apresentados ao ambiente Scratch e a alguns exemplos criados utilizando o ambiente. Foi mostrado aos alunos como criar variáveis no ambiente e atribuir valores a elas. Ao final da aula os alunos foram capazes

de construir um programa simples onde o usuário informava seu nome, que era armazenado na variável padrão "resposta" e o personagem gato dizia o valor armazenado na variável.

A terceira aula abordou os operadores aritméticos, relacionais e lógicos. Foi realizada uma dinâmica denominada "Jogo das Cartas", onde foram distribuídas cartas entre os alunos contendo valores numéricos e operadores aritméticos e lógico-relacionais, sendo que os alunos deveriam montar, usando as cartas recebidas, expressões que resultassem em valores verdadeiros. Para finalizar, os alunos foram incentivados a criar um jogo baseado no conceito da dinâmica utilizada na aula.

A quarta e quinta aulas exploraram a estrutura seqüencial de algoritmos, fazendo forte uso dos operadores aritméticos. Nessas aulas os alunos desenvolveram, utilizando o Scratch, várias calculadoras: calculadora das quatro operações básicas, calculadora do resto da divisão, calculadora de áreas de superfícies diversas (quadrados, triângulos, circunferências) e calculadora de troco.

As estruturas condicionais foram o foco das sexta e sétimas aulas. No decorrer dessas duas aulas os alunos foram estimulados a resolver problemas que envolviam a necessidade de uso dessas estruturas, utilizando o Scratch, como por exemplo: Conhecendo a idade de seu colega, decida se ele é maior ou menor de idade.

Nas últimas 03 aulas do curso foi trabalhado o conceito de estruturas de repetição. Mais uma vez o Scratch foi utilizado para desenvolver problemas que requeriam o uso dessas estruturas, tais como: mostrar a tabuada de multiplicação de um número qualquer.

Dos 13 alunos inscritos, 08 concluíram o curso com frequência igual ou superior a 75% da carga horária de 30h, o que representa 61,5% das inscrições. Ao final do curso, os alunos foram submetidos a um questionário que buscava conhecer a opinião dos alunos sobre o mesmo, tendo sido coletadas 08 respostas.

Quando perguntados sobre as principais dificuldades enfrentadas no decorrer do curso, todos manifestaram dificuldades de entendimento das questões envolvendo cálculos. Os alunos apresentaram desconhecimento dos operadores matemáticos e dificuldades com interpretação de texto, o que confirma os resultados já apontados em outros trabalhos, de que a falta de conhecimento matemático prévio dificulta o desenvolvimento do raciocínio lógico para programação de computadores.

Nenhum aluno se queixou de dificuldades em manusear o Scratch. Foi observado que uma vez que todos os participantes já possuíam contato com o computador, a relação com o Scratch ficou facilitada e os alunos não apresentaram dificuldades para manusear os blocos de comandos propostos pela ferramenta. Durante o curso, a versão web da ferramenta foi apresentada para os alunos e o instalador foi distribuído para práticas em suas residências.

Em relação à prática e persistência, observou-se que os alunos logo no início do curso não aceitavam a idéia de tentar realizar as atividades após um erro, assim perdendo o interesse em solucionar a questão. Isso era perceptível pelo fato de que a maioria dos alunos não costuma resolver problemas lógicos, e boa parte deles não se preocupa em compreender o domínio do problema, muitas vezes dando maior importância às respostas.

Quanto à evasão apresentada durante o curso, foi identificado que a falta de transporte público no período de oferta do curso foi o principal fator determinante para a desistência dos alunos.

Todos os participantes declararam que se sentiram satisfeitos com o ensino e os métodos utilizados pelos ministrantes.

5. Impactos na Disciplina Lógica de Programação

Finalizado o curso de nivelamento, deu-se início às aulas da disciplina Lógica de Programação. Tal disciplina é obrigatória e presencial, com carga horária total de 60h, sendo ofertada aos alunos ingressantes no primeiro semestre do Curso Técnico Subsequente em Informática. O curso é composto por 04 semestres, sendo a disciplina de Lógica de Programação, a primeira das 04 disciplinas de programação ofertadas no decorrer do curso.

A disciplina foi ofertada para os alunos da turma ingressante no segundo semestre de 2017. Haviam 31 alunos matriculados na turma, entretanto, 08 alunos, embora matriculados, nunca assistiram a nenhuma aula.

Os 23 alunos restantes formaram dois grupos: Grupo A: 08 alunos que frequentaram o curso de Introdutório de Programação com Scratch; Grupo B: 15 alunos que não participaram do curso.

A disciplina foi dividida em 04 módulos. No primeiro módulo, foram introduzidos aos alunos os conceitos de lógica e algoritmos, usando ferramentas como descrição narrativa, fluxograma e português estruturado. A ferramenta Visualg foi utilizada para representar os algoritmos através do português estruturado.

No decorrer do primeiro módulo, os alunos resolveram 35 questões propostas sobre lógica e algoritmos, na forma de exercícios. Foram realizadas duas avaliações entre os alunos. A primeira avaliação abordou questões de raciocínio lógico e buscou avaliar o desenvolvimento dos alunos nesse aspecto. Um exemplo de questão utilizada nessa primeira avaliação pode ser visto no Quadro 1.

Quadro 1: Exemplo de Questão de Raciocínio Lógico

As irmãs Luciana, Rosana e Joana, de idades diferentes, possuem, cada uma delas, apenas um cachorro de estimação. Os cães se chamam Rex, Bob e Touro. Um é preto, o outro marrom e o outro branco. A ordem expressa na questão não representa a ordem das cores nem a ordem das donas. Sabe-se que Rex, um cachorro marrom, não é de Joana e pertence à irmã com idade do meio. Rosana, que não é a mais nova, tem um cachorro branco que não se chama Touro. De posse dessas informações, indique o nome de cada irmã, sua idade (mais velha, do meio ou mais nova), o nome e a cor de seu cachorro.

A segunda avaliação foi realizada no laboratório de informática e consistia em usar o Visualg para representar algoritmos para solucionar determinados problemas matemáticos simples. Essa atividade teve por objetivo avaliar a capacidade dos alunos ordenarem logicamente suas ideias a fim de construir uma solução para o problema dado e representar essa solução através do português estruturado. Um exemplo de questão utilizada nessa segunda avaliação pode ser visto no Quadro 2.

Quadro 2: Exemplo de Questão de Representação de Algoritmos usando Visualg

Calcular e mostrar a quantidade de latas de tinta necessárias para pintar uma determinada área, em m^2, fornecida pelo usuário. Considere que 1 litro de tinta cobre 15 m^2, que uma lata de tinta tem 18 litros e que para um resultado melhor serão dadas duas demãos de tinta, ou seja, a área vai receber a pintura duas vezes.

26% dos alunos da turma obteve aprovação no primeiro módulo, sendo que o grupo que frequentou o curso de Scratch teve desempenho melhor que o outro grupo, com 38% de aprovação contra 20% do outro grupo.

No segundo módulo os alunos foram introduzidos à Linguagem C. Nessa etapa os alunos tiveram contato com a sintaxe da linguagem e foram incentivados a desenvolver programas que solucionassem problemas matemáticos simples, que não exigiam o uso de estruturas de seleção nem de repetição.

No decorrer do módulo foram propostas 20 questões em forma de exercícios, e uma avaliação individual foi realizada no laboratório de informática, consistindo em construir um programa em linguagem de programação C para solucionar um dado problema matemático. A avaliação teve por objetivo verificar a capacidade do aluno em representar uma solução algorítmica fazendo uso da linguagem de programação C. Avaliou-se também a compreensão do aluno sobre conceitos como: variáveis e constantes, tipos de dados, operações aritméticas, de atribuição, e de entrada e saída de dados. O Quadro 3 apresenta um exemplo de questão utilizada nessa etapa da disciplina.

Quadro 3: Exemplo de Questão de Construção de Programas em Linguagem C

Sobre o salário bruto de um funcionário são descontados 8% de INSS, 10% de IR e sobre o restante desconta-se 0,5% referente à filiação sindical. Escreva um programa em C que, ao ser fornecido o valor do salário bruto do funcionário, calcule e mostre: 1. O desconto referente ao INSS 2. O desconto referente ao IR 3. O desconto referente à filiação sindical 4. O total de todos os descontos 5. O salário líquido do funcionário.

Houve uma melhora no aproveitamento da turma como um todo no segundo módulo, sendo verificado que 43% da turma obteve êxito. Entretanto, ao analisar os dois grupos separadamente, observou-se que o desempenho do grupo A foi inferior ao do grupo B, indicando que o Scratch não foi eficiente para preparar os alunos para programar em uma linguagem de programação de trabalho. Nesse módulo, apenas 38% dos alunos que participaram do curso de Scratch foram capazes de construir de forma satisfatória um programa em Linguagem C para solucionar o problema proposto, contra 47% dos alunos que não participaram do curso.

O terceiro módulo contemplou as estruturas de decisão IF/ELSE e SWITCH/CASE. Nessa etapa os alunos continuaram a utilizar a linguagem de programação C para solucionar problemas, agora com maior grau de dificuldade. Os alunos deveriam ser capazes de utilizar as estruturas de decisão apresentadas e os operadores lógico-relacionais para compor soluções algorítmicas para os problemas propostos.

Foram propostas 20 questões em forma de exercícios, e uma avaliação individual foi realizada no laboratório de informática, consistindo em construir um programa em linguagem de programação C para solucionar um dado problema envolvendo estruturas de decisão e operadores lógico-relacionais. A avaliação teve por objetivo verificar a capacidade do aluno em representar uma solução algorítmica fazendo uso das estruturas de decisão na linguagem de programação C. Avaliou-se também a compreensão do aluno sobre os operadores lógico-relacionais. O Quadro 4 apresenta um exemplo de questão utilizada nessa etapa da disciplina.

Quadro 4: Exemplo de Questão de Construção de Programas em Linguagem C, exigindo Estruturas de Decisão

Escreva um programa em linguagem de programação C para funcionar em um hemocentro. O programa poderá ser usado por qualquer pessoa interessada em saber se ela preenche os requisitos para ser doadora de sangue.

O programa deverá solicitar a idade do usuário, seu sexo (1 – Masculino ou 2 – Feminino), seu peso e a quantidade de dias decorridos da sua última doação.

O programa deverá responder ao usuário se ele atende ou não aos requisitos para doação de sangue ou plaquetas. Os requisitos são:

- Ter idade entre 18 e 69 anos;
- Pesar no mínimo 50 kg;
- Não ter doado sangue nos últimos 90 dias, para mulheres, e nos últimos 60 dias, para homens.

No terceiro módulo foi registrada a evasão de 02 alunos, sendo 01 do grupo A, e 01 do grupo B. Dessa forma, os grupos passaram a ter 07 e 12 alunos respectivamente. Apesar dessas duas desistências, no terceiro módulo o desempenho da turma como um todo melhorou de forma acentuada, sendo verificado que 86% da turma obteve a nota mínima necessária para aprovação. Curiosamente, o mesmo percentual foi observado nos dois grupos quando analisados individualmente, o que nos leva a suspeitar que a turma como um todo trabalhou de forma mais colaborativa e menos competitiva. Nesse módulo não foi possível observar uma relação direta entre o êxito nas atividades e o contato prévio com o Scratch.

O quarto e último módulo da disciplina abrangeu as estruturas de repetição WHILE, DO/WHILE e FOR. A linguagem de programação C foi também utilizada nessa etapa para solucionar problemas que envolviam a necessidade de repetição de blocos de códigos. Os alunos deveriam ser capazes de utilizar as estruturas de repetição apresentadas para propor programas que solucionassem os problemas apresentados.

Foram propostas 20 questões em forma de exercícios, e uma avaliação individual foi realizada no laboratório de informática, consistindo em construir um programa em linguagem de programação C para solucionar um dado problema envolvendo estruturas de repetição. A avaliação teve por objetivo verificar a capacidade do aluno em representar uma solução algorítmica fazendo uso das estruturas de repetição na linguagem de programação C. O Quadro 5 apresenta um exemplo de questão utilizada nessa etapa da disciplina.

Quadro 5: Exemplo de Questão de Construção de Programas em Linguagem C, exigindo Estruturas de Repetição

Sabendo que uma empresa possui 20 funcionários, faça um programa que leia o salário e o sexo de cada funcionário e informe quantos funcionários ganham mais de R$ 1.000,00 e quantas mulheres ganham acima de R$ 5.000,00. Informe ainda o menor e o maior salário e a média de salário entre homens e mulheres.

No último módulo houve mais uma evasão, dessa vez de um aluno do grupo B; o grupo A então permaneceu com 07 alunos e o grupo B com 13. Analisando o módulo final é possível verificar que o desempenho geral da turma foi inferior ao desempenho observado no módulo anterior, constatando-se uma aprovação de 80% da turma. Ao analisar os grupos individualmente, observou-se que mais uma vez o grupo A teve desempenho menor que o grupo B, sendo observado que 71% dos alunos que participaram do curso de Programação com Scratch foram capazes de construir de forma satisfatória um programa em Linguagem C para solucionar o problema proposto, contra 85% dos alunos que não participaram do curso. Mais uma vez verificou-se que o curso de Scratch não influenciou positivamente para o êxito dos alunos. A Tabela 2 apresenta um resumo dos índices de aprovação em cada módulo e o resultado final da disciplina, separados por grupos.

Tabela 2: Índices de aprovação da turma por etapa

Etapa	Grupo A	Grupo B	Turma
Módulo I	38%	20%	26%
Módulo II	38%	47%	43%
Módulo III	86%	86%	86%
Módulo IV	71%	85%	80%
Resultado Final	71%	77%	75%

6. Considerações Finais

Este trabalho apresentou a condução e os resultados obtidos com a oferta de um curso de Introdução à Lógica de Programação usando Scartch conduzido pela professora de Lógica de Programação e 02 alunos do Curso Superior de Licenciatura em Computação do IFBA, campus Jacobina.

Embora outros trabalhos apontem que o uso do Scratch na introdução de conceitos de programação tenham gerado resultados satisfatórios, do ponto de vista da aprovação dos alunos em disciplinas de Lógica de Programação, nessa experiência isso não foi possível ser comprovado. Ao contrário do que era esperado, o grupo que freqüentou as aulas com o suporte do Scratch apresentou taxas de aprovação sempre igual ou inferior ao grupo de controle, sendo superior apenas na avaliação que abordava os conhecimentos acerca do raciocínio lógico. Esse resultado nos leva a concluir que a ferramenta não foi eficiente no suporte à sintaxe rígida da Linguagem de Programação C, uma vez que a linguagem utilizada no Scratch é muito mais lúdica e visual.

Diante dos resultados obtidos, pretende-se reaplicar o projeto em outras turmas futuras, a fim de se confirmar os resultados obtidos neste trabalho.

Referências

BATISTA, E. et al. Uso do Scratch no ensino de programação em Ponta Porã: das séries iniciais ao ensino superior. In: V Congresso Brasileiro de Informática na Educação/ XXII Workshop de Informática na Escola, 2016, Uberlândia – MG. Disponível em: http://br-ie.org/pub/index.php/wie/article/view/6863/4741.

DIAS, K. e SERRÃO, M. A Linguagem Scratch no Ensino de Programação: Um Relato de Experiência com Alunos Iniciantes do Curso de Licenciatura em Computação. In: XXXIV Congresso da Sociedade Brasileira de Computação, 2014, Brasília, DF. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/wei/2014/0017.pdf.

FERREIRA, A. C. et al. Hello World: relato de experiência de um curso de iniciação à programação. In: V Congresso Brasileiro de Informática na Educação, 2016, Uberlândia – MG. Disponível em: http://www.br-ie.org/pub/index.php/wcbie/article/view/7056/4930.

FARIAS, Carina Machado de et al. Uso de estratégias alternativas para o ensino de lógica de programação: relato de experiência em Jacobina, Bahia. In: XVII ESCOLA REGIONAL DE COMPUTAÇÃO BAHIA ALAGOAS SERGIPE (ERBASE) / WORKSHOP DE EDUCAÇÃO E INFORMÁTICA BAHIA-ALAGOAS-SERGIPE (WEIBASE), 2017, Cruz das Almas - BA. Disponível em: <https://drive.google.com/drive/folders/1UZOpPF0YrPQHLgEZflxYfFAaMWnKbo87>. Acesso em: 20 Nov. 2017.

FRIEDRICH, R. V. et al. Proposta metodológica para a inserção ao ensino de lógica de programação com logo e lego mindstorms. In XXIII Simpósio Brasileiro de Informática na Educação, 2012, Rio de Janeiro - RJ. **Anais do Simpósio Brasileiro de Informática na Educação**. Porto Alegre: SBC, 2012. Vol. 23, No. 1. Disponível em: http://www.br-ie.org/pub/index.php/sbie/article/view/1762/1523.

GOMES, W. et al. Incentivando meninas do ensino médio à area de Ciência da Computação usando o Scratch como ferramenta. In: III Congresso de Informática na Educação/XX Workshop de Informática na Escola, 2014, Dourados, MS. Disponível em: http://br-ie.org/pub/index.php/wie/article/view/3104/2612.

GUEDES, E. Um Estudo Observacional sobre a Disciplina Introdutória de Programação. In: III Congresso de Informática na Educação/XX Workshop de Informática na Escola, 2014, Dourados, MS. Disponível em: http://br-ie.org/pub/index.php/wie/article/view/3140/2648.

OLIVEIRA, M. et al. Ensino de lógica de programação no ensino fundamental utilizando o Scratch. In: XXXIV Congresso da Sociedade Brasileira de Computação, 2014, Brasília, DF. Disponível em: http://csbc2014.cic.unb.br/index.php/anais-menu.

SILVA, G., SOUZA, J. e SILVA, L. Aplicação da Ferramenta Scratch para o Aprendizado de Programação no Ensino Fundamental I. In: V Congresso Brasileiro de Informática na Educação, 2016, Uberlândia – MG. Disponível em: http://br-ie.org/pub/index.php/wcbie/article/view/7054.

SOUZA, S. e CASTRO, T. Investigação em programação com Scratch para crianças: uma revisão sistemática da literatura. In: V Congresso Brasileiro de Informática na Educação, 2016, Uberlândia – MG. Disponível em: http://www.br-ie.org/pub/index.php/wcbie/article/view/7033.

Análise da Motivação em um Estudo Integrado de Programação Baseado em PBL

Ayala L. Ribeiro, Roberto A. Bittencourt, Bianca L. Santana

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

{ayalaedavi,biancasantana.ls}@gmail.com, roberto@uefs.br

Abstract. *The motivation of students majoring in computing in programming courses is a relevant factor that may contribute to learning success. This work analyzes, through the ARCS framework, the motivation of students in an object-oriented programming integrated course that uses Problem-Based Learning (PBL). The levels of motivation found vary during the course, being higher in the beginning, decreasing during the process, but increasing at the end of the course. We conclude that the adequate design of the problems to be solved by students may positively or negatively affect the motivation, and that the levels of motivation found are affected by students' intrinsic motivation.*

Resumo. *A motivação de estudantes da área de computação em disciplinas de programação é um fator relevante que pode contribuir para o sucesso no aprendizado. Este trabalho analisa, através do framework ARCS, a motivação dos estudantes em um período letivo do estudo integrado de programação orientada a objetos que utiliza a abordagem de Aprendizagem Baseada em Problemas (PBL). Os níveis de motivação encontrados variam durante o decorrer da disciplina, sendo mais altos no início, reduzindo-se no decorrer do processo, mas aumentando ao final da disciplina. Conclui-se que o design adequado dos problemas trabalhados pelos estudantes pode afetar positiva ou negativamente a motivação e que os níveis de motivação encontrados são afetados pela motivação intrínseca dos estudantes.*

1. Introdução

Desmotivação, reprovação e evasão são problemas constantes enfrentados por cursos da área de computação, especialmente nas disciplinas de Algoritmos e Programação. Estas disciplinas costumam ter altos índices de evasão e reprovação, dificultando ou impedindo a continuidade dos alunos no curso [Bennedsen and Caspersen 2007]. Dentre possıveis fatores que contribuem para esta situação, pode-se elencar: i) falta de capacidade de abstração e raciocínio lógico para desenvolver soluções algorítmicas; ii) falta de motivação do estudante, que, muitas vezes, encara a disciplina como um grande obstáculo a ser superado; e iii) abordagem de ensino instrucionista, que pode não despertar o interesse do estudante [Jenkins 2002].

É relativamente comum introduzir programação através do paradigma imperativo e, posteriormente, apresentar o paradigma orientado a objetos nas disciplinas de programação de cursos de computação. Esta transição cria um problema

adicional pois provoca um conflito cognitivo nos aprendizes, geralmente demorado de resolver [Bittencourt et al. 2013, Jenkins 2002]. Associado aos fatores anteriores, esta organização curricular aumenta a complexidade na aquisição de habilidades de programação, gerando desmotivação e, consequentemente dificultando a aprendizagem e a retenção do conhecimento.

A programação orientada a objetos (POO) tornou-se, nos últimos anos, o paradigma de programação mais influente, sendo amplamente utilizada na educação e na indústria. No entanto, aprender POO não é fácil [Kölling 1999]. As dificuldades podem ser causadas pela complexidade dos conceitos a serem aprendidos em um curto período de tempo, pela complexidade intrínseca destas linguagens e dos ambientes de desenvolvimento profissionais, agravadas pelo uso de metodologias de aprendizagem centradas no professor.

A utilização de metodologias de aprendizagem ativa é uma alternativa proposta pela comunidade para atacar as dificuldades de aprendizagem de programação. É o caso, por exemplo da Aprendizagem Baseada em Problemas (PBL, do inglês, Problem-Based Learning), uma abordagem instrucional centrada no estudante onde parte importante do estudo ocorre em pequenos grupos que se reúnem para resolver problemas propostos que desencadeiam e motivam o processo de aprendizagem [Santos et al. 2007]. A aprendizagem é autodirigida, baseada na reflexão e no fomento das questões envolvidas. O uso de PBL contribui para aquisição de habilidades como autonomia, iniciativa, comunicação, pensamento crítico, resolução de problemas, trabalho em grupo, além da retenção do conhecimento e sua aplicação em diferentes contextos [Delisle 1997].

Alguns trabalhos relatam a utilização da metodologia PBL como alternativa para o ensino de programação orientada a objetos [Bittencourt et al. 2013, Angelo et al. 2014]. Mas, embora estes trabalhos relatem as experiências em detalhes, ainda não foi feita uma avaliação científica aprofundada do uso de uma abordagem como PBL no ensino de POO, especialmente levando em conta questões de motivação.

O objetivo deste trabalho é avaliar a motivação dos estudantes em uma abordagem de ensino-aprendizagem de programação orientada a objetos no curso de Engenharia de Computação da Universidade Estadual de Feira de Santana (UEFS), no Estudo Integrado de Programação do segundo semestre do curso. A avaliação foi realizada através de uma metodologia de pesquisa quantitativa e uma abordagem de estudo de caso.

A partir do objetivo acima, procuramos responder às seguintes questões de pesquisa:

1. Como a abordagem utilizada influencia a motivação dos estudantes?
2. Quais os resultados de motivação nos diversos componentes do estudo integrado e como se comparam estes resultados?

Os resultados indicam que os níveis de satisfação são relativamente altos durante o semestre letivo. Os níveis de confiança tiveram as menores medianas. Os níveis de atenção e satisfação foram relativamente altos em três dos quatro problemas PBL aplicados no estudo integrado. O cuidado na elaboração do problema, a escolha do domínio do problema, a complexidade do problema, o excesso de conceitos e a inserção da novidades são questões que interferem diretamente na motivação dos estudantes.

2. Fundamentação Teórica

Esta seção apresenta os conceitos essenciais para a compreensão deste trabalho.

2.1. Motivação

Keller (1987) define motivação como aquela que explica a direção e a magnitude do comportamento, ou, em outras palavras, explica quais os objetivos que as pessoas escolhem perseguir e quão ativamente ou intensamente os perseguem. Ray (1964) descreve a motivação fazendo um exame cuidadoso da palavra (motivo) e de seu uso, e conclui que a motivação deverá fazer referência a três componentes: o comportamento de um sujeito; a condição biológica interna relacionada; e a circunstância externa relacionada.

Jenkins (2001) alega que para inspirar os estudantes através de uma instrução verdadeiramente motivacional na aprendizagem de programação, a solução é maximizar os efeitos positivos de cada um desses fatores. E para isso é necessário reconhecê-los. Ele menciona cinco tipos de motivação: extrínseca, intrínseca, social, de realização e nula. Na motivação extrínseca, o fator motivacional é a carreira e as recompensas associadas que resultarão da conclusão bem-sucedida do curso. Na motivação intrínseca, o principal fator motivacional é um interesse profundo na computação (ou especificamente em programação) para seu próprio bem-estar. Na motivação social, o fator motivacional é o desejo de agradar a um terceiro cuja opinião é importante. Na motivação de realização, o fator motivacional é "fazer bem" para a satisfação pessoal. Uma quinta categoria corresponde à "motivação nula", que se relaciona com casos que não se encaixam nas categorias previamente descritas.

Baseando-se na importância da motivação na aprendizagem, Keller (1987) desenvolveu o modelo ARCS da motivação, que se propõe a fornecer estratégias para auxiliar a reconhecer e ajudar a resolver problemas motivacionais dos estudantes. O modelo analisa as necessidades motivacionais dos estudantes através de quatro categorias: Atenção, Relevância, Confiança e Satisfação.

2.2. Aprendizagem de programação orientada a objetos – POO

O ensino de POO como segundo paradigma é muito discutido na literatura. As dificuldades podem estar associadas a aspectos como a complexidade natural destas linguagens ou a complexidade dos ambientes de desenvolvimento profissionais. Alguns autores afirmam que a introdução de paradigmas como orientação a objetos na disciplina introdutória de programação, que é ministrada no início dos cursos de computação, não fornece evidências significativas de facilitar a aprendizagem [Burton and Bruhn 2003]. Em contraposição, Kolling (1999) sugere que os conceitos de orientação a objetos devem ser ensinados desde o início, argumentando que as dificuldades estão nas ferramentas utilizadas para o ensino e não no paradigma em si.

2.3. Aprendizagem Baseada em Problemas – PBL

A Aprendizagem Baseada em Problemas (PBL) é uma metodologia ativa desenvolvida por Howard Barrows na Universidade de McMaster, no Canadá [Barrows and Tamblyn 1980]. Inicialmente idealizada para a área de saúde, esta metodologia ganhou aceitação e está se tornando cada vez mais presente em uma variedade de disciplinas no ensino superior. Problemas inspirados no mundo real são o ponto de

partida para os estudantes, responsáveis pela construção de suas próprias aprendizagens. O professor atua como facilitador e é responsável pela concepção do problema a ser solucionado. O estudante tem autonomia para analisar e trilhar os possíveis caminhos em direção à solução do problema [Delisle 1997].

Em PBL, não só os estudantes são constantemente incentivados a aprender, mas também a desempenhar um papel ativo no processo de construção da aprendizagem [Cintra and Bittencourt 2015]. Um problema em PBL é um gatilho para motivar o estudo. Geralmente, segue-se um ciclo de aprendizagem, repetido enquanto durar o problema: 1) Os estudantes são apresentados a um problema antes de qualquer preparação ou estudo; 2) Os estudantes se reúnem em grupos para organizar ideias e recordar conhecimentos anteriores relacionados ao problema; 3) Através da discussão, os estudantes colocam questões, conhecidas como questões de aprendizagem, que tratam de aspectos do problema que não entendem; 4) As questões de aprendizagem são classificadas em ordem de importância, e os alunos definem metas de aprendizagem para estudo independente e em grupo.

2.4. PBL na Educação em Computação

Uma revisão sistemática com o objetivo investigar como PBL está sendo utilizada nos currículos da área de computação encontrou 63 artigos em disciplinas de vários cursos da área [O'Grady 2012]. Nestes cursos, há uma grande variedade de disciplinas que usam PBL como abordagem de ensino: engenharia de software, programação de computadores, qualidade de software e sistemas operacionais, dentre outras. Concluem constatando que a penetração do PBL nos currículos de computação ainda é superficial.

No Brasil, a utilização de PBL na área de computação ainda é muito restrita. PBL é empregada em seis componentes curriculares no curso de Engenharia de Software da Universidade Federal do Pampa (UNIPAMPA). Esses componentes integram, de modo interdisciplinar e transversal, diferentes conteúdos na abordagem de uma situação-problema que se aproxima da realidade profissional [Cheiran et al. 2017]. Outra instituição que introduziu a abordagem com PBL integrada ao currículo durante todo o curso é a Universidade Estadual de Feira de Santana (UEFS), que adotou PBL no curso de Engenharia da Computação desde a sua criação [Santos et al. 2007].

Bittencourt et al. (2013) relatam uma experiência de integração das disciplinas de programação orientada a objetos, estruturas de dados e projeto de sistemas em um estudo integrado semestral com dez horas semanais, utilizando uma metodologia de aprendizagem baseada em problemas e projetos. A experiência foi aplicada em dois semestres letivos e as principais lições aprendidas foram: a integração de conhecimentos possibilita experiências mais autênticas e práticas de produção de software mais disciplinadas; a metodologia PBL permite adquirir competências mais amplas de comunicação, trabalho em equipe e autodidatismo; problemas de manutenção de software podem reduzir a motivação por acúmulo de deficiências e devem ser evitados; a dosagem de novos conceitos nos problemas propostos deve respeitar um processo gradual e a capacidade de assimilação dos estudantes.

3. Metodologia

Nesta seção são apresentados o cenário educacional, o estudo de caso e os procedimentos de coleta e análise de dados.

3.1. Cenário

O curso de Engenharia da Computação da Universidade Estadual de Feira de Santana (UEFS) tem adotado PBL desde sua criação em 2003. Este curso é caracterizado pela integração e interdependência entre componentes curriculares que agrupam disciplinas com conteúdos relacionados em um mesmo período letivo, compartilhando trabalhos, desafios e oportunidades de aprendizado, evidenciadas particularmente pelos componentes curriculares denominados Estudos Integrados [Bittencourt and Figueiredo 2003]. O Estudo Integrado (EI) objetiva ser um componente integrador sobre certo tema e é organizado em módulos. Durante o estudo integrado, o estudante é apresentado a certo tema ou problemas abrangentes e, para resolver os problemas, torna-se necessário adquirir novos conhecimentos, os quais são agrupados em módulos.

Os conceitos de programação estão presentes em dois estudos integrados desse curso. O EI de Algoritmos é um componente oferecido no primeiro semestre, que faz um estudo introdutório integrando as ideias de algoritmos, estruturas de dados básicas (arrays e registros) e programação estruturada em uma linguagem imperativa. No segundo semestre, o EI de Programação integra a programação orientada a objetos, algoritmos e estruturas de dados avançadas e projeto de sistemas [Bittencourt et al. 2013]. O objetivo geral do EI de Programação é que o estudante seja capaz de projetar e desenvolver software orientado a objetos, utilizando apropriadamente algoritmos e estruturas de dados, com domínio dos fundamentos subjacentes às metodologias e ferramentas utilizadas.

3.2. Estudo de caso

O estudo de caso foi conduzido ao longo do segundo semestre de 2017. Avaliamos o Estudo Integrado (EI) de Programação do curso de Engenharia da Computação da UEFS, componente curricular formado pelos módulos: Algoritmos e Programação II, Estrutura de Dados, Projeto de Sistemas e o Módulo Integrador (MI) de Programação. As turmas eram heterogêneas, compostas por 25 estudantes novatos e veteranos. Para atender às questões éticas e preservando o direito ao anonimato dos participantes da pesquisa, um Termo de Consentimento Livre e Esclarecido (TCLE) foi assinado pelos estudantes que desejaram participar. Por limitações de espaço do artigo, o planejamento do MI de Programação[1] e os problemas[2] são apresentados em outros locais.

O estudo de caso foi iniciado com a aplicação de um questionário pré-intervenção para identificar os dados demográficos dos estudantes, bem como suas impressões em relação à computação e, mais especificamente, à programação em si. No MI de Programação, onde ocorrem as sessões tutoriais de PBL, um questionário foi aplicado ao final de cada um dos quatro problemas propostos para avaliar a motivação dos estudantes e suas percepções sobre a aprendizagem; a seção de motivação do questionário replica o instrumento IMMS – Instructional Materials Motivation Survey [Keller 2010]. IMMS é um instrumento projetado para medir reações a materiais de instrução autodirigidos, embora possa ser adaptado para situações presenciais com foco nos materiais instrucionais. Das 36 questões originais do IMMS, três questões da categoria atenção explicitamente relacionadas a materiais instrucionais puramente textuais foram suprimidas. Os questionários foram analisados através de estatística descritiva e inferencial. A significância dos testes foi determinada considerando o nível de significância de $0,05$.

[1] `https://sites.google.com/site/wei2018programacao`
[2] `http://sites.ecomp.uefs.br/mip-20172/home/tutorial`

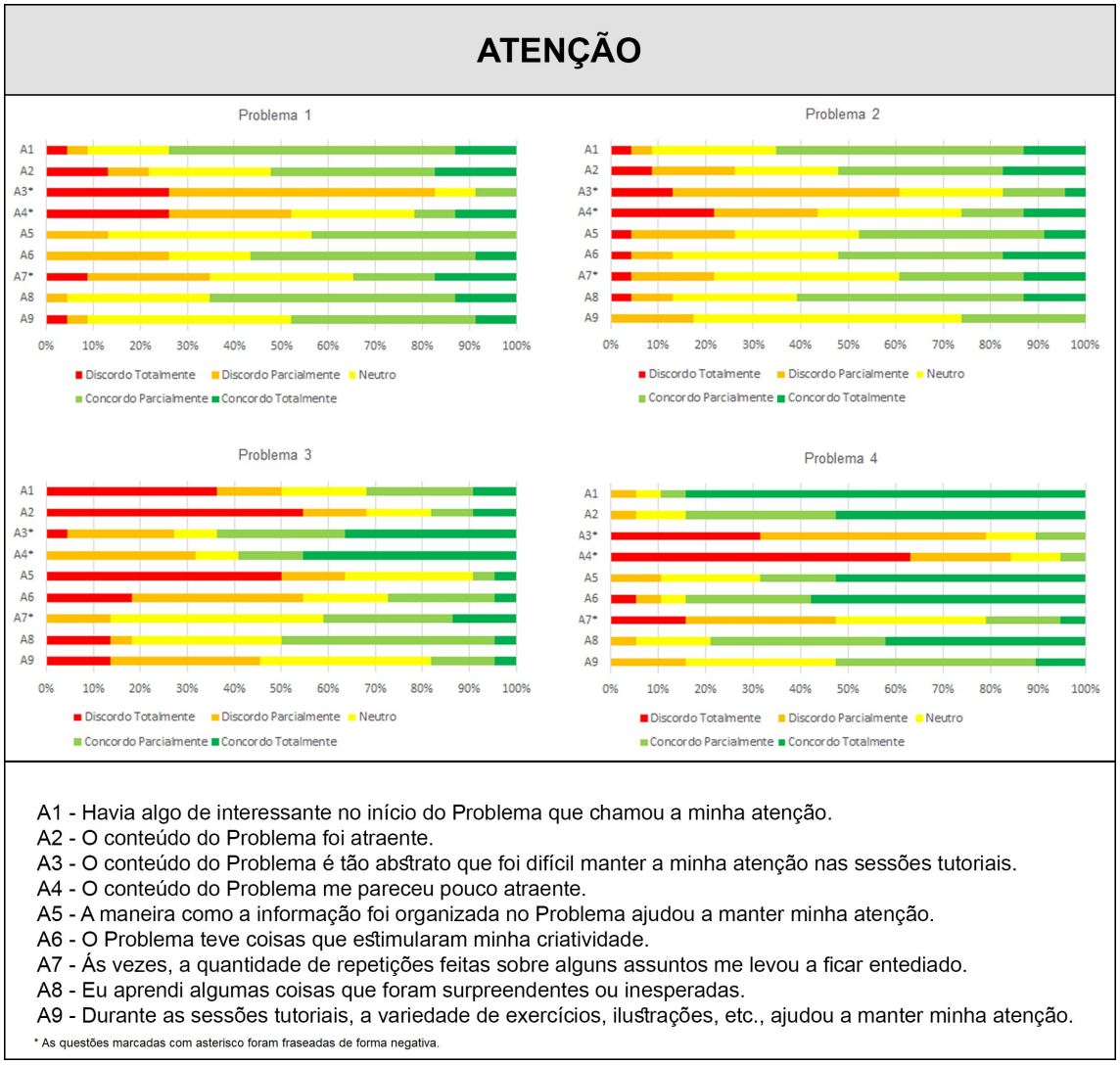


Figura 1. Resultados para a categoria Atenção nos Problemas

4. Resultados

Reunimos os resultados dos questionários IMMS para cada um dos quatro problemas propostos a partir das categorias de Atenção, Relevância, Confiança e Satisfação do modelo ARCS e os apresentamos a seguir. As respostas aos questionários são baseadas em uma escala de Likert de cinco níveis. Para facilitar a interpretação, consideramos que houve concordância quando as respostas foram Concordo Parcialmente ou Totalmente e discordância quando as respostas foram Discordo Parcialmente ou Totalmente.

Para cada categoria do ARCS, produzimos um escore a partir da soma dos valores das respostas de cada questão, convertendo a escala nominal para uma escala numérica variando de 1 (Discordo Totalmente) a 5 (Concordo Totalmente), e tratando as questões fraseadas na forma negativa com uma escala invertida de 5 a 1. Os escores foram normalizados entre 1 e 5, dividindo cada soma pelo número de questões de cada categoria. Confirmamos a normalidade de todos os escores através do teste de aderência de Kolmogorov-Smirnov.

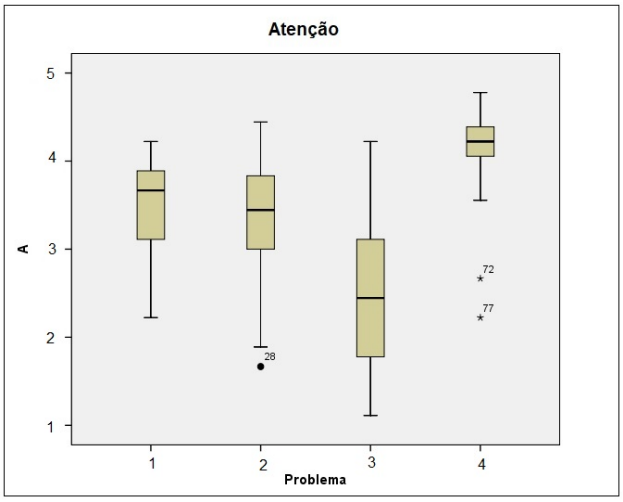


Figura 2. Box-Plot dos escores de Atenção nos Problemas

4.1. Atenção

A Figura 1 apresenta os resultados sobre a categoria Atenção nos problemas do MI de Programação. Em todos os problemas, os estudantes afirmaram que aprenderam coisas surpreendentes ou inesperadas (A8 – concordância de 65% em P1, 65% em P2, 50% em P3, 79% em P4). A maior parte deles concordou que havia coisas interessantes que chamavam atenção (A1 – 74% em P1, 65% em P2, 89% em P4), e achou o conteúdo dos problemas atraente (A2 – 52% em P1, 52% em P2, 84% em P4). Entretanto, a maioria dos estudantes considerou o Problema 3 pouco atraente (A4 – 68%), tampouco gostou da maneira como a informação foi organizada (A5 – 65%), além de considerar que o conteúdo foi abstrato, dificultando a atenção nas sessões tutoriais (A3 — concordância de 64%). O Problema 4 obteve os melhores resultados. Uma parcela significativa de estudantes concordou que havia coisas que estimulavam a criatividade (A6 – 84%), não achou o conteúdo abstrato (A3 – discordância de 79%), e julgou que a maneira como a informação foi organizada contribuiu para manter a atenção (A5 – concordância de 69%).

A Figura 2 traz os box-plots do escore da categoria Atenção nos problemas. Observamos que a mediana da atenção obteve resultados positivos nos Problemas 1 e 2 (escores entre 3 e 4), bastante positivos no Problema 4 (escore acima de 4), e negativos no Problema 3 (entre 2 e 3). Sobre a dispersão dos dados, nos Problemas 1 e 2, notamos uma dispersão menor que a do Problema 3 e maior que a do Problema 4. Assim, os escores sofreram menos variações nos Problemas 1 e 2, o que é ainda mais evidente no Problema 4. A dispersão dos escores é maior no Problema 3.

Utilizamos a análise de variância (ANOVA) para testar se o escore de Atenção varia significativamente entre os Problemas, o que foi confirmado ($F = 18,01$, $p < 0,001$). O resultado evidencia que a distribuição de pelo menos um dos grupos difere das demais, mas não indica entre quais grupos a diferença é significativa. Para se determinar quais pares de escores são diferentes entre si, adotamos o Teste *post-hoc* de Tukey. Houve diferenças estatisticamente significativas (valor-p $<$ 0,05) entre os Problemas: a) 1 e 3; b) 1 e 4; c) 2 e 3; d) 2 e 4; e) 3 e 4. Os níveis de atenção foram mais altos no Problema 4, seguidos pelos Problemas 1 e 2 e mais baixos no Problema 3.

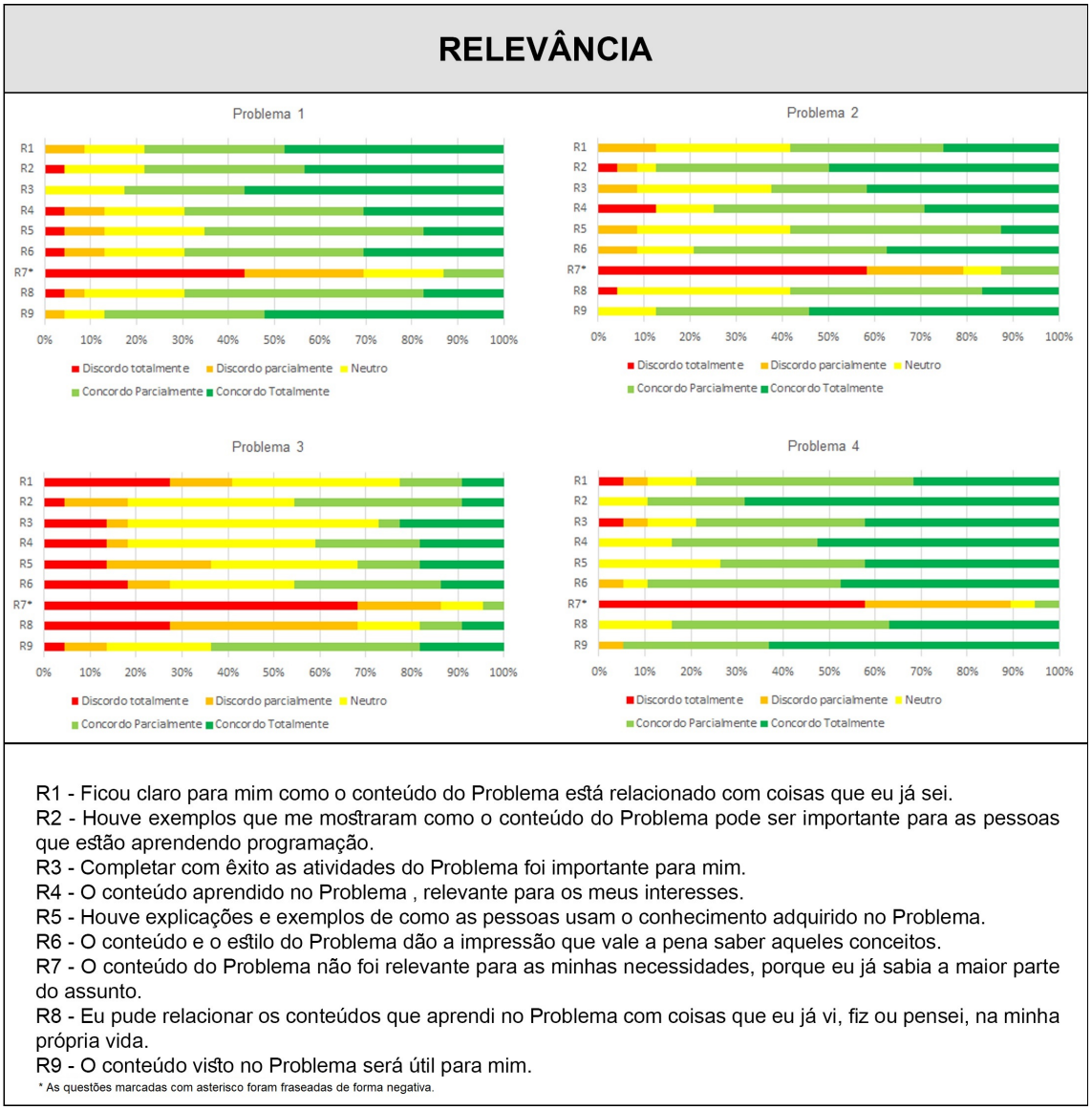


Figura 3. Resultados para a categoria Relevância nos Problemas

4.2. Relevância

Na Figura 3, a categoria Relevância nos problemas é ilustrada. Estudantes reconheceram a utilidade do conteúdo adotado em todos os problemas (R9 – concordância de 87% em P1, 88% em P2, 64% em P3, 95% em P4). Constataram a importância dos conteúdos, principalmente para aqueles que estão aprendendo a programar (R4 – 78% em P1, 88% em P2, 89% em P4). A percepção da relevância fica menos evidente no Problema 3, pois a maioria não relacionou os conteúdos aprendidos com coisas que já foram feitas, vistas ou pensadas (R8 – discordância de 68% em P3) e considerou que o conteúdo não foi relevante para atender suas necessidades (R7 – 86% em P3). Em contrapartida, no Problema 4, a percepção da relevância é mais significativa pois uma parcela expressiva dos estudantes reconheceu os benefícios de aprender os conceitos com o conteúdo e o estilo usados (R6 – concordância de 89% em P4), e concordaram (R8 – 84% em P4) ao relacionar os conteúdos aprendidos com coisas que já foram vistas, feitas ou pensadas.

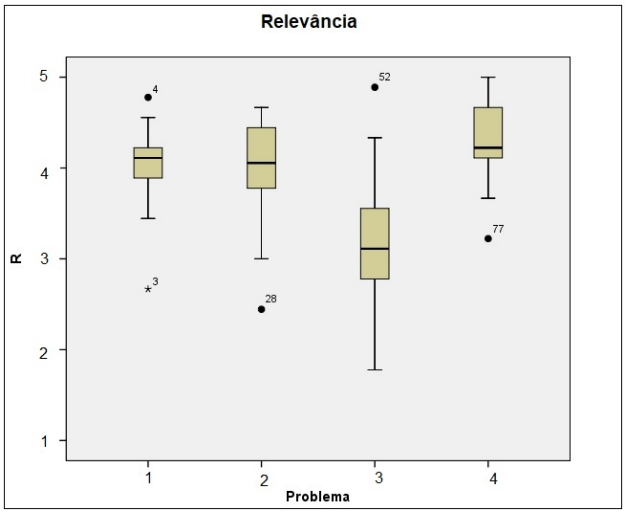


Figura 4. Box-Plot dos escores de Relevância nos Problemas

A Figura 4 exibe o box-plot do escore da categoria Relevância nos problemas. A mediana da relevância é maior nos problemas 1, 2 e 4, menor no problema 3, porém positiva em todos os problemas. Em relação à dispersão dos dados, observamos que nenhum dos problemas teve dispersão expressiva.

Utilizamos ANOVA para testar se a Relevância sofre mudanças significativas entre os Problemas, confirmado com $F = 13,67$, $p < 0,001$. Utilizamos o Teste *post hoc* de Tukey para identificação das diferenças específicas entre os pares de escores. O resultado demonstrou diferenças significativas entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes entenderam que todos os problemas foram relevantes, porém, consideraram mais relevantes os Problemas 1, 2 e 4.

4.3. Confiança

A Figura 5 apresenta os resultados sobre a categoria Confiança nos problemas do MI de Programação. Os estudantes concordaram que após as primeiras sessões tutoriais sentiram-se mais confiantes do que deviam aprender (C3 – 74% em P1, 71% em P2, 63% em P4) e perceberam que eram capazes de passar na avaliação (C7 – 74% em P1, 74% em P2, 69%e em P4). Observamos que o Problema 3 não despertou a confiança dos estudantes da mesma maneira que os outros problemas, visto que a maioria qualificou os assuntos como mais difíceis do que gostariam que fossem (C2 – 91%), não conseguiram compreender como algumas coisas eram feitas (C8 – 86%), bem como não se sentiram confiantes do que realmente deveriam aprender (C3 – discordância de 77%).

Na Figura 6, está representado o box-plot do escore da dimensão Confiança nos problemas. Observamos que a mediana da confiança alcançou resultados positivos nos problemas 1, 2 e 4. O problema 3 apresentou mediana negativa (abaixo de 3). A dispersão dos dados é homogênea nos Problemas 1 e 2. Os Problemas 3 e 4 apresentam dispersão maior, indicando uma maior variabilidade nos escores.

Usamos ANOVA para testar se a Confiança varia significativamente entre os Problemas, o que foi confirmado com $F = 16,84$, $p < 0,001$. O Teste *post-hoc* de Tukey revela que os resultados apresentaram diferenças significativas entres os grupos, cons-

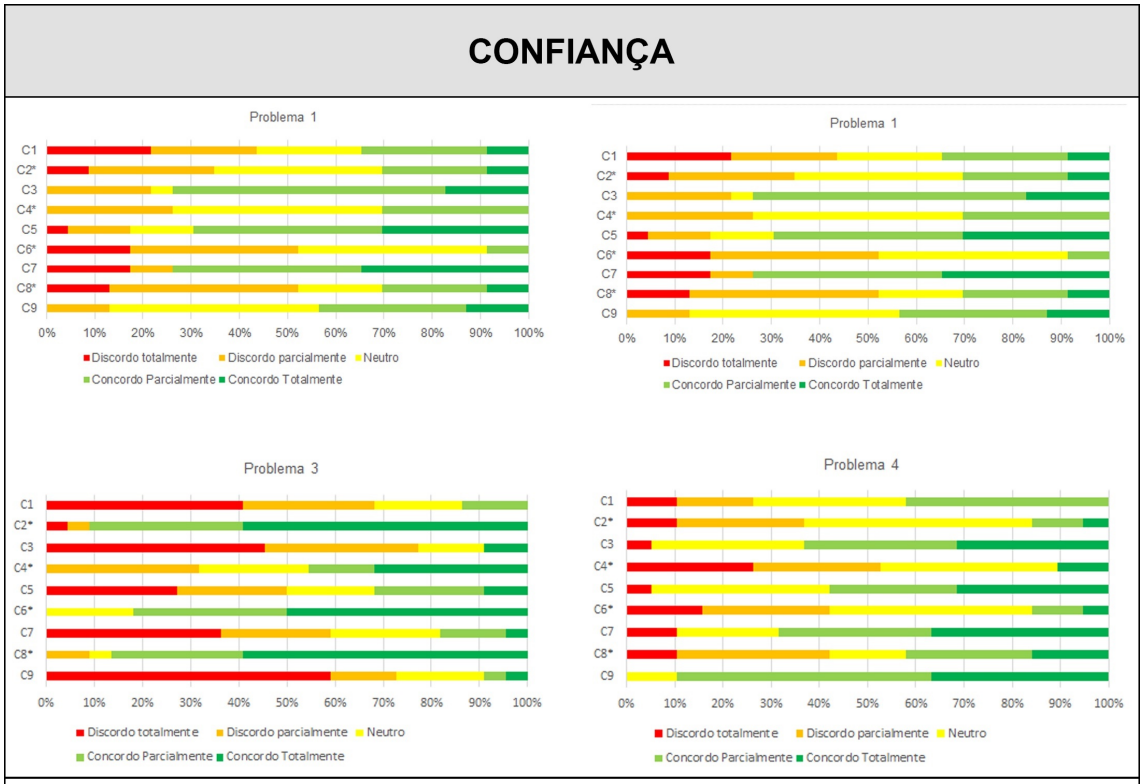


Figura 5. Resultados para a categoria Confiança nos Problemas

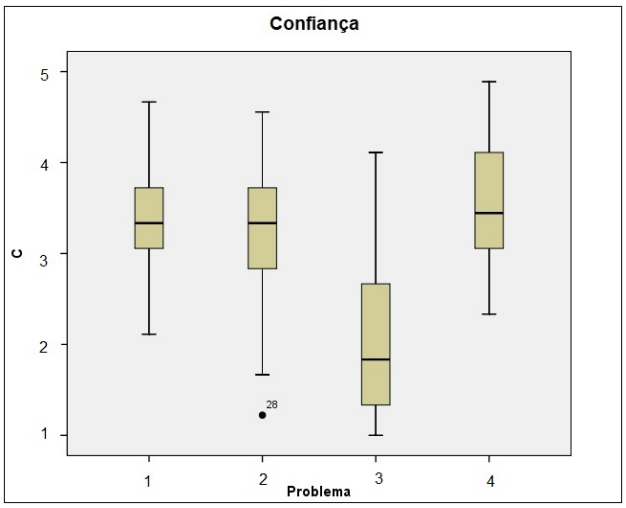


Figura 6. Box-Plot dos escores de Confiança nos Problemas

tatado entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes permaneceram confiantes nos Problemas 1, 2 e 4, contudo não se mantiveram assim no Problema 3.

4.4. Satisfação

Na Figura 7, exibimos os resultados sobre a categoria Satisfação nos Problemas do MI de Programação. A maioria dos estudantes afirmou que ao completar os exercícios sentiram-se uma sensação gratificante de realização (S1 – 74% em P1, 71% em P2, 84% em P4), e que se sentiram bem ao concluir o desafio com êxito (S5 – 61% em P1, 71% em P2, 63% em P4). Todavia, verificamos que a satisfação no Problema 3 demonstrou resultados insatisfatórios. Boa parte dos estudantes discordou que foi um prazer estudar a metodologia utilizada (S6 – 64%), e também discordou que gostava de estudar programação (S3 – 59%), assim como de saber mais sobre o conteúdo (S2 – 55%). Por outro lado, o Problema 4 se destacou no quesito satisfação, quando os estudantes relataram o desejo de saber mais sobre os conteúdos abordados (S2 – 79%), e afirmaram que realmente gostaram de estudar programação (S3 – 79%).

A Figura 8 apresenta o box-plot do escore da categoria Satisfação nos problemas. Verificamos que a mediana da satisfação obteve resultados bastante positivos nos Problemas 1, 2 e 4. O Problema 3 apresentou mediana negativa. Em relação à dispersão dos dados, nos Problemas 1, 2 e 3 verificamos uma dispersão maior que a do Problema 4. Assim, os escores sofreram mais variações nos Problemas 1, 2 e 3. A dispersão dos escores é menor no Problema 4, logo, os dados são mais homogêneos.

Utilizamos ANOVA para testar se a Satisfação difere significativamente entre os Problemas, o que foi confirmado com $F = 9,533$, $p < 0,001$. Utilizamos o Teste *post hoc* de Tukey para identificação das diferenças específicas entre os pares de escores. O resultado demonstrou diferenças significativas entre os Problemas: a) 1 e 3; b) 2 e 3; c) 3 e 4. Os estudantes alcançaram níveis de satisfação mais altos nos Problemas 1, 2 e 4 e menores no Problema 3.

5. Discussão

Nesta seção, discutimos a motivação a partir da perspectiva dos problemas realizados, das categorias do modelo ARCS, além sintetizar as lições aprendidas.

5.1. Diferenças entre os problemas

As categorias Atenção, Confiança, Relevância e Satisfação, de modo geral, apresentaram resultados razoáveis nos Problemas 1 e 2, negativos no Problema 3, e mais positivos no Problema 4. As hipóteses que os justificam são baseadas em evidências coletadas qualitativamente a partir das nossas observações, percepções e do *feedback* dos estudantes.

As principais características da motivação intrínseca são: a satisfação, o interesse, o desafio, a curiosidade e a novidade [Keller 1987]. Nossos resultados sugerem que a novidade influenciou em melhores níveis de Atenção, Relevância, Confiança e Satisfação nos Problemas 1, 2 e 4, e piores no Problema 3. Observamos que no Problema 1, houve a introdução de conceitos de POO, gerando uma novidade. No Problema 2, inseriu-se o conceito de testes, outra novidade considerada interessante pelos estudantes, pois agora teriam uma forma diferente de trabalhar. O Problema 4 trouxe o conceito de interface gráfica, proporcionando aos estudantes uma experiência estimulante, permitindo o *feedback* visual imediato. O Problema 3 não apresentava muitos conceitos novos, com exceção do conceito de árvore binária, os conceitos de POO eram similares e a dificuldade do problema era similar à dos anteriores.

Por outro lado, o domínio do problema, com a descrição de um cenário que apresente um contexto problematizador, escolhido a partir de um contexto real, torna o problema mais atraente e motivador. O domínio nos Problemas 1, 2 e 4 proporcionou uma identificação imediata dos estudantes. Principalmente no Problema 4, onde o domínio do problema tratava do desenvolvimento de um aplicativo similar a um já existente no mercado, que permitia o planejamento de viagens. Todavia, o domínio do Problema 3 não obteve a mesma aceitação. O domínio era desconhecido para grande maioria dos estudantes, tratando de um sistema gerenciador de carteira de ações. A maioria dos estudantes não considerou o problema relevante, gerando uma grande dificuldade motivacional.

Finalmente, o problema deve ter a complexidade ideal. O problema não pode ser complexo demais, que impeça o entendimento dos conceitos, nem simples demais que impossibilite a reflexão e a discussão acerca do que deve ser aprendido. O problema deve ter a clareza e o tamanho necessários para incentivar os estudantes no desenvolvimento da solução do problema. As observações sugerem que a complexidade existente no Problema 3 influenciou nos baixos índices motivacionais em todas as dimensões do ARCS. Os estudantes consideraram o problema abstrato e confuso. O Problema 4 obteve a melhor avaliação dentre os problemas. Os estudantes acharam-no bem estruturado, com informações claras e objetivo bem definido, além de o considerarem mais intuitivo, por fazer alusão a uma aplicação existente.

5.2. Diferenças entre as dimensões do Modelo ARCS

Em respeito à Atenção, Keller (1987) argumenta que mesmo os melhores programas não obterão resultados positivos caso os estudantes não estejam motivados para aprender. Para isso, é necessário estabelecer um equilíbrio nas atividades do aprendiz que permita manter sua atenção. Portanto, é importante que se utilizem estratégias que incluam a variação de

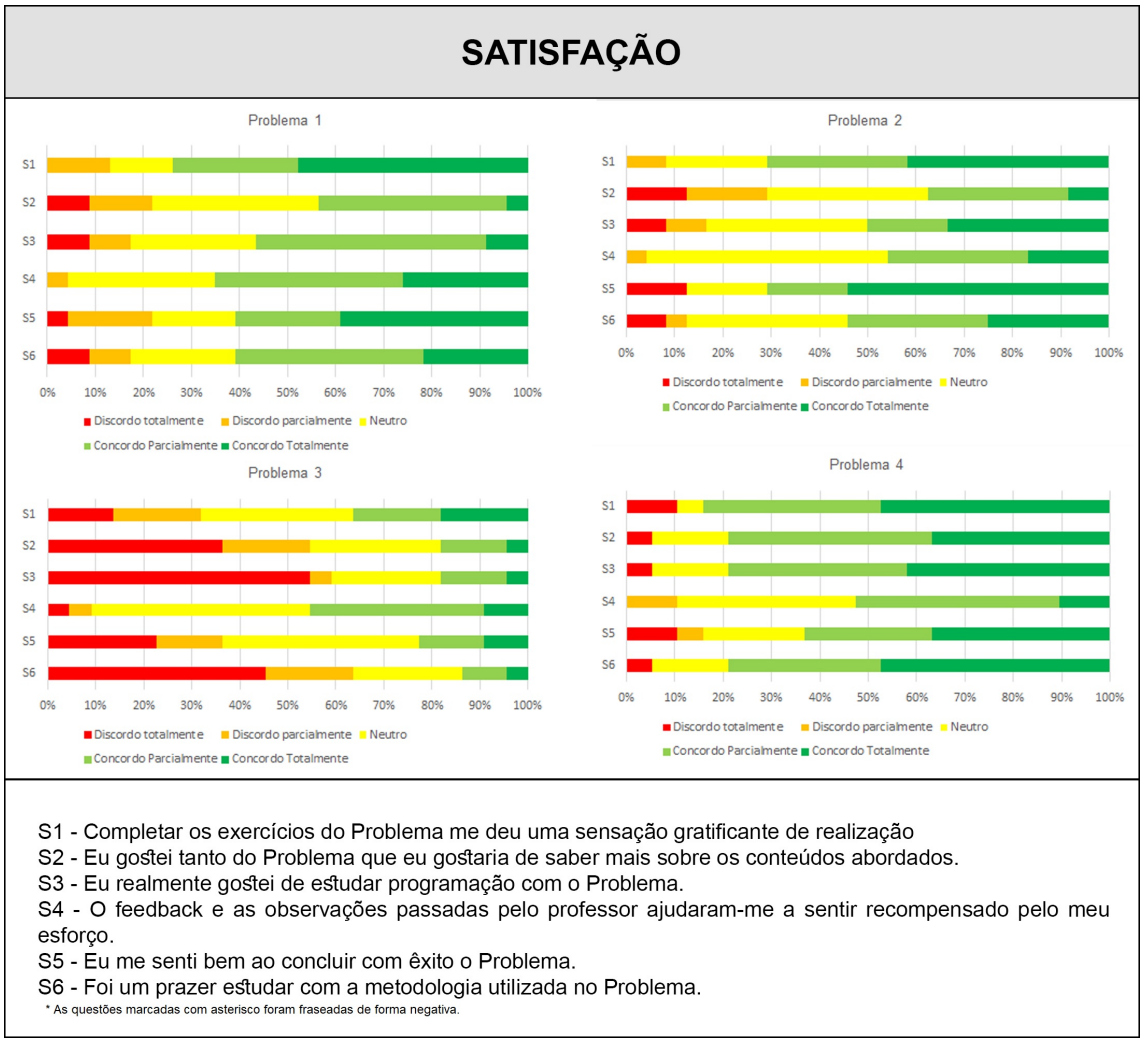


Figura 7. Resultados para a categoria Satisfação nos Problemas

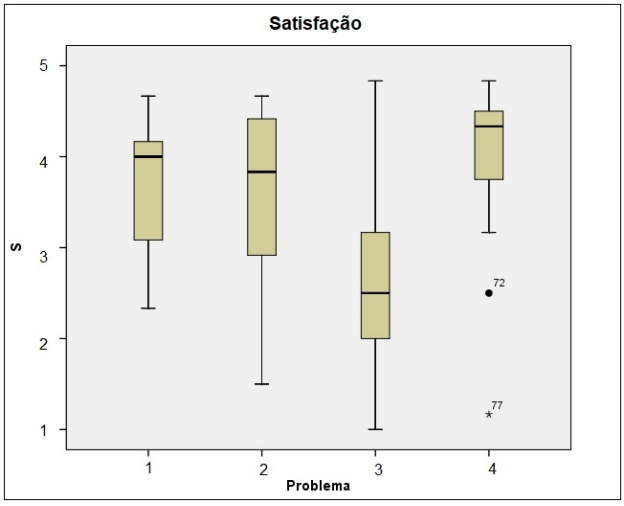


Figura 8. Box-Plot dos escores de Satisfação nos Problemas

ritmo ou estilo do material pedagógico, o uso do humor ou o envolvimento do estudante nas atividades. Os resultados sugerem que a atenção tem níveis relativamente altos nos Problemas 1, 2 e 4, e baixos no Problema 3.

Keller (1987) afirma que a Relevância consiste em fazer os estudantes perceberem a importância do que está sendo ensinado, e a utilização imediata destes ensinamentos devem fornecer respostas aos seus motivos e valores. As concepções de cada estudante certamente determinam os rumos por cujo intermédio seus objetivos podem ser alcançados. O fato de constatar relevância no assunto ensinado traz para o estudante a confiança de que ele está no caminho certo e que deve continuar a lutar para alcançar seus objetivos. A percepção da relevância foi relativamente alta nos Problemas 1, 2 e 4, e mais baixa no Problema 3. Porém, ainda sendo baixa no Problema 3, foi mais alta quando comparada às outras dimensões. Uma possível explicação para este grau de relevância é que os estudantes fazem uma disciplina de programação em um curso de computação, que é central neste curso e muito relacionada com a realidade profissional deles.

Na dimensão Confiança, foram obtidos os menores níveis em todos os problemas, quando comparadas às outras dimensões. Nos Problemas 1, 2 e 4, os resultados são mais neutros, e mais baixos no Problema 3. As prováveis justificativas seriam a complexidade dos conteúdos, o excesso de conceitos e as diversas competências que devem ser adquiridas para solucionar os problemas. Os estudantes enfrentam dificuldades ao lidar com a novidade dos conceitos de POO, Estruturas de Dados e Projeto de Sistemas.

Na dimensão Satisfação, foram alcançados níveis altos nos Problemas 1 e 2, baixos no Problema 3, e mais altos no Problema 4. A satisfação, segundo Keller (1987), é o resultado da avaliação cognitiva dos estudantes da equidade entre o esforço investido e os resultados percebidos ao final do processo de aprendizagem, a partir da interação com um dado objeto educacional. Deste modo, os resultados com dispersão ampla sugerem que nem todos vivenciam o sucesso ou a motivação da mesma maneira, nem gostam igualmente das suas experiências. Porém, os estudantes avaliaram que vale a pena continuar investindo seu esforço, confirmado nos Problemas 1, 2 e 4.

5.3. Lições aprendidas

O **cuidado na elaboração dos problemas** é relevante para a motivação dos estudantes. A qualidade dos problemas PBL está entre os maiores desafios, pois está diretamente vinculada ao aprendizado. O excesso de conceitos, a escolha do domínio do problema, a inserção de novidades e a complexidade exigida são aspectos que devem ser considerados.

O **excesso de conceitos** pode interferir no entendimento e no desenvolvimento da solução do problema. Bittencourt et al. (2013) mencionam haver uma grande variedade de conceitos que eram difíceis para os estudantes adquirirem. Eles afirmam que a dosagem de novos conceitos nos problemas propostos deve respeitar um processo gradual e a capacidade de assimilação dos estudantes.

A **escolha do domínio do problema** é relevante e deve introduzir fundamentos próximos à realidade dos estudantes, sendo uma estratégia para tornar o problema mais atraente e motivador. Um problema deve motivar os estudantes, principalmente ao inserir elementos próximos de suas realidades [Angelo et al. 2014]. E as questões iniciais dos problemas devem ser abertas, baseadas em conhecimentos prévios e/ou controversas, de forma a proporcionar discussão entre os aprendizes.

A **complexidade do problema** deve garantir que, com cooperação, os estudantes consigam solucionar o problema. Angelo et al. (2014) reiteram que os problemas devem ser complexos o bastante para que seja necessária a cooperação de todos os membros em sua solução. Ao mesmo tempo, descobrimos, neste trabalho, que a complexidade excessiva, como foi no problema 3, pode reduzir a motivação dos estudantes. Assim, é preciso encontrar um compromisso entre complexidade e motivação.

A **inserção de novidades** é essencial para manter os níveis de motivação. Keller (1987) afirma que não importa o quão bem sucedidas sejam algumas das estratégias de ensino, elas não permanecerão assim para sempre. Um erro cometido por professores e instrutores é que quando eles encontram uma estratégia que é altamente bem-sucedida, eles se exaltam e tendem a abusar dela. Cada nova estratégia tem um efeito inovador junto com qualquer nível mais profundo de conexão motivacional. O simples fato de ser novo pode estimular uma certa quantidade de interesse. Mas quando a novidade passar, a estratégia continuará a ser motivadora apenas se tiver uma conexão substancial e significativa com as exigências motivacionais dos alunos. Mesmo assim, eles se cansarão disso, porque o desejo de novidade é um aspecto da motivação humana.

6. Conclusões

Este trabalho apresentou um estudo de caso de uma abordagem baseada em PBL para ensino e aprendizagem de programação orientada a objetos durante um semestre letivo de um curso de Engenharia da Computação, onde foram avaliados os níveis de motivação dos estudantes utilizando o Modelo ARCS.

Os resultados apontam que os níveis de relevância foram relativamente altos em todos os problemas, possivelmente explicados por tratar-se de estudantes da área de TI. Os níveis de confiança foram um pouco mais baixos pelas prováveis dificuldades que os estudantes têm com os conceitos de programação orientada a objetos e as diversas competências a serem adquiridas. Os níveis de satisfação e atenção foram relativamente altos em três dos quatro problemas. O domínio dos Problemas 1, 2 e 4 foi considerado relevante pois estava mais próximo da realidade dos estudantes. O Problema 3 obteve os piores resultados e uma das possíveis explicações é a complexidade elevada e o domínio desconhecido. O Problema 4 alcançou os melhores resultados em todas as dimensões.

O cuidado na elaboração do problema é uma das etapas mais importantes em PBL. O domínio do problema, a complexidade dos conteúdos, o excesso de conceitos e introdução de novidades são questões essenciais. O domínio escolhido deve partir de um contexto real, que faça parte da vida dos estudantes, para que proporcione uma identificação, e assim, motive-os a prosseguir no desenvolvimento da atividade investigativa. A complexidade dos conteúdos e a quantidade de conceitos devem ter o tamanho ideal. A introdução de novidades é fundamental para a manutenção da motivação.

Em trabalhos futuros, faremos uma avaliação qualitativa dos dados que foram coletados das observações e entrevistas. Pretendemos analisar também a motivação nos módulos teóricos de Estruturas de Dados, Programação Orientada a Objetos e Projeto de Sistemas, que compõem o estudo integrado de Programação. Pretendemos também mensurar a aprendizagem nesta abordagem. Finalmente, a partir das lições aprendidas e da análise de dados realizada, devemos replicar este estudo de caso em outra turma do mesmo curso.

Referências

Angelo, M. F., Loula, A. C., Bertoni, F. C., and Santos, J. A. M. (2014). Aplicação e Avaliação do Método PBL em um Componente Curricular Integrado de Programação de Computadores. *Revista de Ensino de Engenharia*, 33(2):31–43.

Barrows, H. and Tamblyn, R. (1980). *Problem based-learning: An approach to medical education*, volume 1. Springer Publishing Company.

Bennedsen, J. and Caspersen, M. E. (2007). Failure rates in introductory programming. *SIGCSE Bull.*, 39(2):32–36.

Bittencourt, R. A. and Figueiredo, O. A. (2003). O Currículo do Curso de Engenharia de Computação da UEFS: Flexibilização e Integração Curricular. In *Anais do XXIII Congresso da SBC*, pages 171—-182, Campinas, São Paulo. SBC.

Bittencourt, R. A., Rodrigues, C. A., and Cruz, D. S. S. (2013). Uma Experiência Integrada de Programação Orientada a Objetos, Estruturas de Dados e Projeto de Sistemas com PBL. In *XXXIII Congresso da SBC – XXI WEI.*

Burton, P. J. and Bruhn, R. E. (2003). Teaching Programming in the OOP Era. *ACM SIGCSE Bulletin*, 35(2):111–114.

Cheiran, J. F. P., de M Rodrigues, E., de S Carvalho, E. L., and da Silva, J. P. S. (2017). Problem-based learning to align theory and practice in software testing teaching. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*.

Cintra, C. S. and Bittencourt, R. A. (2015). Being a PBL Teacher in Computer Engineering : An Interpretative Phenomenological Analysis. In *FIE Conference*. IEEE.

Delisle, R. (1997). *How to use problem-based learning in the classroom*. Ascd.

Jenkins, T. (2001). The motivation of students of programming. *SIGCSE Bull.*, 33(3):53–56.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, pages 53–58.

Keller, J. M. (1987). Development and Use of the ARCS Model of Instructional Design. *Journal of instructional development*, 10(3):2–10.

Keller, J. M. (2010). *Motivational Design for Learning and Performance: The ARCS Model Approach*. Springer US.

Kölling, M. (1999). The problem of teaching object-oriented programming, part 1: Languages. *Journal of Object-Oriented Programming*, 11(8):8–15.

O'Grady, M. J. (2012). Practical problem-based learning in computing education. *Trans. Comput. Educ.*, 12(3):10:1–10:16.

Ray, W. S. (1964). *The Science of psychology: an introduction*. Macmillan.

Santos, D. M. B., Pinto, G. R. P. R., Sena, C. P. P., Bertoni, F. C., and Bittencourt, R. A. (2007). Aplicação do Método de Aprendizagem Baseada em Problemas no Curso de Engenharia de Computação da Universidade Estadual de Feira de Santana. In *XXXV Congresso Brasileiro de Educação em Engenharia*.

Uma Abordagem Gamificada para o Ensino de Lógica de Programação: relato de experiência

Carina Machado de Farias[12], Fellipe Pereira Azevedo[1], José Elias de Jesus Dias[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Av. Centenário, 500, Nazaré - Jacobina - Bahia, 44700-000, Brasil

[2] Pesquisadora do Grupo de Pesquisa Automação, Eficiência Energética e Produção

carina.farias@ifba.edu.br, azevedofellipe16, elias.uzumak{@gmail.com }

Abstract. *The introductory course in Programming Logic is the basis for the training of students in technical and superior courses in the area of Computing. Assimilating the content covered in this course requires students a new way of thinking and skills that are normally not developed in regular education. As a result, the discipline has high rates of disapproval and withdrawal, and is usually one of the disciplines that fail most in these courses. The concern with high failure rates and avoidance in this discipline has led teachers and researchers to seek suitable alternatives for the traditional way of teaching algorithms and programming, where the student is the passive agent of learning, and the teacher is the agent that transmits the contents . In this context, the present work presents a proposal of gamification of the discipline Logic of Programming and reports the results obtained with application of the proposal in a class of the Technical Course of Informatics, of the IFBA, Jacobina campus.*

Resumo. *A disciplina introdutória de Lógica de Programação constitui a base de formação dos estudantes dos cursos técnicos e superiores da área de Computação. Assimilar os conteúdos abordados nessa disciplina requer dos estudantes uma nova forma de pensar e habilidades que normalmente não são desenvolvidas no ensino regular. Como resultado, a disciplina apresenta altos índices de reprovação e desistência, sendo normalmente uma das disciplinas que mais reprova nesses cursos. A preocupação com as altas taxas de reprovação e evasão nesta disciplina tem levado professores e pesquisadores a buscarem alternativas à forma tradicional de ensino de algoritmos e programação, onde o aluno é o agente passivo da aprendizagem, e o professor o agente transmissor dos conteúdos. Nesse contexto, o presente trabalho apresenta uma proposta de gamificação da disciplina Lógica de Programação e reporta os resultados obtidos com aplicação da proposta em uma turma do Curso Técnico de Informática, do IFBA, campus Jacobina.*

1. Introdução

A disciplina Lógica de Programação, presente nas primeiras etapas dos cursos técnicos e superiores da área de Computação, constitui a base de formação dos estudantes, requerendo deles uma nova forma de pensar e habilidades que dificilmente são desenvolvidas no ensino regular, levando os estudantes a terem grandes dificuldades e, consequentemente, apresentarem baixo rendimento durante o curso (GUEDES, 2014).

Em um estudo realizado na Universidade de São Paulo, Bosse e Gerosa (2015) apontam que o índice de reprovação na disciplina Introdução a Programação, foi de 30%, sendo que em vários casos ultrapassaram os 50%, considerando os anos de 2010 a 2014.

Concluíram ainda que mais de 25% dos alunos aprovados neste período fizeram 2 ou mais vezes a disciplina. Comparando-se com disciplinas do mesmo semestre, nos cursos de computação, Introdução à Programação é uma das que mais reprova.

A origem do problema das reprovações e evasões na disciplina Lógica de Programação diz respeito a alunos, professores e até mesmo às metodologias utilizadas, sendo possível destacar alguns fatores que contribuem para o cenário atual, tais como: precária base lógico-matemática dos alunos; falta de dedicação aos estudos; limitações do professor; material didático de apoio e metodologia de ensino ineficientes; etc.

Considerando a importância dessa problemática e buscando alternativas para minimizar o número de reprovações e abandonos, muitos professores e pesquisadores têm buscado apoio na gamificação, visando engajar e motivar os alunos e, consequentemente, aumentar seu rendimento e melhorar seu aprendizado. No decorrer da execução desse projeto foram encontrados alguns trabalhos que tratam da inclusão de elementos de jogos em disciplinas como forma de motivar os alunos e alcançar melhores resultados (DA SILVA FIGUEREDO et al., 2015), (MAEKAWA et al., 2015), (JUCÁ et al., 2014), (AGUIAR, 2015).

Segundo Fardo (2013), gamificação, tradução do termo *gamification*, criado pelo programador britânico Nick Pelling em 2003, consiste no uso de elementos, mecanismos, dinâmicas e técnicas de jogos na rotina profissional, escolar e social do indivíduo num contexto fora do jogo, com o objetivo de incrementar a participação e gerar engajamento e comprometimento por parte dos usuários e maior interação entre as pessoas e empresas com base na oferta de incentivos que estimulam a colaboração entre os envolvidos, para que realizem de forma mais prazerosa suas tarefas.

No ambiente escolar, gamificar é a arte de ensinar conteúdos didáticos através de jogos. Essa é uma técnica muito utilizada nos EUA e que no Brasil, começa a receber a devida atenção. A utilização de jogos, sejam eles virtuais ou não, incentiva o aprendizado de forma interativa e divertida. Dessa forma, a gamificação surge no cenário educacional brasileiro como uma ferramenta capaz de combater a falta de interesse e a dispersão dos alunos em sala de aula.

Neste trabalho, a gamificação foi utilizada na disciplina Lógica de Programação, do Curso Técnico Subsequente de Informática, do IFBA, campus Jacobina, com a finalidade de promover o engajamento dos alunos com relação à matéria, mediado por elementos de games que possam auxiliar nesse processo. Nesse contexto, a gamificação foi também utilizada com o intuito de desenvolver nos alunos o hábito de estudar regularmente, uma vez que a disciplina apresenta conteúdos de caráter cumulativo, de forma que estudar de forma contínua poderia auxiliar os alunos a terem um melhor desempenho na resolução dos exercícios propostos ao longo da disciplina, e com isso aumentar o número de aprovações apresentados na disciplina.

2. A Disciplina Lógica de Programação

No Curso Técnico de Informática analisado, Lógica de Programação é uma disciplina obrigatória e presencial, com carga horária total de 60h, ofertada aos alunos ingressantes no primeiro semestre. O curso é composto por 04 semestres, sendo a disciplina Lógica de Programação, a primeira das 04 disciplinas de programação ofertadas no decorrer do curso.

Embora ministrada por diferentes professores, a metodologia padrão de ensino dessa disciplina entre 2013 e 2016 foi baseada na exposição dos conteúdos, seguido da aplicação de

listas de exercícios, resolvidos utilizando o computador. As avaliações eram baseadas em listas de exercícios, desenvolvidas em grupos, ou provas individuais no computador.

As taxas de reprovação registradas na disciplina superaram os 90%, como apresentado na Tabela 01, fato que incomodou os autores desse trabalho, fazendo-os buscarem alternativas que possibilitassem reduzir essas taxas.

Tabela 01: Aprovações e Reprovações na Disciplina Lógica de Programação

INGRESSO	INICIANTES	APROVADOS	REPROVADOS	TAXA DE REPROVAÇÃO
2013.2	42	18	24	57,14%
2014.2	42	15	27	64,28%
2015.2	37	03	34	91,89%
2016.2	50	26	24	48%
2017.2	31	18	13	41,93%

A partir dessa inquietação, surgiu a proposta de gamificação da disciplina, como uma tentativa de ampliação das taxas de aprovação na disciplina. Tal proposta é apresentada na próxima seção.

3. Proposta de Gamificação da Disciplina

A fim de inserir o conceito de gamificação na disciplina Lógica de Programação, foi criado um jogo conceitual intitulado "Quem quer ser Programador?"[1], com a proposta de auxiliar no engajamento dos alunos, e conseqüente melhorar seu desempenho na disciplina.

Para tornar a estratégia adotada mais próxima de um jogo real foi criada uma nomenclatura, para que o aluno pudesse se sentir participando de um game. A nomenclatura estabelecida é apresentada no Quadro 01.

Quadro 01: Nomenclatura do Jogo

Termo Tradicional	Termo Equivalente
Disciplina Lógica de Programação	Jogo Quem Quer ser Programador
Aula	Treinamento
Aluno	Jogador
Professor	Mestre Programador
Notas	Bytes
Exercício	Batalha Amigável
Avaliação	Batalha Mortal
Avaliação de Recuperação	Batalha de Recuperação
Prova	Missão

[1] Documento de Regras do Jogo, disponível em https://is.gd/q2ZtHQ.

O jogo foi concebido em 05 fases, cada fase com sua duração previamente definida e uma pontuação máxima de Bytes a ser conquistada, conforme apresentado na Tabela 02.

Tabela 02: Detalhamento das Fases do Jogo

Fase	Denominação	Pontuação Máxima
1	Quero ser Estagiário	100 Bytes
2	Quero ser Trainee	150 Bytes
3	Quero ser Programador Junior	200 Bytes
4	Quero ser Programador Pleno	250 Bytes
5	Quero ser Programador Sênior	300 Bytes
	TOTAL	1000 Bytes

O objetivo do jogo é que o jogador acumule Bytes suficientes para se tornar um programador sênior. Cada 100 Bytes conquistados no jogo equivale a 1 ponto na disciplina, de forma que ao final do jogo foi possível converter a pontuação do jogador em nota. Para obter aprovação na disciplina o jogador precisava acumular no mínimo 600 Bytes dos 1000 Bytes possíveis.

Em cada fase, foram propostos treinamentos, batalhas e missões para o jogador conquistar Bytes e alcançar o objetivo do jogo. Ao final do jogo, somou-se a quantidade de Bytes que o jogador obteve em cada uma das fases, de forma a obter a pontuação total do jogador. Tal pontuação foi utilizada para definir o nível que o jogador alcançou no jogo, conforme previsto no Quadro 02.

Quadro 02: Quantidade de Bytes Necessários para alcançar cada Nível

Nível	Pontuação Alcançada
Estagiário	0 a 100 Bytes
Programador Trainee	101 a 250 Bytes
Programador Júnior	251 a 450 Bytes
Programador Pleno	451 a 700 Bytes
Programador Sênior	701 a 1000 Bytes

Os conteúdos previstos no plano da disciplina foram distribuídos entre as fases do jogo, de forma que para cada fase, foi previsto um conjunto de conteúdos previamente selecionados a serem trabalhados com a turma, como mostra o Quadro 03.

Quadro 03: Distribuição dos Conteúdos entre as Fases do Jogo

Fase	Denominação da Fase	Conteúdos
1	Quero ser Estagiário	Lógica, Algoritmos e Linguagem Algorítmica
2	Quero ser Trainee	Variáveis e Constantes, Comandos de Entrada e Saída de Dados, Operadores Aritméticos, Comandos de Atribuição, Estrutura Sequencial
3	Quero ser Programador Junior	Operadores Lógicos e Relacionais, Estruturas de Seleção (Condicionais)
4	Quero ser Programador Pleno	Estruturas de Repetição
5	Quero ser Programador Sênior	Ponteiros, Vetores, Matrizes e Arquivos

3.1 Elementos de Games Presentes na Proposta

Elementos de game são mecanismos presentes em jogos com o objetivo de promover uma experiência lúdica ao jogador durante o jogo (Fardo, 2013). Em vista dessa definição, as seções a seguir apresentam os elementos de game identificados no jogo "Quem Quer ser Programador?". Cada elemento é conceituado, e a forma como o elemento se apresenta no jogo é discutida.

Objetivo

Todo jogo precisa ter um objetivo bem definido, sem ambigüidades. O objetivo do jogo define o propósito e o resultado a ser alcançado no jogo. A proposta do jogo "Quem quer ser Programador?" definiu que o objetivo do jogador era acumular o máximo de bytes possível e, com isso, chegar ao nível de Programador Sênior. É um objetivo bem definido e absoluto, não permitindo interpretações ambíguas por parte do jogador, o que faz com que o entendimento acerca da meta principal do jogo fique mais clara.

Regras

As regras definem como o jogador deve jogar, estabelecendo restrições sobre as ações do jogador. É necessário, portanto, compreender as regras do jogo para estar apto a jogar. As regras do jogo "Quem que ser Programador?" foram estabelecidas em um documento, compartilhado com todos os jogadores antes do início do jogo, de forma que todos tiveram acesso às regras, estando, portanto, aptos a jogar. O quadro 04 apresenta um trecho do documento de regras do jogo, descrevendo parcialmente as regras para a primeira fase.

Quadro 04: Trecho do Documento de Regras do Jogo

A fase 1 é composta por 03 treinamentos. A presença em cada treinamento pontua 1 Byte. Nessa fase são realizadas 03 batalhas amigáveis. Essas batalhas servem para fortalecer o conhecimento de cada jogador. Em cada batalha amigável que o jogador participa ele pontua 4 Bytes.

Recompensas

São elementos fundamentais de um jogo. São gratificações que os jogadores recebem para estimular e motivar a realização das atividades. Em todas as 05 fases de "Quem quer ser Programador?" as recompensas se apresentam na forma de direitos adquiridos, ou seja, o jogador adquire direitos durante o jogo caso realize uma determinada ação durante o jogo. O quadro 05 destaca um exemplo de recompensa presente na fase 3, "Quero ser Programador Júnior".

Quadro 05: Exemplo de Recompensa presente na Fase 03 do Jogo

Ao final desta fase, os jogadores que conquistarem 120 Bytes, ou mais, recebem como recompensa o direito de escolher entre dois problemas qual prefere resolver durante o desafio parcial da fase seguinte.

Feedback

Feedback é um elemento de jogo que possibilita a visualização da evolução do jogador diante do objetivo do jogo. A cada missão cumprida ou batalha travada, os resultados da missão ou batalha eram inseridos no Canvas[2], na forma de um placar, de forma que o jogador podia acompanhar a sua evolução no jogo.

Níveis

Um jogo pode ser subdividido em níveis de forma que o objetivo principal do jogo é quebrado em vários objetivos menores. O jogador muda de nível à medida que cumpre os requisitos previstos no nível anterior. "Quem quer ser Programador?" foi projetado para acontecer em 05 etapas, conforme consta na Tabela 01, sendo que o nível de dificuldade dos conteúdos abordados em cada etapa é crescente, devendo o jogador percorrer o caminho que se inicia no nível mais básico, "Quero ser Estagiário", até chegar ao nível mais avançado, "Quero ser Programador Sênior".

Pontuação

Representa a progressão do jogador no jogo de maneira numérica. A pontuação permite que o jogador saiba até onde ele conseguiu chegar no jogo. Em "Quem quer ser Programador?" a pontuação foi definida a partir do acúmulo de Bytes, onde a quantidade de Bytes acumulada indica o nível alcançado pelo jogador, como pode ser observado no Quadro 02.

Erro

Durante um jogo os jogadores têm que superar desafios. Eventualmente, não obtêm êxito e tentam novamente. No jogo "Quem quer ser Programador?" o elemento "erro no processo" está caracterizado pelas "Batalhas de Recuperação", que permitem aos jogadores tentar de novo, com o objetivo de recuperar os pontos perdidos em uma batalha mortal. O jogador que não obteve êxito em alguma Batalha Mortal poderá participar das batalhas de recuperação e resgatar os pontos perdidos.

4 Execução da Proposta

A introdução da gamificação na disciplina de Lógica de Programação foi executada na turma ingressante no segundo semestre de 2017 do Curso Técnico de Informática do IFBA, campus, Jacobina. Haviam 31 alunos matriculados na turma, entretanto, 08 alunos, embora

[2] Ambiente virtual de aprendizagem, disponível em https://canvas.instructure.com.

matriculados, nunca assistiram a nenhuma aula. No decorrer do semestre outros 03 alunos evadiram, de forma que no final do período a turma tinha apenas 20 alunos.

No primeiro dia de aula, os alunos foram submetidos a um questionário[3], buscando identificar o perfil da turma em relação ao hábito de jogar. 19 alunos responderam ao questionário, o que correspondeu a 82,6% da turma inicial de 23 alunos.

Em relação à idade dos alunos, percebeu-se a predominância de um público jovem, com idade entre 17 e 22 anos. Essa foi uma informação relevante, uma vez que se trata de um público normalmente inserido na cultura dos jogos.

Outras duas informações extraídas dos questionários apontaram que existia chance de sucesso em aplicar a gamificação nessa turma, uma vez que mais de 60% dos indivíduos indicaram já ter jogado mais de 10 jogos no decorrer de sua vida, ao mesmo tempo em que mais de 60% da turma afirmou que costuma dedicar entre 1 e 7h do seu tempo semanal aos jogos.

Por fim, uma última contribuição dos questionários foi a respeito das expectativas dos alunos em relação à introdução da gamificação na disciplina. Quase 100% dos alunos esperava que a disciplina se tornasse mais fácil, enquanto mais de 50% tinha a expectativa de se sentirem mais motivados. Esses resultados congraçaram com os objetivos do projeto de motivar e melhorar o desempenho dos alunos.

Diante da análise dos questionários, o perfil dos alunos foi traçado, e concluiu-se que a turma selecionada era adequada à proposta do projeto, por ser uma turma jovem, afinada com os games, e com boas expectativas acerca do formato diferente utilizado na disciplina.

O ambiente do jogo foi o Laboratório de Informática, entretanto, o ambiente virtual Canvas foi também utilizado a fim de facilitar a comunicação entre o professor da disciplina e os alunos. No Canvas, a professora criou a disciplina virtualmente, inseriu os alunos participantes, e no decorrer do semestre disponibilizou o material de treinamento, bem como as batalhas e missões a serem travadas pelos jogadores, sendo possível também ter uma visão geral da realização das tarefas de cada aluno e da turma como um todo. Além da professora, cada aluno participante tinha também uma conta no ambiente virtual, sendo possível acessar o material disponibilizado pela professora, enviar suas respostas às batalhas e missões, além de acompanhar seu desempenho, consultar sua pontuação e, consequentemente, o nível alcançado no jogo.

No decorrer do jogo os jogadores participaram de 20 treinamentos, ocorridos no Laboratório de Informática, 12 batalhas amigáveis, 16 batalhas mortais (e 16 batalhas de recuperação) e 05 missões, disponibilizadas através do Canvas.

20 jogadores jogaram até o final do jogo, sendo que 60% alcançou o nível máximo previsto, que é o nível de Programador Sênior, como mostra a Tabela 03.

Tabela 03: Nível alcançado pelo jogador de acordo com sua pontuação

Nível Alcançado	Quantidade de Alunos
Estagiário	1
Trainee	1
Programador Junior	1
Programador Pleno	5
Programador Sênior	12
Total	20

[3] Questionário de identificação do perfil do aluno, disponível em https://is.gd/PozaSR.

5 Resultados e Discussões

A inclusão de elementos de games na disciplina Lógica de Programação, do curso Técnico Subsequente de Informática, do IFBA, campus Jacobina, foi realizada buscando alcançar os seguintes objetivos:

i. Engajar/motivar os alunos a ter melhor desempenho na disciplina;
ii. Desenvolver nos alunos o hábito de estudar regularmente; e
iii. Aumentar o número de aprovações apresentados na disciplina.

A fim de verificar se os objetivos foram alcançados, a análise dos resultados foi realizada a partir de duas perspectivas:

i. desempenho dos alunos nas atividades avaliativas propostas na disciplina; e
ii. impressões dos alunos acerca da experiência.

Sob a ótica do desempenho dos alunos, foi possível perceber que o trabalho alcançou seu objetivo de aumentar as taxas de aprovação na disciplina. Esse fato pode ser comprovado através da consulta à Tabela 01, apresentada anteriormente, onde pode-se constatar que a turma de 2017.2, onde a gamificação foi utilizada, apresentou a menor taxa de reprovação, quando comparada às turmas de semestres anteriores.

A fim de buscar verificar se os demais objetivos do projeto foram atingidos, analisou-se as impressões dos alunos em relação ao formato adotado na disciplina, a partir da aplicação de um questionário[4] ao final do período.

Foram obtidas 20 respostas a esse questionário, a partir das quais foi possível constatar que o projeto conseguiu motivar os alunos na disciplina, uma vez que 80% das respostas qualificaram o formato adotado na disciplina como Muito motivante ou Motivante. Além disso, a maioria (80%) dos alunos acredita que seria válido aplicar a gamificação em outras disciplinas. Quando perguntados se o formato da disciplina influenciou em sua dedicação aos estudos, 70% respondeu que sim. 70% dos alunos também indicaram que acreditam ter tido um melhor aprendizado devido ao formato adotado na disciplina.

Foi possível constatar também, a partir das respostas ao questionário, que a quantidade de horas dedicadas à disciplina girou em torno de 1 a 5 horas semanais, como afirmado por 80% dos alunos.

A fim de melhorar a proposta para experiências futuras em outras turmas, buscou-se identificar, através das respostas obtidas com a aplicação do questionário, quais os pontos positivos e negativos da proposta, do ponto de vista dos alunos.

Quanto aos elementos de jogo identificados pelos alunos na proposta, **desafios** foi citado por 90%, seguido por **competição**, **recompensas**, **progressão**, **regras** e **ranking**, todos com 70%. Um ponto importante a ser ressaltado nessa questão é que nenhum aluno apontou **diversão** nem **abstração da realidade** como elementos presentes no projeto, destacando aspectos que precisam ser revistos em propostas futuras.

Quando questionados sobre os aspectos positivos da metodologia adotada, os alunos apontaram: **motivação**, **dedicação**, **aprendizado** e **trabalho em grupo**. O depoimento de um aluno, durante a aplicação dos questionários, reflete o entendimento da maioria sobre os fatores positivos da proposta: "Muito estimulante, instiga a competição. Cada problema colocado aos alunos permite evoluir o raciocínio lógico e matemático."

[4] Questionário sobre as impressões dos alunos acerca da experiência, disponível em https://is.gd/nf1I1i.

Já os pontos negativos identificados foram: **pouco estímulo às atividades em grupo**, **excesso de atividades extra-classe**, **muita pressão**, **pouco tempo para o desenvolvimento das tarefas**, **dificuldade de administração do tempo para dedicar às outras matérias**.

Do ponto de vista da professora da disciplina, "o maior ganho obtido com o formato adotado na disciplina foi o empenho dos alunos em resolver as tarefas propostas, além da dedicação semanal ao estudo dos conteúdos. Tal dedicação se refletiu nos melhores índices de aprovação registrados para a disciplina desde 2013", conforme consta na Tabela 01.

Por outro lado, os participantes do projeto, e autores deste artigo consideraram que o planejamento da gamificação, bem como o acompanhamento da realização das tarefas exigiram mais tempo do que o exigido para o planejamento e acompanhamento da disciplina na forma tradicional. Na opinião dos autores, é necessário ainda buscar uma ferramenta que gerencie melhor a gamificação na disciplina, a fim de se obter melhores resultados. Segundo os autores, o Canvas ajudou, mas não é a ferramenta mais adequada para a proposta, uma vez que não acomoda os conceitos de elementos de games de maneira natural.

6 Considerações Finais

Este trabalho apresentou uma proposta de gamificação para uma das disciplinas de maior índice de reprovação em cursos de Computação e áreas afins, a disciplina Lógica de Programação. O estudo também relatou a experiência da aplicação dessa proposta na turma de 2017.2 do Curso Técnico Subsequente de Informática do IFBA, campus Jacobina.

Apesar de representar um desafio para a docente, a introdução da gamificação na disciplina Lógica de Programação alcançou os seguintes benefícios:

- Empenho dos alunos na disciplina;
- Persuasão dos alunos sobre a importância do hábito de estudar regularmente; e
- Aumento do percentual de aprovados na disciplina.

Por outro lado, a experiência destacou alguns elementos que requerem melhorias, a fim de se obter resultados ainda mais satisfatórios com a gamificação:

- Necessidade de uma plataforma virtual mais adaptada aos conceitos de gamificação e elementos de jogo.
- Inclusão da diversão como elemento do jogo.
- Maior estímulo à colaboração entre os jogadores.
- Redução do número de tarefas extra-classe, a fim de não trazer prejuízos para o desempenho dos alunos em outras disciplinas.

Diante dos aspectos analisados durante o projeto, como trabalho futuro, pretende-se ajustar a proposta de gamificação, introduzindo os aspectos identificados neste estudo, e aplicá-la em outras turmas.

Referências

AGUIAR, Janderson. Experiência baseada em Gamificação no Ensino sobre Herança em Programação Orientada a Objetos. In: **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**. 2015. p. 1444.

BOSSE, Yorah; GEROSA, Marco Aurélio. Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo: Um Estudo Preliminar. In: XXXV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO / XXIII WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 2015, Recife/PE. **Anais do XXXV Congresso da Sociedade Brasileira de Computação.** Porto Alegre: Sociedade

Brasileira de Computação, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2015/044.pdf>. Acesso em: 06 fev. 2018.

BRAZIL, André; BARUQUE, Lúcia. Gamificação Aplicada na Graduação em Jogos Digitais. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. 2015. p. 677.

DA SILVA FIGUEREDO, K. et al. Uma Abordagem Gamificada para o Ensino de Programação Orientada a Objetos. In: ***23º Workshop sobre Educação em Computação***. 2015. Recife, PE: SBC - Sociedade Brasileira de Computação.

FARDO, Marcelo Luís. **A gamificação como estratégia pedagógica:** estudo de elementos dos games aplicados em processos de ensino e aprendizagem. 2014.

GUEDES, E. Um Estudo Observacional sobre a Disciplina Introdutória de Programação. In: ***XX Workshop de Informática na Escola***. 2014. Dourados, MS: SBC - Sociedade Brasileira de Computação, pp.552-561. Disponível em: http://br-ie.org/pub/index.php/wie/article/view/3140/2648.

JUCÁ, Paulyne M.; ROLIM, G. Aplicação da Gamificação na Disciplina de Empreendedorismo. In: **XXII Workshop sobre Educação em Computação (WEI 2014)**. 2014.

MAEKAWA, Christian; NAGAI, Walter; IZEKI, Claudia. Relato de Gamificação da disciplina Projeto e Análise de Algoritmos do curso de Engenharia de Computação. In: **Anais do Workshop do Congresso Brasileiro de Informática na Educação**. 2015. p. 1425.

SoccerCraft: Relato de Atividade para Ensino Aprendizagem de Habilidades do Pensamento Computacional Aplicada no Sexto Ano do Ensino Fundamental*

Simão Martin[1], **Simone Cavalheiro**[1], **Renata Reiser**[1], **Luciana Foss**[1], **Ana Rita Mazzini**[1], **André Du Bois**[1], **Clause Piana**[1]

[1]Centro de Desenvolvimento Tecnológico - Universidade Federal de Pelotas (UFPel), Rua Gomes Carneiro, 1 - 96.010-610 - Pelotas - RS, Brasil

Abstract. *Computational Thinking (CT) is a fundamental skill in many areas and involves several techniques for solving problems, based on fundamental concepts of Computer Science. This paper presents an activity proposal for the sixth year of elementary school that develops algorithmic thinking, one of the abilities of the CT. The methodology consists of playful and fun tasks, without the use of the computer, that simulate a soccer match with characters similar to those of the Minecraft game. The activity is denominated SoccerCraft. In addition to the proposed methodology, the results and reports of the application of the activity to a class of a public school in the city of Pelotas are also presented.*

Resumo. *O Pensamento Computacional (PC) é uma habilidade fundamental para diversas áreas e envolve várias técnicas para a resolução de problemas, embasadas em conceitos fundamentais da Ciência da Computação. Este artigo apresenta uma proposta de atividade para o sexto ano do ensino fundamental que desenvolve o pensamento algorítmico, uma das habilidades do PC. A metodologia consiste em tarefas lúdicas e divertidas, sem o uso do computador, que simulam um jogo de futebol com personagens similares aos do jogo Minecraft. Esta atividade foi denominada SoccerCraft. Além da metodologia proposta, também são apresentados os resultados e relatos da aplicação da atividade a uma turma de uma escola pública do município de Pelotas.*

1. Introdução

O Pensamento Computacional (PC, do inglês *Computational Thinking*) consiste em uma metodologia para resolução de problemas baseada nos conceitos da ciência da computação [Wing 2006]. O PC é um método fundamental para todos que utiliza conceitos e técnicas aplicados na criação de programas computacionais, com o intuito de resolver problemas de diversas áreas, não necessariamente ligadas à computação [Marques et al. 2017, Barbosa et al. 2017, dos Reis et al. 2018].

Neste contexto, torna-se relevante estimular as crianças a terem contato desde cedo com o PC [França et al. 2012]. Com o propósito de auxiliar esse precoce incremento e consolidação do raciocínio lógico e computacional nas crianças, as escolas de Ensino Fundamental são consideradas locais importantes para a introdução desta

*Projeto realizado com o apoio da PREC, PRPPG e PPGC / UFPel

metodologia. Vários trabalhos têm sido desenvolvidos neste sentido, tanto nacionalmente, [Barbosa et al. 2017, França et al. 2012, Santos et al. 2016], quanto internacionalmente, como a proposta da Google, em 2013, apresentando um conjunto de atividades do PC no intuito de auxiliar alunos do ensino primário e secundário dos Estados Unidos [Google 2013].

Por sua vez, o futebol destaca-se como o esporte mais praticado e difundido entre as crianças brasileiras e é adorado pela maioria delas. Um dos jogos eletrônicos que atualmente faz sucesso dentre as crianças é o jogo Minecraft [Persson and Bergensten 2011], sendo o terceiro mais jogado do mundo [TNH1 2017].

Motivados por este contexto, este trabalho apresenta a concepção, modelagem e desenvolvimento de um jogo educacional denominado SoccerCraft, considerando também uma etapa de validação, na qual são descritos e relatados o conjunto de atividades aplicadas em alunos de escola pública. A metodologia proposta explora regras de futebol utilizando personagens similares aos do jogo Minecraft com o intuito de desenvolver, via estratégias inerentes ao PC, a habilidade de pensamento algorítmico nos alunos do sexto ano do ensino fundamental. Além do incentivo ao desenvolvimento do raciocínio lógico via PC, dentre os objetivos secundários que podem ser alcançados pode-se citar o incremento da motivação do aluno em participar das aulas e das atividades propostas.

Este artigo está estruturado com a seguinte organização: a Seção 2 apresenta uma breve explicação sobre o conceito de PC e suas particularidades. A Seção 3 descreve a proposta metodológica utilizada na aplicação da atividade, incluindo materiais utilizados e os planos de aula. Na sequência, segue a Seção 4 que relata a aplicação da atividade, além de apresentar e discutir os resultados provenientes das tarefas. Na conclusão, têm-se as considerações finais e continuidade do trabalho.

2. Pensamento Computacional

Segundo a Sociedade Internacional de Tecnologia na Educacão (ISTE), em [ISTE 2013], o PC é um processo de resolucão de problemas que inclui (mas não é limitado a) as seguintes características:

- Reformular um problema no intuito de resolvê-lo mais facilmente.
- Organizar dados logicamente.
- Analisar dados complexos por partes, utilizando abstração e decomposição.
- Utilizar algoritmos para a automatização de soluções complexas.
- Formular soluções mais eficientes para problemas já resolvidos.
- Generalizar a solução de um problema para a resolução de problemas mais amplos.

A ISTE, juntamente com a CSTA (*Computer Science Teachers Association*) e a NSF (*National Science Foundation*), elaboraram um conjunto de ferramentas visando a competência de habilidades básicas do PC no Ensino Médio chamado *Computational Thinking in K-12 Education Leadership Toolkit*. As ferramentas apresentadas nesse *toolkit* apresentam nove conceitos fundamentais do PC em um quadro de progressão, são eles: coleta de dados, análise de dados, representação de dados, decomposição de problemas, algoritmos e procedimentos, automação, simulação e paralelismo.

Existe uma ampla gama de projetos com o objetivo de levar à escolas do Ensino Básico atividades para a difusão do PC em diversos níveis de ensino. Alguns trabalhos relacionados estrangeiros merecem destaque, como [Barr and Stephenson 2011, Carnegie Mellon 2013, Google 2013, Lee et al. 2011, Kafura and Tatar 2011, Lugo and Olabe 2018].

No Brasil existem alguns projetos do PC que merecem destaque como, por exemplo, o de [Teixeira et al. 2015] que apresenta o projeto entitulado "Escola de Hackers", no município de Passo Fundo-RS, no qual mais de 300 alunos do 6º ao 9º ano do Ensino Fundamental, entre 11 e 14 anos, utilizam a ferramenta Scratch para desenvolver competências na área de programação de computadores e de raciocínio lógico matemático. Nacionalmente também se destaca o *Computação Desplugada*, que tem o objetivo de divulgar fundamentos da Ciência da Computação sem o uso do computador para alunos de escolas públicas [Silva et al. 2017] e o *Explorando o PC para a Qualificação do Ensino Fundamental* (ExpPC) [UFPel 2018], que visa a elaboração e aplicação de atividades lúdicas e interessantes que desenvolvam conceitos considerados essenciais no PC, para estudantes de escolas da rede pública [Campos et al. 2014, Junior et al. 2017, Marques et al. 2017, Weisshahn et al. 2016]. No Brasil ainda há trabalhos como os de [França et al. 2012, Zanetti et al. 2017, Batista et al. 2015].

3. Metodologia da Atividade

Este trabalho, que foi concebido junto ao projeto ExpPC, tem por objetivo trabalhar os conceitos básicos de algoritmos mediante a aplicação do jogo educacional denominado SoccerCraft. Sua proposta de metodologia defende utilizar uma abordagem de aprendizagem de algoritmos baseada em um jogo de futebol com personagens semelhantes aos do jogo Minecraft, sendo esse o principal diferencial deste trabalho em relação às demais abordagens da literatura. A atividade proposta foi dividida em cinco aulas, cada uma sendo um pouco mais complexa que a anterior.

O professor introduz novos conceitos de algoritmos à medida que as regras do jogo são adicionadas ao decorrer das aulas. As tarefas não possuem dependência de computadores. O público-alvo foram crianças do sexto ano do Ensino Fundamental de uma escola da rede pública de Pelotas, no RS. Essas aulas, seus materiais e seus conteúdos estão descritos no decorrer deste capítulo.

A atividade foi aplicada majoritariamente em dois encontros semanais, em aulas de 50 minutos, organizada em cinco tarefas, abordando conceitos e exercícios ligados a algoritmos. O detalhamento do material, planos desenvolvidos, fotos e instruções, podem ser obtidos no *site*[1] do projeto. Com o intuito de medir a aprendizagem dos alunos quanto às habilidades em algoritmos, elaborou-se um total de 4 testes, os três primeiros aplicados nas três primeiras aulas e um teste final aplicado na quinta e última aula, este envolvendo todos os conceitos abordados.

3.1. Tarefa I: Introdução ao SoccerCraft e Exercícios Iniciais

Objetivos: esta tarefa tem como finalidade oferecer uma introdução aos alunos de como é o jogo SoccerCraft em seu modo básico, apresentar alguns exemplos de execução do

[1] *https://wp.ufpel.edu.br/pensamentocomputacional/pt/*

Figura 1. Campo, jogadores e bola do jogo SoccerCraft.

jogo e realizar alguns exercícios simplificados. Aqui é introduzido comandos básicos e o conceito de sequência, despertando nos alunos o pensamento algorítmico, uma das habilidades do PC.
Material: a realização dessa tarefa utiliza alguns materiais, são eles:

- Campo de futebol fabricado em isopor ou algum material de fácil fixação de papéis com alfinetes. O campo deve ter linhas horizontais e verticais dividindo-o em quadrados. A Figura 1 exemplifica esse campo.
- Uma bola e personagens estilo Minecraft [Persson and Bergensten 2011] impressos em cartolina ou em algum material mais resistente que papel, apresentados também na Figura 1. Os jogadores preferencialmente devem ser de várias etnias e ambos sexos, a fim de facilitar a imersão dos alunos dentro do jogo. Os uniformes dos jogadores podem variar de acordo com as cores dos principais times da região para uma maior proximidade do aluno.
- Alfinetes para fixação dos jogadores e bola no campo.
- Folhas de exercícios utilizando metade do campo de jogo, cada um com um cenário diferente (posicionamento diferente dos jogadores), com o objetivo de criar um algoritmo para fazer o gol com o menor número de unidades de tempo possível.

Metodologia: inicialmente são mostrados os materiais utilizados no jogo bem como explicadas as regras do mesmo, familiarizando o aluno ao SoccerCraft. As quatro regras iniciais são: (i) só o jogador que está com a bola pode se movimentar; (ii) o jogador só pode se movimentar por exatamente um quadrado; (iii) não podem haver dois jogadores no mesmo quadrado; e (iv) um jogador somente pode fazer gol de dentro da pequena área, sendo que cada movimento conta 1 minuto (cronômetro). Nesta fase, além do jogador com a bola, há somente adversários no campo. Os alunos devem escrever um algoritmo objetivando fazer o gol com apenas 4 comandos (movimentos) possíveis, que são: correr para direita, correr para esquerda, correr para frente e chutar. Caso não haja nenhum adversário no caminho e no momento do chute o jogador estiver dentro da pequena área, ele faz o gol. Considerando o cenário de jogo da Figura 2, um exemplo de algoritmo que permite fazer gol a partir do jogador azul com a bola, seria descrita como segue: correr para direita, correr para frente, correr para frente, correr para frente, correr para esquerda, chutar. Após a explicação do jogo utilizando o material, é entregue aos alunos uma folha de exercícios contendo novos cenários para que criem algoritmos visando fazer gol usando somente os comandos (movimentos) aprendidos até esta etapa. No final da aula, os exercícios são recolhidos para posterior correção.

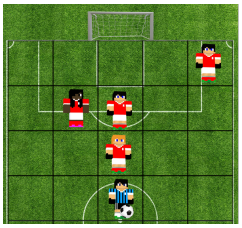


Figura 2. Exemplo de campo inicial do jogo SoccerCraft.

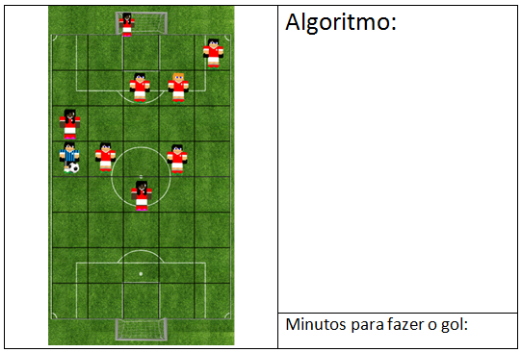


Figura 3. Exemplo de questão utilizada nos exercícios (testes) da aula 2.

3.2. Tarefa II: Incrementando o nível de Dificuldade do Jogo e a Adição do Goleiro

Objetivos: esta tarefa tem como finalidade incrementar a dificuldade do jogo SoccerCraft em relação à última aula, aumentando o tamanho do campo, adicionando o conceito do goleiro e adicionando quatro novos comandos. As habilidades do PC exploradas nesta tarefa são o pensamento algorítmico e a análise de custos.
Material: mesmos materiais da aula anterior (campo, jogadores e bola), um goleiro feito do mesmo material dos outros jogadores e também exercícios utilizando o campo completo de jogo, com diferentes cenários. A Figura 3 mostra um exemplo de exercício.

Metodologia: a atividade é iniciada relembrando aos alunos como jogar o SoccerCraft e revisando as estratégias consolidadas na última aula. O passo seguinte é a explicação de um novo comando, o "correr para trás". Agora o jogador com a bola também pode andar um quadrado para trás com o mesmo custo dos outros comandos (uma unidade de tempo). Em seguida insere-se o conceito do goleiro, localizado embaixo das traves ocupando as posições: central, direita ou esquerda, não podendo se movimentar nem receber a bola. Outros três novos comandos são adicionados: chutar para esquerda, chutar para direita e chutar para frente, sendo que antigo comando chutar é extinto. Agora, o jogador para fazer o gol deve chutar em um local onde o goleiro não se encontra posicionado. Após a explicação dos novos comandos para os alunos é proposto um novo conjunto de possíveis estratégias, descritas como exercícios, para estimular a fixação das novas regras e novos comandos. Além disso, são propostos exercícios contendo questões com o mesmo objetivo da última aula, porém mais complexos e com o campo inteiro. O objetivo será fazer o gol na goleira superior. A Figura 3 mostra um exemplo de exercício.

3.3. Tarefa III: Novos Comandos - Toco y Me Voy

Objetivos: o objetivo desta tarefa é de adicionar um novo conceito ao jogo SoccerCraft – o passe – e quatro novos comandos. O pensamento algorítmico e a análise de custo são

Figura 4. Exemplo de questão utilizada nos exercícios (testes) da aula 3.

explorados nesta tarefa. **Material:** mesmos materiais da aula anterior (campo, jogadores e bola), jogadores adversários e exercícios utilizando o campo de jogo pela metade ou completo, assim como das primeiras duas aulas, mas com a inclusão de jogadores companheiros. Um exemplo de exercício é mostrado na Figura 4.

Metodologia: a atividade é iniciada relembrando aos alunos como jogar o SoccerCraft e o que foi feito na última aula. Após acontece a introdução a um novo conceito, o passe (ou toque), e quatro novos comandos: tocar para trás, tocar para frente, tocar para esquerda e tocar para direita. Assim, além de jogadores adversários, o jogo também conta com jogadores companheiros. Agora o jogador com a bola também pode tocar a bola para um companheiro de equipe, caso ele esteja em linha reta com o jogador e não haja jogadores adversários no caminho entre eles. Caso haja um jogador companheiro, em linha reta, entre um jogador com a bola e outro jogador companheiro, ao efetuar um passe, o companheiro mais próximo é quem recebe a bola. O jogador que receber a bola é quem deve ser comandado no próximo movimento. O passe tem o mesmo custo dos outros comandos, uma unidade de tempo. Após a explicação dos novos comandos, os alunos devem realizar exercícios contendo questões com o mesmo objetivo da última aula, porém com a possibilidade de utilização dos novos comandos.

3.4. Tarefa IV: A Peleia

Objetivos: o objetivo desta aula é oferecer uma tarefa mais descontraída aos alunos, proporcionando um jogo em que potencialmente todas crianças participam, envolvendo o SoccerCraft. O pensamento algorítmico e a análise de custo são explorados nesta tarefa.
Material: mesmos materiais da aula anterior (campo, jogadores e bola). Nesta aula não há exercícios ou testes.

Metodologia: neste quarto encontro, acontece uma atividade mais descontraída, um jogo utilizando potencialmente todos os alunos da sala de aula. A turma é inicialmente dividida em duas, sendo cada metade identificando uma equipe no SoccerCraft. Caso o número de jogadores seja ímpar, o professor completa um dos times. Os alunos-jogadores ficam dispostos em dois times, de maneira espelhada pelo campo do jogo. Logo após, deve ser explicado o incremento às regras do jogo à turma, sendo que as regras vindas da aula anterior continuam vigentes: as duas equipes têm uma bola em sua posse e o jogo acontece em turnos, cada equipe tendo direito a um movimento em seu turno. Assim, a equipe que primeiro marcar o gol ganha o jogo. Quando uma equipe chega com a bola dentro da pequena área adversária, acontece um mini jogo. Nesse mini jogo, o goleiro adversário escreve sua estratégia (identificando aonde ele vai se atirar para defender o chute do adversário) em um papel e posteriormente deve escondê-la. Após isso o jogador

Figura 5. Exemplo de campo de jogo utilizado na aula 4.

com a bola deve escolher o movimento de chute (para esquerda, direita ou frente). Caso ele escolha a estratégia (mesma direção) do goleiro, o gol não é concretizado e a equipe perde o turno, continuando com a bola em posse do jogador dentro da pequena área. Caso contrário, o gol acontece e a partida acaba. A Figura 5 mostra um exemplo de campo espelhado do jogo SoccerCraft de duas equipes de 7 jogadores, envolvendo assim 14 alunos. O número de jogadores no campo pode ser maior que o número de alunos, nesse caso pode-se deixar jogadores menos importantes no campo como não-controláveis ou controláveis somente pelo professor. Caso o número de jogadores seja menor que o número de alunos, alguns alunos devem formar duplas, cada uma controlando um jogador.

O tempo de aula é todo consumido em jogos com os alunos. Vários jogos podem acontecer. O ideal é realizar um rodízio do jogador que começa com a posse de bola, fazendo com que mais alunos participem mais efetivamente dos jogos. Após a realização dos jogos é feita a contagem de quantos jogos cada time ganhou e a divulgação de um vencedor. Um prêmio pode ser dado aos alunos vencedores.

3.5. Tarefa V: Teste Final

Objetivos: esta última tarefa tem por finalidade testar os conhecimentos adquiridos pelos alunos com a aplicação de um teste final envolvendo todos os conteúdos trabalhados.
Material: para a realização dessa aula foram utilizados exercícios contendo questões sobre o jogo SoccerCraft além de questões envolvendo conceitos de algoritmos abordados em tarefas anteriores.

Metodologia: propõe-se um teste final contendo seis questões envolvendo habilidades de algoritmos previamente apresentados nas tarefas do jogo, envolvendo ou não o jogo SoccerCraft. A Figura 6 mostra exemplos de questões do teste final.

4. Validação da Proposta e Resultados Obtidos

Para validar as estratégias de aprendizagem exploradas no SoccerCraft, várias atividades foram aplicadas em uma turma do sexto ano do Ensino Fundamental da E.M.E.F. Ferreira Vianna para uma amostragem de 19 alunos. A turma era composta por 8 alunos do sexo masculino e 11 alunos do sexo feminino. Em relação ao cronograma, a atividade foi aplicada nos meses de novembro e dezembro de 2017, em cinco encontros.

Figura 6. Exemplos de questões do teste final.

Para a realização das atividades foram necessários um responsável por apresentar os conceitos e regras do jogo e até três colaboradores para auxiliar os alunos, quando requisitados. Em todos os encontros havia pelo menos um professor participante do projeto e também o(a) professor(a) da turma, no intuito de auxiliar em características individuais dos alunos.

As avaliações consistiam de um teste escrito com número variado de questões, versando sobre os temas desenvolvidos nas aulas, que totalizavam dez pontos. A descrição do desempenho dos alunos em cada avaliação da atividade é apresentada na Tabela 1. Note-se que o número de alunos avaliados variou de uma tarefa para outra em razão de eventuais ausências nos dias das avaliações.

Tabela 1. Descrição do desempenho geral dos alunos nas avaliações da atividade.

Avaliação	n	Média	DP	CV(%)	Mínimo	Q_1	Md	Q_3	Máximo
1	15	8	3,6	44,6	0,0	6,6	10,0	10,0	10,0
2	18	6,7	2,9	42,9	0,0	4,7	7,5	9,0	10,0
3	12	6,5	3,0	45,6	1,0	3,5	7,3	9,1	10,0
4	13	6,4	2,6	40,4	1,0	5,1	6,9	8,3	9,0

Nota: n = número de alunos; DP = desvio padrão; Q_1 = primeiro quartil; CV = coeficiente de variação; Md = mediana; Q_3 = terceiro quartil.

Nas avaliações da atividade SoccerCraft, segundo a Tabela1, o melhor desempenho foi alcançado na primeira avaliação, na qual a média da turma foi 8,0 e pelo menos metade dos alunos obtiveram nota 10,0. Isso demonstra a facilidade que os alunos tiveram na primeira versão do jogo. Nas três avaliações seguintes a média da turma ficou em torno de 6,5, com pelo menos metade dos alunos atingindo nota igual ou maior que 7,0, evidenciando um bom entendimento dos alunos na atividade proposta, embora tendo piores resultados se comparados à primeira avaliação. Em todas as quatro avaliações a variabilidade das notas foi elevada, com CV variando entre 40% e 45%. Essa heterogeneidade de notas mostra uma disparidade de entendimento, e/ou ainda de motivação, existente na turma.

5. Conclusão

Este trabalho apresentou a concepção, modelagem e o desenvolvimento de um jogo educacional denominado SoccerCraft, o qual explora conceitos ligados ao futebol utilizando personagens similares aos do jogo Minecraft com o objetivo de desenvolver, via estratégias do PC, a habilidade de pensamento algorítmico a alunos do sexto ano do Ensino Fundamental. Este trabalho também apresentou uma etapa de validação da proposta, na qual os resultados se mostraram bons. Esses resultados mostraram um entendimento dos alunos em relação ao conteúdo proposto, embora tenha havido uma alta discrepância entre algumas notas.

Durante o desenvolvimento das atividades, pôde-se observar um maior engajamento dos alunos nos dois primeiros encontros, aparentemente devido ao SoccerCraft ainda ser uma novidade para eles. Essa motivação diminuiu no terceiro encontro, causando desatenção e consequente dificuldade dos alunos em realizar os exercícios. Já o quarto encontro despertou novamente a motivação das crianças, devido ao espírito de competitividade explorado pela atividade.

Como trabalho futuro, pode-se popor uma continuidade no ensino dos conceitos de algoritmos, com a inserção dos comandos *if* e *while*, por exemplo, integrados ao jogo SoccerCraft.

Referências

Barbosa, D. N. F., Miorelli, S. T., Rasch, L. G., and Silva, C. G. (2017). Ensinando lógica com as tecnologias da informação: Desenvolvendo o raciocínio lógico e o pensamento computacional. *CATAVENTOS-Revista de Extensão da Universidade de Cruz Alta*, 9:54–72.

Barr, V. and Stephenson, C. (2011). Bringing computational thinking to k-12: what is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1):48–54.

Batista, E. J. S., Castro Jr, A., Larrea, A. A., and Bogarim, C. A. C. (2015). Utilizando o scratch como ferramenta de apoio para desenvolver o raciocínio lógico das crianças do ensino básico de uma forma multidisciplinar. In *Anais do WIE 2015*, volume 21, pages 350–359.

Campos, G. et al. (2014). Organização de informações via pensamento computacional: Relato de atividade aplicada no ensino fundamental. In *Anais do WIE 2014*, pages 390–399.

Carnegie Mellon (2013). Center for Computational Thinking. http://www.cs.cmu.edu/~CompThink/. Acesso: fevereiro/2018.

dos Reis, C. E. R., Duso, G. B., and Webber, C. G. (2018). Robótica educacional aplicada à simulação do sistema digestório. *Scientia cum Industria*, 5:186–192.

França, R. S., Silva, W. C., and Amaral, C. J. H. (2012). Ensino de ciência da computação na educação básica: Experiências, desafios e possibilidades. In *Proc. CSBC'12*.

Google (2013). Exploring computational thinking. http://www.google.com/edu/computational-thinking/. Acesso: fevereiro/2018.

ISTE (2013). Operational definition of computational thinking. http://www.iste.org/learn/computational-thinking/ct-operational-definition. Acesso: fevereiro/2018.

Junior, B., Cavalheiro, S., and Foss, L. (2017). A última árvore: exercitando o pensamento computacional por meio de um jogo educacional baseado em gramática de grafos. In *Anais do SBIE 2017*, volume 28, page 735.

Kafura, D. and Tatar, D. (2011). Initial experience with a computational thinking course for computer science students. In *SIGCSE 2011*, pages 251–256. ACM.

Lee, I. et al. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1):32–37.

Lugo, M. J. R. and Olabe, X. B. (2018). Diseñando un material educativo digital: nuevas formas de enseñar habilidades del pensamiento computacional. *Teknologia berrien erabilera eta gaur egungo Hezkuntza joerak Uso de nuevas tecnologías y tendencias actuales*, pages 81–94.

Marques, M., Cavalheiro, S., Foss, L., Avila, C., and Bordini, A. (2017). Uma proposta para o desenvolvimento do pensamento computacional integrado ao ensino de matemática. In *Anais do SBIE 2017*, volume 28, page 314.

Persson, M. and Bergensten, J. (2011). Minecraft. http://minecraft.net. Acesso: fevereiro/2018.

Santos, E. R. d., Soares, G., Dal Bianco, G., Rocha Filho, J. B. d., and Lahm, R. A. (2016). Estímulo ao pensamento computacional a partir da computação desplugada: uma proposta para educação infantil.

Silva, V., da Silva, L. L., and França, R. (2017). Pensamento computacional na formação de professores: experiências e desafios encontrados no ensino da computação em escolas públicas. In *Anais do WIE 2017*, volume 23, page 805.

Teixeira, A., Martins, J. R., Batistela, F., Pazinato, A., and Oro, N. (2015). Programação de computadores para alunos do ensino fundamental: A escola de hackers. In *Anais do WIE 2017*, volume 1, pages 144–163.

TNH1 (2017). Confira os 10 games mais jogados do mundo. http://www.tnh1.com.br/noticias/noticias-detalhe/cultura/confira-os-10-jogos-de-videogame-mais-jogados-do-mundo/. Acesso: fevereiro/2018.

UFPel (2018). EXP-PC - Explorando o Pensamento Computacional para a Qualificação do Ensino Fundamental. http://wp.ufpel.edu.br/pensamentocomputacional. Acesso: fevereiro/2018.

Weisshahn, Y., Pinho, G., Cavalheiro, S., Du Bois, A., Aguiar, M., Foss, L., Reiser, R., and de Brum, C. F. (2016). Representação e análise de dados no quinto ano do ensino fundamental: Proposta de atividade e relato de aplicação. In *Anais do WIE 2016*, volume 22, page 201.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Zanetti, H. A. P., Borges, M. A. F., Leal, V. C. G., and Matsuzaki, I. Y. (2017). Proposta de ensino de programação para crianças com scratch e pensamento computacional. *Tecnologias, Sociedade e Conhecimento*, 4(1):43–58.

Um Simulador Educacional para Apoio ao Projeto de Sistemas Computacionais: Hardware, Software e suas Interfaces

Guilherme Esmeraldo[1], Lucas Fontes[1], Cícero Samuel Mendes[1], Edson Lisboa[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Crato

[2]Instituto Federal de Educação, Ciência e Tecnologia de Sergipe – Aracaju

guilhermealvaro@ifce.edu.br, lfonteesc@gmail.com,
mendes.samuel99@gmail.com, edson.lisboa@academico.ifs.edu.br

Abstract. *This paper presents a graphical simulator for educational purposes in design of computational systems, covering hardware architecture and organization, low level programming and its interfaces. The proposed simulator includes graphical and interactive features aiming to simplify configuration, programming, simulation, visualization and performance evaluation of a complete system. The proposed tool abstracts interfaces and behaviors of its components to make learning more attractive, but without compromising the quality of the learned concepts. The proposed application has been evaluated in two classes of a Computer Organization and Architecture course and results showed that the learning became more attractive, dynamic and effective.*

Resumo. *Este artigo apresenta um simulador gráfico para apoio educacional ao projeto de sistemas computacionais, abrangendo arquitetura e organização de hardware, software de baixo nível e suas interfaces. O simulador proposto inclui recursos gráficos e interativos para simplificar a configuração, programação, simulação, visualização e avaliação de desempenho de um sistema completo. A ferramenta proposta abstrai interfaces e comportamentos de seus componentes para tornar o aprendizado mais atrativo, porém sem comprometer a qualidade dos conceitos apreendidos. A aplicação proposta foi avaliada em duas turmas de Organização e Arquitetura de Computadores e os resultados mostraram que o aprendizado tornou-se mais atrativo, dinâmico e efetivo.*

1. Introdução

A evolução tecnológica e a consequente alta escala de integração no projeto de circuitos

digitais (CIs) têm disponibilizado soluções de hardware com alto poder computacional, associada a um aumento substancial da sua complexidade. Exemplos de tais dispositivos são: SoC e MPSoC (sistemas com múltiplos processadores em um único chip), dispositivos de eletrônica reconfigurável (FPGA), sensores inteligentes, dentre outros [Ghaffarian 2016].

Essa evolução impacta diretamente no estudo dos aspectos e funcionamento do computador, em cursos de Computação, o qual não só beneficia àqueles que desejam desenvolver novos sistemas computacionais, mas também como apoio na aprendizagem e compreensão de lógica de programação [Zeferino et al. 2012], de conceitos de sistemas operacionais [Mustafa 2013], para o projeto de novas interfaces de interação humano-computador [Larraza-Mendiluze and Garay-Vitoria 2015] e aumento de desempenho dos sistemas de software [Bertazi, Auler and Borin 2014].

No entanto, esse estudo não é uma tarefa simples, pois além de envolver os conteúdos constantes de disciplinas de Organização e Arquitetura de Computadores – que basicamente abordam o projeto de sistemas digitais, as estruturas de componentes de hardware, seu funcionamento e interconexões, bem como programação em baixo nível –, também deve estar alinhado com as novas tendências em projetos de sistemas computacionais, como são os casos da Internet das Coisas (IoT), Computação de Alto Desempenho e Automação. Isso tem implicado em uma demanda crescente por formas mais efetivas na transferência desses conhecimentos.

Na literatura, com frequência tem-se utilizado simuladores como abordagem de apoio ao aprendizado em projeto de sistemas computacionais. Podem ser encontrados simuladores que buscam abstrair os detalhes de implementação para aumentar o desempenho das simulações e que abordam desde componentes específicos de hardware, para tratar de conceitos específicos ou mais avançados –, como são os casos dos simuladores de processadores e de memórias cache [Xavier, Rodrigues and Júnior 2011] –, até os de sistemas completos [Penna and Freitas 2013], que trazem uma visão macro das funcionalidades e de comunicação entre componentes. Outros trabalhos incluem simuladores ao nível de hardware, como os descritos em linguagem de descrição de hardware (HDL) [Awedh and Mueen 2015], que visam abordar diferentes níveis de abstração no projeto de um sistema computacional, e os que utilizam componentes físicos de hardware [Black 2016], os quais fazem com que os estudantes sejam motivados a produzirem sistemas digitais reais.

As abordagens que trabalham em nível de hardware são particularmente interessantes, pois, uma vez que lidam com componentes eletrônicos reais, permitem compreender com maior profundidade os aspectos relacionados aos subsistemas de entrada/saída, como, por exemplo, os diferentes tipos e características dos periféricos, entrada/saída programada e o uso de interrupções para sincronização e aumento de

desempenho. O estudo apresentado em [Larraza-Mendiluze and Garay-Vitoria 2015] mostra que há lacunas no processo de ensino-aprendizagem em subsistemas de entrada/saída devido aos métodos de ensino empregados que limitam-se às aulas teóricas e ao uso de simuladores em alto nível de abstração.

Diante do cenário apresentado pelas várias soluções analisadas, este trabalho apresenta uma ferramenta, denominada CompSim, que implementa recursos tanto para realizar simulações quanto para tratar com hardware real, abordando todos os aspectos de uma plataforma computacional completa, em diferentes níveis de abstração. Suas funcionalidades buscam simplificar o aprendizado em projetos de sistemas computacionais e, do ponto de vista de simulação, ela disponibiliza recursos gráficos que permitem configurar, programar, simular, visualizar e avaliar o desempenho de uma plataforma de hardware simulável, a qual inclui diferentes componentes de um computador. Considerando os aspectos de hardware real, o CompSim objetiva preencher a lacuna no aprendizado em subsistemas de entrada/saída, através de uma interface de integração com a plataforma Arduino UNO e, com isso, estimular o público-alvo a criarem projetos utilizando, e reutilizando, soluções de sistemas eletrônicos reais.

O artigo está estruturado da seguinte maneira: Na seção a seguir, apresenta-se o simulador CompSim e seus principais recursos; A Seção 3 apresenta os resultados de avaliações do CompSim, pelos estudantes; e, Na Seção 4, são demarcadas as considerações finais.

2. O Simulador CompSim

CompSim é um simulador de sistema completo, o qual inclui uma plataforma de hardware simulável, conhecida por “Mandacaru”, que é composta pelos seguintes componentes: um modelo de processador conceitual, memórias cache e RAM, barramentos de sistema e de periféricos, um dispositivo virtual de entrada (Teclado) e outro de saída (Vídeo). A Figura 1 ilustra a plataforma Mandacaru e seus componentes.

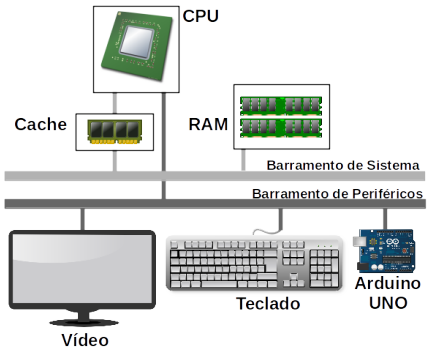


Figura 1. Plataforma Mandacaru e seus componentes.

A plataforma Mandacaru possui ainda uma interface de Entrada/Saída (E/S) para

integração com uma plataforma Arduino UNO e, com isso, interfacear com periféricos físicos reais.

O processador conceitual, chamado de "Cariri", possui uma arquitetura típica de processadores de 16-bits reais, implementada pelos seguintes componentes: Unidade Lógica e Aritmética (ULA), Unidade de Controle (UC), registradores específicos e de propósito geral, 16 instruções que envolvem operações lógicas e aritméticas, transferência de dados, transferência de controle e entrada/saída, bem como operandos inteiros com sinalização (*signed int*) e cadeias de bytes (*strings*). Ele foi concebido para simplificar o aprendizado dos conteúdos relacionados à Arquitetura de Computadores, como, por exemplo, as fases do ciclo de instrução, representação de dados numéricos, não-numéricos e de estruturas de dados, conjunto de instruções da arquitetura (tipos, tamanhos e formatos de instruções), modos de endereçamento, programação em nível de máquina, representação de construções de linguagens de alto nível, entre outros.

Os componentes propostos para memória cache e RAM, assim como no processador Cariri, incluem características de componentes reais, as quais foram definidas para suportar o estudo de hierarquia e organização de memórias, técnicas de mapeamento e políticas de atualização/substituição de cache e cálculo de tempos de acesso aos dados.

Os demais componentes, barramentos e periféricos, podem ser utilizados para o estudo dos fundamentos de comunicação e de entrada/saída, como tipos e protocolos de *handshaking* de barramentos, entrada e saída programada e orientada à interrupção, interface e módulo de entrada/saída e cálculos dos tempos para entrada/saída. Além disso, o CompSim possibilita a integração com a plataforma Arduino UNO, que tem sido muito usada em soluções de sistemas embarcados e Internet das coisas, através da conexão ao barramento de periféricos da plataforma Mandacaru. Este recurso da ferramenta permite interfacear com periféricos físicos reais e realizar computação em tempo real integrada ao ambiente de simulação, possibilitando estudos e práticas de conceitos básicos de eletrônica, microcontroladores e projeto de sistemas eletrônicos completos. Esse suporte agrega escalabilidade de conhecimento a uma visão moderna do processo de ensino de arquitetura e organização de computadores.

O CompSim inclui também uma interface gráfica (*Graphical User Interface* - GUI) projetada para apoiar as atividades de configuração da plataforma Mandacaru, programação do processador Cariri, gerenciamento de simulação, visualização da simulação e análise de desempenho de uma aplicação. As subseções a seguir detalham os principais recursos gráficos para apoio a cada uma dessas atividades.

2.1. Configuração dos Componentes da Plataforma Mandacaru

O CompSim inclui um painel de configuração de plataforma, que pode ser visto na

Figura 2, onde nele é possível configurar o número de blocos da memória RAM, bem como técnicas de mapeamento (direto, associativo e associativo por conjunto), políticas de atualização (*write-through*, *write-back*, *write allocate* e *write around*) e de substituição (FIFO, menos recentemente utilizada e aleatória) e número de linhas de Cache. Ainda é possível configurar a porta serial para conexão de uma plataforma Arduino UNO e, com isso, integrá-la à plataforma Mandacaru.

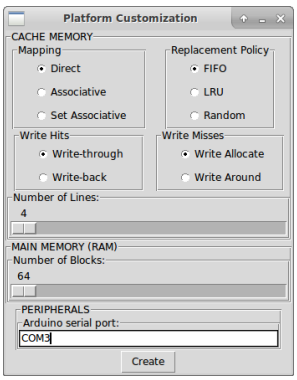
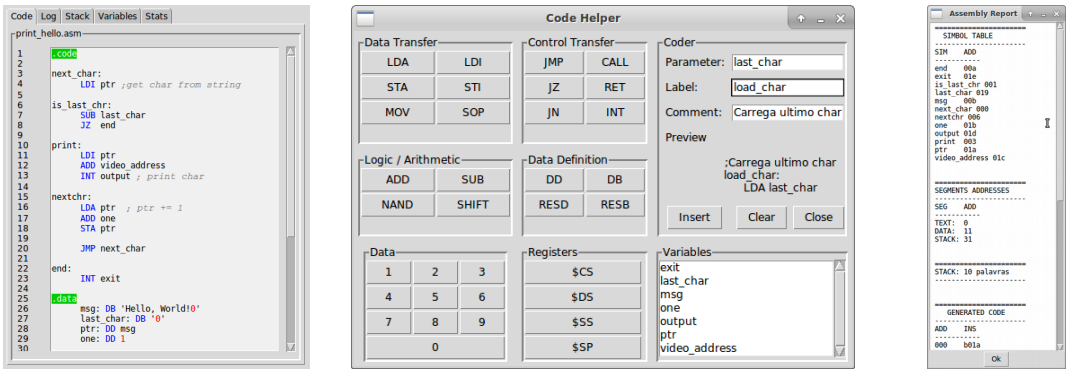


Figura 2. Painel de Configuração da Plataforma Mandacaru.

2.2. Programação do Processador Cariri em Nível de Máquina

O CompSim conta com os seguintes componentes gráficos para apoio à programação do processador Cariri: 1) Code Editor (Figura 3(a)): suporte à codificação de uma nova aplicação em baixo nível para execução no simulador. Inclui recursos de *sintax highlight*, exibe número de linhas e de nome de arquivo em edição, recursos de Undo/Redo e Cut/Copy/Paste, teclas de atalho, entre outros. O editor está integrado ao Assembler da linguagem do processador Cariri (em caso de erros no código, será destacada a linha e coluna do primeiro erro); 2) Code Helper (Figura 3(b)): assistente para auxílio na construção das instruções do programa com a sintaxe correta. Inclui atalhos para todas as instruções do processador Cariri, para números e nomes de registradores e de variáveis do programa. 3) Asm Report (Figura 3(c)): componente integrado ao Assembler, realiza análise léxica, sintática e semântica, e gera relatório com tabela de símbolos, endereços de memória dos segmentos de código, dados e pilha, tamanho da pilha de programa e o respectivo código de máquina.



(a) Code Editor (b) Code Helper (c) Asm Report

(a) Code Editor(b) Code Helper(c) Asm ReportFigura 3. Componentes gráficos para apoio à programação e análise de código.

2.3. Gerenciamento de Simulação

Após a configuração da plataforma, codificação e validação da aplicação, o passo seguinte consiste em configurar e executar a simulação. Compsim dispõe de controles para configuração e gerenciamento das simulações, sendo possível configurar o tempo de simulação e frequência de clock do sistema, bem como executar, interromper e reiniciar a simulação, como pode ser visto na Figura 4 (a).

2.4. Visualização de Simulação

Durante uma simulação, é possível acompanhar os estados dos componentes de hardware, através dos componentes gráficos: 1) CPU (Figura 4(b)): exibe os estados dos registradores do processador e distingue, em cores diferenciadas, cada um dos registradores de endereçamento à memória; 2) CACHE (Figura 4(c)): exibe os estados das linhas de memória cache, em tempo de execução, destacando a linha e a coluna de acordo com palavra buscada pelo processador; 3) RAM (Figura 4(d)): exibe os dados nos diferentes endereços da memória principal, destacando as posições referenciadas pelos registradores de endereçamento do processador Cariri, de acordo com as respectivas cores. É possível, através dos componentes *Input Buffer* (Teclado) (Figura 4(e)) e *Output Buffer* (Vídeo) (Figura 4(f)), interagir (Entrada/Saída) com uma aplicação, durante a simulação.

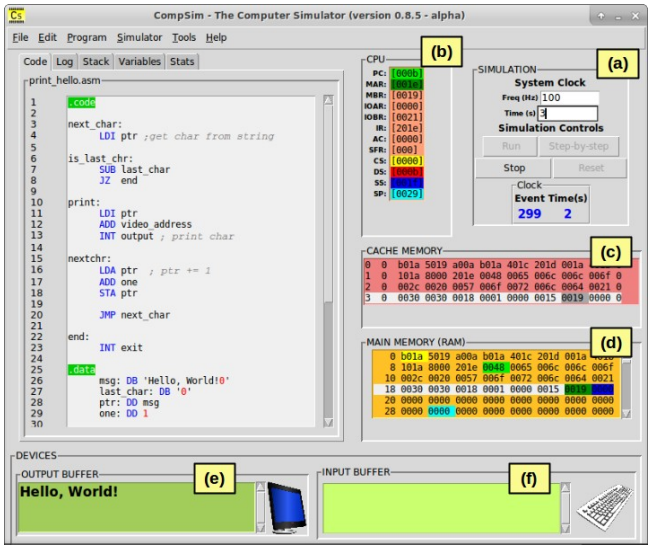
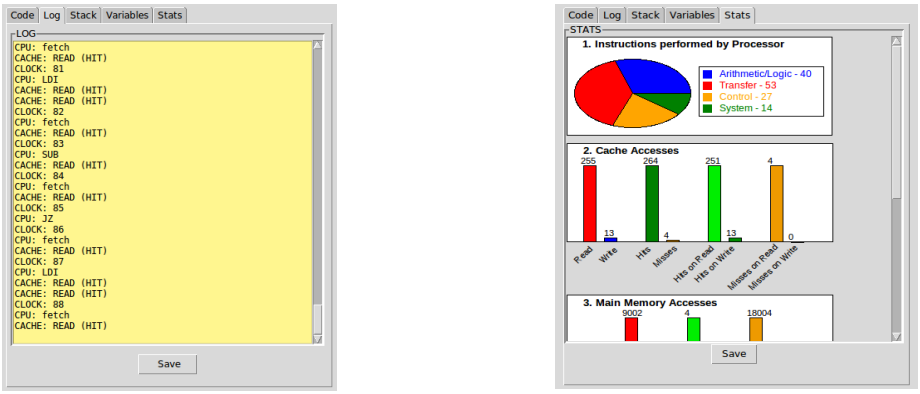


Figura 4. Componentes gráficos para visualização dos estados dos componentes da plataforma Mandacaru.

Para acompanhar a execução da aplicação, CompSim apresenta os componentes gráficos *Variables* para exibir os valores assumidos pelas variáveis e estruturas de dados do programa, e *Stack* para exibir o conteúdo da pilha do programa.

2.5. Análise de Desempenho da Aplicação

CompSim disponibiliza dois componentes para análise de desempenho da aplicação, após uma simulação (eles estão ilustrados na Figura 5).



(a) Logs de execução. **(b) Gráficos estatísticos.**

Figura 5. Componentes gráficos para análise de desempenho da aplicação.

O primeiro deles, Logs na Figura 5 (a), exibe informações de eventos de simulação gerados pelos componentes da plataforma. Já o segundo, Stats na Figura 5 (b), exibe gráficos estatísticos que sumarizam: quantidades e tipos de instruções

executadas pelo processador; leituras/escritas, taxas de hits/misses, hits/misses em leitura e em escrita da memória cache; palavras e blocos lidos/escritos na memória RAM; e operações de leitura/escrita em memória e periféricos nos barramentos. Tanto os logs de simulação, quanto os gráficos estatísticos podem ser exportados para arquivos em disco.

3. Resultados Experimentais

O simulador aqui proposto foi utilizado em práticas laboratoriais por duas turmas da disciplina de Organização e Arquitetura de Computadores de um curso de Bacharelado em Sistemas de Informação.

Ao final dos semestres letivos, nas duas turmas, 26 estudantes avaliaram o CompSim, através de uma rubrica [Yuan and Recker 2015] com os seguintes indicadores: 1) Conteúdo curricular: relação entre as práticas realizadas com os componentes curriculares da disciplina; 2) Pensamento de alto nível: possibilidade de construção de novos conhecimentos, através da análise, avaliação e síntese; 3) Nível de Graduação: nível de conteúdos abordados no simulador é apropriado para o público-alvo; 4) Componentes de avaliação: suporte para autoavaliação do nível de aprendizado dos estudantes; 5) Efetividade no ensino: suporte efetivo no processo de ensino-aprendizagem dos conteúdos abordados; 6) Desempenho do estudante: taxa de aprendizado atende às necessidades individuais dos estudantes; 7) Interatividade: relação entre a atividade do usuário com o sistema e o suporte ao estímulo/motivação para estudos; 8) Amigabilidade: design limpo, fácil de utilizar e dispõe de meios para auxílio ao estudante no aprendizado; 9) Acessibilidade: informações gráficas são devidamente rotuladas, fontes são consistentes e fáceis de ler, e se estão sendo empregados diferentes estilos de aprendizagem e níveis de habilidade. Cada um dos indicadores incluiu os seguintes aspectos de qualidade, com respectivos escores: Ruim (1 ponto), Razoável (2 pontos), Bom (3 pontos) e Excelente (4 pontos).

Os resultados na Figura 6 mostram que, em média, os estudantes avaliaram a qualidade todos os indicadores entre “bom” e “excelente”. Entre os indicadores, segundo os estudantes, “Pensamento de alto nível” obteve a maior média (3,9), indiciando assim a efetividade da abordagem de ensino-aprendizagem com apoio do simulador aqui proposto. Por outro lado, os estudantes apontaram “Interatividade” como o de menor média (3,15) e fizeram algumas sugestões de melhorias, como a adição de maior *feedback* na interface e a possibilidade de inclusão de temas para personalização da interface gráfica, ambos para aumentar a qualidade da experiência de uso do simulador. Os estudantes sugeriram ainda a criação de novas mídias complementares, como tutoriais e vídeos, para orientação na programação do processador Cariri e para uso do CompSim.

Figura 6. Resultados da Avaliação do CompSim.

Ressalta-se a grande produtividade dos alunos no desenvolvimento de aplicações computacionais com o uso do simulador CompSim, as quais trataram: 1) Operações matemáticas: multiplicação e divisão utilizado somas ou subtrações sucessivas ou com deslocamento de bits; 2) Construções de linguagens de programação de alto nível: IF/ELSE, SWITCH/CASE e FOR/WHILE; 3) Funções diversas, com passagem e retorno de parâmetros e recursividade: máximo divisor comum, contagem de números primos, geração de sequência de Fibonacci e busca linear; 4) Manipulação de estruturas de dados: ordenação, cópia e soma de vetores; 5) Rotinas de entrada/saída: leitura e escrita de números inteiros sinalizados; 6) Projetos de sistemas eletrônicos: controle de LEDs com botões, controle de luminosidade de LEDs com potenciômetros, uso de displays de 7 seguimentos, entre outros.

4. Conclusões

Este artigo apresentou o CompSim, uma ferramenta para suporte ao aprendizado em projetos de sistemas computacionais. O simulador proposto inclui plataforma computacional virtual e uma interface gráfica integrada, a qual permite simplificar a configuração dos componentes de hardware, programação do processador, execução, visualização e avaliação de desempenho de novos sistemas computacionais. O CompSim foi utilizado nas práticas laboratoriais e avaliado por duas turmas de uma disciplina de OAC. Os resultados mostraram que a ferramenta proposta apresentou suporte educacional efetivo e qualidade na experiência de uso.

Os trabalhos futuros incluem: a adição de novos componentes à plataforma Mandacaru, para disponibilizar cenários reais e fluxo completo de projetos de sistemas computacionais; o suporte de diferentes tipos de dados, um compilador de linguagem de alto nível e um sistema operacional de tempo real para o processador Cariri; e a criação de cenários para o projeto de novos sistemas eletrônicos reais.

Agradecimentos

Agradecemos a IFS/PROPEX/Petrobrás e IFCE/FUNCAP por fomentarem esta pesquisa através da concessão de bolsas e apoio financeiro.

Referências

Awedh, M., Mueen, A. (2015) "Teaching Computer Organization Using Field Programmable Gate Array: An Incremental Approach". Asian Journal Of Advanced Basic Sciences, Vol. 4, p.5-11.

Bertazi, G., Auler, R., Borin, E. (2014) "Uma plataforma para o ensino de organização de computadores e linguagem de montagem". International Journal of Computer Architecture Education, Vol. 3, p.13.

Black, M. (2016) "Export to arduino: a tool to teach processor design on real hardware". Journal of Computing Sciences in Colleges, 31(6), p.21-26.

Ghaffarian, R. (2016) "Microelectronics packaging technology roadmaps, assembly reliability, and prognostics". *Facta universitatis-series: Electronics and Energetics*, *29*(4), pp.543-611.

Larraza-Mendiluze, E., Garay-Vitoria, N. (2015) "Approaches and tools used to teach the computer input/output subsystem: A survey". IEEE Transactions on Education, 58(1), pp.1-6.

Mustafa, B. (2013) "YASS: A System Simulator for Operating System and Computer Architecture Teaching and Learning". European Journal of Science and Mathematics Education,1(1), p.34-42.

Penna, P. H. M. M., Freitas, H. C. (2013) "Análise e Avaliação de Simuladores de Sistemas Completos para o Ensino de Arquitetura de Computadores". International Journal of Computer Architecture Education, Vol. 2, p.13. Xavier, M. A. S., Rodrigues, J. C., Júnior, O. A. L. (2011) "Simuladores de Memória Cache, um Estudo Comparativo Direcionado ao Ensino", In: Workshop sobre Educação em Arquitetura de Computadores (WEAC 2011), p. 7-12.

Yuan, M., Recker, M. (2015) "Not all rubrics are equal: A review of rubrics for evaluating the quality of open educational resources. The International Review of Research in Open and Distributed Learning, 16(5).

Zeferino, C. A., Raabe, A. L. A., Vieira, P. V. and Pereira, M. C. (2012) "Um enfoque interdisciplinar no ensino de arquitetura de computadores". C. Martins, P. Navaux, R. Azevedo, S. Kofuji. Arquitetura de Computadores: educação, ensino e aprendizado. Editora SBC.

Um ensaio sobre a experiência educacional na programação de computadores: a abordagem tradicional versus a aprendizagem baseada em projetos

Alexandre Grotta[1], Edmir P. V. Prado[1]

[1]Programa de Pós-graduação em Sistemas de Informação – Universidade de São Paulo
Rua Arlindo Béttio, 1000, São Paulo – SP, 03828-000

grotta@ifsp.edu.br, eprado@usp.br

Abstract. *Towards the innovation on computer programming education, there are reports about changes of two educational aspects. First, the use of constructivist teaching methods like project-based learning (PjBL). Second, the measuring non-mandatory metrics like motivation. But there are also reports about the challenges on measuring and evaluating these changes. Thus, given the educational continuum on computer programming education, what and how to evaluate, regarding the teaching and learning via PjBL versus the traditional educational experience? Based on the literature review, this essay proposes a common pattern to evaluate them. Regarding the teaching, we propose to evaluate the teaching methods and the educational resources. Regarding learning results, we propose to evaluate supportive metrics, like the motivation to learn and the cognitive gap. As future discussions, this pattern might be extended to related computational education and future research.*

Resumo. *Para inovar na educação em programação de computadores, há relatos de mudanças em dois aspectos educacionais. Primeiro, uma maior utilização de métodos construtivistas de ensino, tais como a aprendizagem baseada em projetos (PjBL). Segundo, a introdução de avaliações auxiliares, tais como a motivação dos alunos. No entanto, há também relatos de desafios para a avaliação destes dois aspectos. Assim, dado o continuum educacional de programação de computadores, o que avaliar, e como avaliar, no ensino PjBL, com relação à experiência educacional tradicional ? Embasado em uma revisão de literatura, este ensaio propõem um padrão para estas avaliações, a partir da visão do ensino e da aprendizagem. Primeiro, com relação ao ensino, avaliar métodos e recursos educacionais. Segundo, com relação à aprendizagem, utilizar avaliações auxiliares, tais como a motivação para aprender ou a lacuna cognitiva. Por fim é discutida a possibilidade de que este padrão venha a ser estendido ao ensino computacional e a pesquisas futuras.*

1. Introdução

Elevadas taxas de evasão têm atingido o ensino computacional de nível superior, tais como cursos de Ciência da Computação e de Sistemas de Informação, inclusive no contexto de países desenvolvidos. A educação computacional é desafiante. A aprendizagem nesta área do conhecimento usualmente requer maior capacidade técnica dos alunos. Neste contexto, o ensino e aprendizagem de programação de computadores (EA-PROG) é uma área fundamental e também desafiadora. O EA-PROG é complexo e laborioso. Demanda alta

carga cognitiva dos alunos e possui longa curva de aprendizagem. Devido a estas características, há relatos de um alto nível de evasão dos alunos em cursos introdutórios. Além destes desafios, as novas gerações de alunos irão programar *smartphones*, robôs, entre outras máquinas computacionais cada vez mais presentes na sociedade, em complemento aos computadores como são atualmente conhecidos [Chan Mow 2008; Giraffa et al. 2014; Iskander 2008; Queirós 2014; Silva Filho et al. 2007].

Diante deste cenário, há dois relevantes aspectos para pesquisas educacionais no EA-PROG. Primeiro, há um forte debate na educação a respeito dos méritos da educação construtivista versus a educação instruída. Segundo, há relatos da utilização de avaliações auxiliares no EA-PROG, tais como a motivação dos alunos [Queirós 2014; Boruchovitch 2008].

A abordagem construtivista, também referenciada como centrada no aluno, tende a diminuir a lacuna cognitiva existente entre diferentes alunos, pois utiliza o nível cognitivo mais espontâneo e próximo aos estudantes. De fato, a educação pode ser definida a partir de polos bastantes distintos, até mesmo opostos entre si: o ensino instrutivista e o ensino construtivista. Estes polos do ensino são denominados visões ou abordagens. Cada uma dessas duas visões é aplicada na educação por meio de métodos específicos de ensino, no qual a Aprendizagem Baseada em Projetos (PjBL) é um representante da visão construtivista. Por outro lado, além das inovações nos métodos de EA-PROG, há também relatos de pesquisas que utilizaram avaliações auxiliares, tais como a motivação para aprender e o engajamento dos alunos, em complemento às avaliações educacionais tradicionais, tais como as notas [Biggs and Tang 2011; Boruchovitch 2008; Fior and Mercuri 2013; Goulding 2013; Guzman-Ramirez and Garcia 2013; Manogaran 2013; Payne 2009].

Com base em uma revisão de literatura, este ensaio propõe um padrão para as avaliações no EA-PROG. Primeiro, é proposto avaliar o ensino por meio da experiência educacional, seus métodos e recursos educacionais. Para avaliar os resultados estudantis, é proposto a utilização de avaliações auxiliares, tais como a motivação, o engajamento ou a lacuna cognitiva dos alunos. Em complemento a esta seção, o ensaio possui outras seis seções. A Seção 2 apresenta a fundamentação teórica. A Seção 3 apresenta o objetivo do ensaio, seguida pela proposta do ensaio na Seção 4. As limitações e discussões estão disponíveis na Seção 5. Na Seção 6 são tecidas as conclusões.

2. Fundamentação Teórica

Esta seção apresenta os conceitos fundamentais utilizados por este ensaio. No EA-PROG é plausível considerar o instrutivismo e o construtivismo como duas metáforas que ajudam a compor a visão do todo, ocupando os extremos do *continuum* educacional. Para compreender o todo, se faz necessário primeiramente entender estes dois polos [Payne 2009; Porcaro 2011]. Assim, a subseção 2.1 dispõe uma breve introdução às visões instrutivista e construtivista no EA-PROG. A subseção 2.2 detalha PjBL como método educacional construtivista no EA-PROG. Por fim, a subseção 2.3 fundamenta os fatores tradicionais de avaliação da aprendizagem.

2.1. Ensino Instrutivista de Programação de Computadores

A visão instrutivista é assim denominada pois o ensino ocorre de modo instruído. Há um fluxo de aprendizagem do educador para os educandos, instruindo-lhes novos

conhecimentos. Também é referenciada como abordagem centrada no professor, bem como abordagem tradicional ou clássica, pois é a predominante no ambiente educacional. A abordagem instrutivista também pode ser referenciada como passiva, uma vez que o aluno recebe a instrução [Biggs and Tang 2011; Payne 2009].

De modo similar ao que ocorre na educação em geral, a literatura aponta que o EA-PROG é influenciado pelas visões instrutivista e construtivista de ensino. A visão tradicional do EA-PROG continua sendo a instrutivista. As aulas consideradas tradicionais em EA-PROG são instrutivistas: expositivas, usualmente seguidas por provas de validação de entendimento dos conceitos por meio de questões. O EA-PROG tradicional é voltado para atividades de escrever códigos e aprender a sintaxe da linguagem de programação. Tende a ser verticalizado, com foco em determinada disciplina ou tecnologia, sem integrar diversos tópicos ou mesmo integrar disciplinas entre si [Corno and De Russis 2017; Goulding 2013; Greening 2012].

Por outro lado, o construtivismo tem sido uma das filosofias de maior influência na educação recente. A abordagem construtivista é referenciada como aprendizagem ativa, aprendizagem construída ou centrada no aluno, pois a educação se constrói a partir do aluno. No EA-PROG, o ensino construtivista se apresenta como alternativa ao ensino tradicional. Estudos apontam diversas vantagens da abordagem construtivista sobre a abordagem instrutivista para a educação na área de tecnologia. Exemplos de métodos de ensino construtivistas são: discussões iterativas, aprendizagem coletiva, aprendizagem baseada em problema (PBL), aprender pelo fazer (do inglês *learn by doing*, tradução nossa), com especial relevância para PjBL [Beck and Kosnik 2012; Biggs and Tang 2011; Greening 2012; Guzman-Ramirez and Garcia 2013; Payne 2009; Zhang and Liu 2012].

2.2. O que é Aprendizagem Baseada em Projetos

PjBL pode ser definida como a teoria e a prática da utilização de projetos próximos à vida real dos estudantes. Tem como objetivo facilitar a aprendizagem, individual e coletiva, maximizando-se a performance em um ambiente de aprendizagem por meio do fazer. Há inclusive uma forte concordância sobre os grandes potenciais educacionais da PjBL para a educação tecnológica. PjBL é baseada no conceito construtivista de aprendizagem. PjBL também pode ser definida como uma abordagem de ensino e aprendizagem que se propõe a engajar alunos em um contexto autêntico, exploratório e orientado à resolução de problemas [DeFillippi 2001; Guzman-Ramirez and Garcia 2013; Mioduser and Betzer 2007; Romeike and Göttel 2012]. Neste sentido, projetos e produtos finais guiam a construção do conhecimento, pautados em contextos próximos à realidade dos alunos, de modo colaborativo, envolvendo-os em investigações e atividades de design, como apoio de tecnologias de aprendizagem e recursos educacionais voltados a PjBL.

2.3. Fatores para avaliação de aprendizagem em programação de computadores

Esta subseção fundamenta dois fatores para avaliação da aprendizagem. Primeiro, a avaliação do rendimento escolar e sua relação com o ensino tradicional. Segundo, avaliações auxiliarias, tais como a motivação para aprender dos alunos. O rendimento escolar, por vezes representado pelas notas, é uma avaliação fundamental dentro do ambiente educacional. Atrelado ao rendimento escolar, é necessário avaliar a presença dos alunos no contexto do ensino presencial [Luckesi 2008]. No Brasil, esta prática é regulamentada pela Lei de Diretrizes e Bases da Educação Nacional, cujo objetivo final é verificação do acúmulo de conhecimento adquirido, bem como a avaliação da

frequência mínima requerida por Lei [BRASIL 1996]. Portanto, o rendimento escolar e a frequência escolar são avaliações indispensáveis no contexto do ensino superior presencial de computação.

Por outro lado, avaliações auxiliares têm sido relevantes no contexto internacional na educação em vários segmentos de escolarização. A motivação para aprender (doravante denominada motivação) é uma das principais avaliações auxiliares, portanto abordada por este ensaio. A motivação possui características preditivas de eventos no contexto escolar [Boruchovitch 2008]. Já o engajamento pode ser definido como "a relação que se estabelece entre o aluno e uma atividade escolar" (STELKO-PEREIRA; VALLE; WILLIAMS, 2015, pág. 207). Além disso, é aceita a correlação entre motivação, engajamento escolar e interesse escolar [Fior and Mercuri 2013].

3. Objetivo

Com base no contexto apresentado, o objetivo deste ensaio é propor um padrão para o que avaliar, e como avaliar, no EA-PROG por meio de PjBL, quando comparado ao EA-PROG tradicional. Para tornar a proposta mais compreensível, parte-se de um exemplo teórico e dois perfis de alunos T e F, todos obtidos da obra de Biggs e Tang (2011) e adaptados ao contexto de uma disciplina de ensino de programação. O primeiro perfil, nomeado F, possui menor afinidade acadêmica e um histórico de pior rendimento escolar. Nas aulas tradicionais, o perfil F não faz muitas perguntas pois seu conhecimento prévio é mais limitado: seu objetivo é basicamente ser aprovado. O segundo perfil, nomeado T, é mais acostumado ao meio acadêmico e possui um bom histórico de rendimento. Utiliza seu conhecimento prévio para formular questões e acompanhar o que está sendo ensinado nesta aula tradicional. O perfil T parece mais engajado às aulas tradicionais de EA-PROG do que o perfil F, conforme observado no lado direito da figura 1 a seguir:

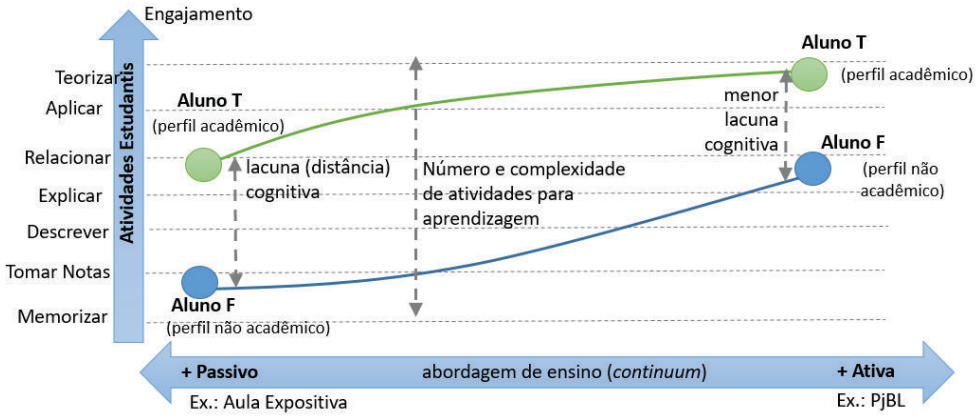


Figura 1. *Continuum* educacional aplicado à educação em programação

Na figura 1 nota-se uma relação direta entre os métodos de ensinos adotados, no eixo X, e respectivos fenômenos, visualizados no eixo Y. Assim, ao tornar a experiência educacional mais construtivista por meio de PjBL há uma sinergia e vários aspectos dos alunos melhoram: aumenta a capacidade de realização de atividades educacionais, aumenta o engajamento e diminui a lacuna cognitiva. Os benefícios tendem a retirar o perfil F da região de notas de reprovação. Assim, diante das modificações de métodos no *continuum* educacional de programação de computadores, o que avaliar, e como avaliar, na experiência educacional tradicional versus PjBL ?

4. O ensaio

Nesta seção, são apresentadas as propostas para o que avaliar e como avaliar relativas ao EA-PROG. Iniciando pela aprendizagem, a tabela 1 resume a proposta das avaliações auxiliares e alguns exemplos ilustrativos. Sugere-se ainda que as avaliações auxiliares sejam divididas em três categorias: do indivíduo, dos grupos e do coletivo. Além disso, propõe-se que tais avaliações auxiliares sejam classificadas em dois tipos. Primeiro tipo, avaliações psicométricas, tais como motivação e engajamento. Segundo tipo, avaliações de desempenho do coletivo, a exemplo da lacuna cognitiva em Biggs e Tang (2011).

Tabela 1. Proposta para avaliação da aprendizagem em programação de computadores

Abordagem	Medida Individual	Medida de Grupo	Medida do Coletivo
Tradicional	**O que:** Nota, atividades estudantis. **Como:** Prova, teste, relatório, apresentação ou trabalho individual.	**O que:** Nota, atividades estudantis. **Como:** Trabalho, apresentação, seminário, provas em grupo.	Não encontrado.
PjBL	**Avaliações auxiliares:** avaliações psicométricas individuais, tais como motivação, engajamento, comunicação, capacidade de planejamento, de execução, de contribuição ao coletivo. Adicionalmente, lacuna cognitiva do indivíduo para com o grupo. **Como:** Construtos e/ou projetos.	**Avaliações auxiliares:** psicométricas coletivas, tais como capacidade de trabalho em equipe, de convencimento. Adicionalmente, lacuna cognitiva entre grupos. **Como:** Construtos e/ou projetos. Ex.: subprojetos interdependentes, rotação de projetos entre grupos.	**Avaliações auxiliares:** motivação, engajamento, entre outros aspectos psicométricos. Lacuna cognitiva do coletivo discente. **Como:** Construtos e/ou projetos. Ex.: projeto da sala como grupo único, projeto do curso, projetos de turmas anteriores, contribuições de sujeitos externos à sala de aula.

Na tabela 1, para avaliações psicométricas, existem diversos instrumentos psicométricos, ou construtos. Assim, é sugerido classificar estes construtos com relação às necessidades da pesquisa em dois diferentes tipos: construtos específicos ou construtos abrangentes. Como exemplo de construto específico, há o de avaliação de motivação em programação, disponível na pesquisa de Santana, Figuerêdo e Bittercour (2017). Outro exemplo é o *Ticket Driven Development* (desenvolvimento orientado a bilhete, tradução nossa), que avalia a contribuição do indivíduo para com o grupo de estudo por meio de bilhetes que descrevem as atividades esperadas de cada aluno [Igaki et al. 2014]. Segundo, como exemplo de construto abrangente, há um instrumento para validação da motivação no ensino superior, que mensura a motivação extrínseca e intrínseca dos alunos [Boruchovitch 2008]. Existem, assim, diversos instrumentos destinados à coleta de avaliações auxiliares. Estes podem inclusive vir das áreas humanas, tais como a Psicologia e, ainda assim, servirem para avaliação da aprendizagem por meio de PjBL. Portanto, sempre que possível, é sugerida a opção por construtos reutilizáveis de outras pesquisas, de modo a simplificar, acelerar e padronizar futuras pesquisas, em detrimento da criação de novos construtos.

Relativo à lacuna cognitiva, em Biggs e Tang (2011) não há uma avaliação definida para como medi-la. Assim, propõe-se que a lacuna cognitiva no EA-PROG pode ser avaliada estatisticamente pela distância geométrica entre dois centros de gravidades: dos centros dos perfis F e T. A título de exemplo ilustrativo, dadas as notas dos alunos, e

assumindo que é possível classificar os alunos em perfis T e F, uma lacuna cognitiva de valor igual a zero significa que os rendimentos escolares foram similares (notas estatisticamente iguais). A medida que a distância geométrica das notas destes perfis T e F aumenta, a lacuna cognitiva assume valor diferente de zero. Este valor representaria a lacuna cognitiva ilustrada na Figura 1 e referenciada na Tabela 1.

Propõe-se também que os projetos, por si só, podem se configurar instrumentos de avaliações auxiliares, conforme Tabela 1. Quando os docentes de um estudo descrevem a motivação e alegria dos estudantes ao verem seu projeto PjBL realizado com sucesso [Kastl et al. 2016], há neste relato uma clara avaliação auxiliar, mesmo que qualitativa. Outro relato similar foi a inclusão da satisfação de um cliente externo à aula como parte da composição da nota final. Outro exemplo é o envolvimento dos demais sujeitos do processo educacional, tais como alunos, grupos e pessoas externas às aulas, todos contribuindo para a composição das notas. Ou relatos da autoavaliação do aluno visando compor avaliações auxiliares ou sua própria nota [Abdool and Pooransingh 2014; Francese et al. 2015]. Todas estas avaliações auxiliares podem ter diversos ângulos, podem ser agregadas ou não às notas. Em PjBL, os projetos são a espinha dorsal pelo qual o EA-PROG acontece. No entanto, sempre se deve observar a aprendizagem como objetivo final da experiência educacional.

No que se refere à medição do ensino, existem diversas dificuldades e aspectos a serem considerados nas aferições de intervenções metodológicas de ensino [Bordenave and Pereira 1991]. Uma alta complexidade nas inovações ou nas avaliações educacionais pode desestimular ou até mesmo inviabilizar pesquisas na educação [Bell 2016]. Assim, é proposta uma avaliação simplificada nomeada *experiência educacional*. Esta é uma resultante em escala Likert do quão construtivista ou quão instrutivista foi a experiência dos alunos com base no tempo de aula. Mais especificamente, a medição da experiência educacional ser aferida por uma escala de cinco pontos, dentro do *continuum* (ou gradiente) educacional. Este gradiente é classificado em escala ordinal por meio de cinco letras, nesta sequência: I, i, ic, c e C. A letra (I), significa totalmente instrutivista, equivale a menos dois pontos e representa o polo instrutivista. Menos um ou sigla (i), representa um método predominantemente instrutivista. A sigla (ic) representa uma experiência educacional próxima ao centro do gradiente e, portanto, igualmente instrutivista e construtivista. E assim por diante, até chegar ao outro lado do gradiente, com a letra (C), que significa uma experiência próxima ao polo construtivista. Valores intermediários devem ser arredondados por meio da regra matemática padrão.

Também se propõe uma avaliação ordinal a ser agregada à experiência educacional: recursos educacionais PjBL para o EA-PROG. Estes recursos podem ser de dois tipos: físicos ou digitais. Visam mensurar o quão preparados estão os ambientes físico e digital à implantação de PjBL. Exemplos destes recursos podem ser: presença de salas de aula não lineares ou salas de reunião coletiva; acesso a espaços exteriores à sala de aula, tais como a comunidade ou empresas; ferramentas digitais de gestão de projetos.

A tabela 2 sintetiza a proposta do que avaliar, e de como avaliar, relativos aos dois aspectos ensino e aprendizagem de programação de computadores. Primeiro, relativo ao ensino é proposto avaliar o grau de experiência educacional e respectivos recursos educacionais. Segundo, relativo à aprendizagem, é proposto a utilização de avaliações auxiliares, em complemento as avaliações tradicionais requeridas por Lei:

Tabela 2. Resumo das variáveis para medição do EA-PROG por meio de PjBL

EA-PROG	Nome	Tipo	Níveis e intervalos	Como (método)
Ensino	Recursos Educacionais para PjBL	Ordinal	(0) ausente; (1) parcial; (2) presente.	Aferição da presença de recursos para EA-PROG com PjBL nos ambientes físicos e digitais.
Ensino	Experiência Educacional em PjBL	Ordinal	(I) muito instrutivista; (i) moderadamente instrutivista; (ic) igualmente instrutivista e construtivista; (c) moderadamente construtivista; (C) muito construtivista.	Escala Likert de cinco pontos, variando de menos dois a mais dois. Avalia os métodos educacionais com base no tempo de aula. Por exemplo, dadas quatro aulas possíveis, se três foram atividades coletivas por meio de PjBL e se uma foi expositiva então a resultante é uma experiência educacional "c" (moderadamente construtivista), ou seja, 75% do tempo de atividades construtivistas.
Aprendizagem	Medida Auxiliar: Psicométrica	Racional	0 a 100	Utilização de construtos psicométricos, preferencialmente validados no contexto nacional, tais como Motivação [Boruchovitch 2008] ou engajamento [Stelko-Pereira et al. 2015].
Aprendizagem	Medida Auxiliar: Desempenho do Coletivo	Racional	0 a 100	Lacuna Cognitiva por meio da distância geométrica entre grupos ou indivíduos. Quanto menor o valor, menor a lacuna cognitiva.

5. Limitações e Discussões

Existem limitações ao escopo deste ensaio. Uma relevante limitação está relacionado ao fato de que o método de ensino e os métodos de avaliação da aprendizagem não são os únicos fatores de impacto no ensino de graduação [Bordenave and Pereira 1991; Queirós 2014]. Adicionalmente, as causas sociais não atuam de modo isolado, como é o caso da educação. Além disso, há de se considerar limitações e cuidados ao se estabelecer casualidade e implicações na área educacional [Bell 2016]. Estes fatos impõem limites à amplitude e à profundidade das proposições, configurando tópicos para estudos futuros.

Há também diversas discussões a respeito do tema. Estas podem ser tomadas desde a perspectiva do indivíduo até à perspectiva da soberania nacional. Relativo aos estudantes, depois de vários anos utilizando a abordagem tradicional de EA-PROG, chegam ao mercado com uma alta bagagem teórica, mas lacunas em áreas-chave do EA-PROG. Estão familiarizados com conceitos abstratos e de difícil compreensão, porém não são raras as queixas de empresas a respeito do pouco conhecimento prático dos alunos [Zhang and Liu 2012]. São relatadas dificuldades não técnicas destes alunos, tais como a baixa capacidade de comunicação, de trabalho em equipe ou de resolução de problemas [Morimoto 2016]. Talvez por isso avaliações auxiliares tentem entender este fenômeno em sala de aula, sendo um relevante tópico para estudos futuros.

Se estas lacunas dificultam o acesso dos alunos ao mercado de trabalho no contexto em que foram estudados [Morimoto 2016; Zhang and Liu 2012], de modo análogo, no contexto da soberania nacional, há relatos da ligação entre o

subdesenvolvimento econômico versus a capacidade de inovação educacional da Nação. Novos métodos de ensino e aprendizagem são fatores-chave para o progresso de uma nação [Porcaro 2011]. Assim, podem ser tópicos de interesse a utilização de PjBL no EA-PROG como meio para: a redução de lacunas de empregabilidade; a orientação e o fomento de programas educacionais para necessidades regionais; uma maior integração com mercados locais ou internacionais de trabalho.

Por fim, são igualmente relevantes as limitações para as avaliações utilizadas no contexto PjBL. No ensino tradicional, usualmente avalia-se o resultado das ações de um aluno [Luckesi 2008] tais como resultados de provas, trabalhos ou projetos. Pesquisas futuras poderiam investigar os métodos, pesos, ponderações e instrumentos mais adequados ao EA-PROG por meio de PjBL. As avaliações também poderiam ser adaptadas ou ponderadas de acordo com o grau da experiência educacional. Ou ainda, baseado em diferentes perfis de alunos, investigar quais avaliações e adaptações podem ocorrer para o acolhimento das diferenças como fator de sinergia educacional, trazendo à rotina acadêmica a pluralidade usualmente encontrada em projetos do mundo real.

6. Conclusão

São vários os relatos e pesquisas a respeito do tema de inovação educacional por meio de PjBL para a programação de computadores, com relevante dedicação ao fenômeno de benefícios aos alunos. Estes benefícios estão relacionados às avaliações tradicionais, na forma de notas ou presença dos alunos. Ou relacionados às avaliações auxiliares: i) melhorar aspectos psicométricos – a motivação, a capacidade de trabalho em equipe, a comunicação, entre outros; ou ii) reduzir a lacuna cognitiva dos alunos, a exemplo do fenômeno de redução da variância no rendimento escolar.

Neste desafiante contexto, este ensaio analisou a padronização sobre o que avaliar, e como avaliar, no EA-PROG por meio de PjBL. Com relação ao ensino, foi proposto avaliar os métodos de ensino e seus recursos educacionais. Com relação à aprendizagem, foram propostas avaliações auxiliares. Estas podem ser agrupadas em: avaliações psicométricas, como a motivação; ou avaliações do desempenho coletivo, a exemplo da lacuna cognitiva. Podem também ser classificadas em três tipos: individuais, dos grupos e do coletivo. Foram abordados exemplos de avaliações auxiliares, sem, no entanto, exaurir a discussão. Adicionalmente, foram feitas propostas para futuras pesquisas.

Uma eventual padronização destas avaliações poderia colaborar com a simplificação da coleta, da análise e do compartilhamento de dados nesta linha de pesquisa. A padronização também poderia contribuir para criação de ferramentas digitais reaproveitáveis, ou melhorar a integração da programação de computadores com outras áreas interdisciplinares. Em suma, esta padronização poderia colaborar com a identificação de benefícios adicionais aos alunos, ajudar no combate à repetência e à evasão, contribuindo para uma melhor formação de futuros programadores.

Referências Bibliográficas

Abdool, A. and Pooransingh, A. (2014). An Industry-Mentored Undergraduate Software Engineering Project. In *2014 IEEE Frontiers in Education Conference (FIE)*. , Frontiers in Education Conference. IEEE.

Beck, C. and Kosnik, C. (2012). Innovations in Teacher Education: A Social Constructivist Approach. SUNY series, Teacher Preparation and Development. New

York: State University of New York Press. p. 7–23.

Bell, J. (2016). Projeto de Pesquisa: guia para pesquisadores iniciantes em educação, saúde e ciências sociais. 4. ed. São Paulo: Artmed Editora. p. 21–22.

Biggs, J. B. and Tang, C. S. (2011). Teaching for quality learning at university. *SRHE and Open University Press imprint*. SRHE and Open University Press Imprint. 4. ed. New York, USA: McGraw-Hill Education. p. 1–16.

Bordenave, J. D. and Pereira, A. M. (1991). *Estratégias de ensino-aprendizagem*. 12. ed. Petrópolis: Vozes.

Boruchovitch, E. (2008). Escala de Motivação Para Aprender de Universitários: Propriedades Psicométricas. *Avaliação psicológica*, v. 7, n. 2, p. 127–134.

BRASIL (1996). Lei nº 9.394, de 20 de dezembro de 1996. *Brasil*. http://www.planalto.gov.br/ccivil_03/leis/L9394.htm.

Chan Mow, I. T. (2008). Issues and difficulties in teaching novice computer programming. In: Iskander, M.[Ed.]. . *Innovative Techniques in Instruction Technology, E-Learning, E-Assessment, and Education*. Dordrecht: Springer Netherlands. p. 199–204.

Corno, F. and De Russis, L. (2017). Training Engineers for the Ambient Intelligence Challenge. *IEEE Transactions on Education*, v. 60, n. 1, SI, p. 40–49.

DeFillippi, R. J. (2001). Introduction: Project-Based Learning, Reflective Practices and Learning. *Management Learning*, v. 32, n. 1, p. 5–10.

Fior, C. A. and Mercuri, E. (2013). Evidências de validade da Escala de Envolvimento Acadêmico para universitários. *Avaliação Psicológica*, v. 12, n. 1, p. 81–89.

Francese, R., Gravino, C., Risi, M., Scanniello, G. and Tortora, G. (2015). Using Project-Based-Learning in a mobile application development course—An experience report. *Journal of Visual Languages & Computing*, v. 31, p. 196–205.

Giraffa, L. M. M., Moraes, M. C. and Uden, L. (2014). Teaching Object-Oriented Programming in First-Year Undergraduate Courses Supported By Virtual Classrooms. In: Uden, L.; Tao, Y.-H.; Yang, H.-C.; Ting, I.-H.[Eds.]. . *The 2nd International Workshop on Learning Technology for Education in Cloud*. Dordrecht: Springer Netherlands. p. 15–26.

Goulding, T. (2013). A first semester freshman project: The enigma encryption system in C. *ACM Inroads*, v. 4, n. 1, p. 43–46.

Greening, T. (2012). Computer Science Education in the 21st Century. In: Springer Science & Business Media[Ed.]. . Ballarat: Springer New York. p. 47–59.

Guzman-Ramirez, E. and Garcia, I. A. (2013). Using the Project-Based Learning Approach for Incorporating an FPGA-Based Integrated Hardware/Software Tool for Implementing and Evaluating Image Processing Algorithms Into Graduate Level Courses. *Computer Applications in Engineering Education*, v. 21, n. 1, p. E73–E88.

Igaki, H., Fukuyasu, N., Saiki, S., Matsumoto, S. and Kusumoto, S. (2014). Quantitative Assessment with Using Ticket Driven Development for Teaching Scrum Framework. [A. Jalote, P and Briand, L and VanDerHoek, Ed.]In *36th International Conference on Software Engineering (ICSE Companion 2014)*. . Association for Computing

Machinery.

Iskander, M. (2008). Innovative Techniques in Instruction Technology, E-learning, E-assessment and Education. SpringerLink: Springer e-Books. New York: Springer Netherlands. p. 199–203.

Kastl, P., Kiesmüller, U. and Romeike, R. (2016). Starting out with projects - Experiences with agile software development in high schools. [B. E. Vahrenhold J. Barendsen E., Ed.]In *ACM International Conference Proceeding Series*. . Association for Computing Machinery.

Luckesi, C. C. (2008). *Avaliação da Aprendizagem Escolar: estudos e proposições*. Cortez Editora.

Manogaran, E. (2013). ACt-PBL: An adaptive approach to teach multi-core computing in university education. In *Proceedings - 2013 IEEE 5th International Conference on Technology for Education, T4E 2013*. . IEEE Computer Society.

Mioduser, D. and Betzer, N. (2007). The contribution of Project-based-learning to high-achievers' acquisition of technological knowledge and skills. *Int J Technol Des Educ*, v. 77, n. 27.

Morimoto, C. (2016). Improvement of IT Students' Communication Skills using Project Based Learning. In *Proceedings of the 8th International Conference on Computer Supported Education*. . SciTePress.

Payne, C. R. (2009). Information Technology and Constructivism in Higher Education: Progressive Learning Frameworks. Hershey, PA, USA: IGI Global. p. 1–25.

Porcaro, D. (2011). Applying constructivism in instructivist learning cultures. *Multicultural Education & Technology Journal*, v. 5, n. 1, p. 39–54.

Queirós, R. (2014). Innovative Teaching Strategies and New Learning Paradigms in Computer Programming. Advances in higher education and professional development (AHEPD) book series. Hershey, PA, USA: IGI Global. p. 131–133.

Romeike, R. and Göttel, T. (2012). Agile projects in high school computing education - Emphasizing a learners' perspective. In *ACM International Conference Proceeding Series*.

Santana, Bianca L; Figuerêdo, José S; Bittencour, R. A. (2017). Motivação de Estudantes Non-Majors em uma Disciplina de Programação. *25º WEI - Workshop sobre Educação em Computação*, p. 2287–2296.

Silva Filho, R. L. L. e, Motejunas, P. R., Hipólito, O. and Lobo, M. B. D. C. M. (2007). A evasão no ensino superior brasileiro. *Cadernos de Pesquisa*, v. 37, n. 132, p. 641–659.

Stelko-Pereira, A. C., Valle, J. E. and Williams, L. C. A. (2015). Escala de engajamento escolar: análise de características psicométricas. *Avaliação Psicológica*, v. 14, n. 2, p. 207–212.

Zhang, Y. and Liu, Y. (2012). Management enhanced double PBL based reform in advanced programming design course. In *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communications, HPCC-2012 - 9th IEEE International Conference on Embedded Software and Systems, ICESS-2012*.

An Interdisciplinary Approach to Software Engineering Teaching: An Experience Report

Gláucia Braga e Silva[1], **Daniel Mendes Barbosa**[1], **Fabrício A. Silva**[1]

[1]Institute of Exact and Technological Sciences - Federal University of Viçosa (UFV)
Florestal – MG – Brazil

{glaucia, danielmendes, fabricio.asilva}@ufv.br

Abstract. *This work presents the report of an interdisciplinary approach for teaching Software Engineering that involves four related disciplines: Software Engineering II, Software Architecture, Database Systems, and Object-Oriented Programming. The approach was conducted through an specific methodology in which students were grouped together and had to assume roles and responsibilities in the scope of each discipline. In addition, computational tools were used to support collaborative tasks and evaluation and monitoring mechanisms were also included. This interdisciplinary approach was adopted in a Computer Science major course during 2015 and 2016. The results reveal positive impacts in motivation, as well as in learning aspects of the involved students.*

1. Introduction

Due to the increasing demand for software applications in industry, there are many challenges of teaching software engineering [Zeidmane and Cernajeva 2011]. The industry requires more and more professionals with multi-functional skills and capable of working in multidisciplinary environments. Although over the last 45 years the practice in Software Engineering has made significant progress, there are significant gaps between the teaching and the needs of the software industry [Bass 2016, Moreno et al. 2012]. The teaching of Software Engineering is a hard task because there are a lot of competences and abilities to be taught in order to prepare a professional which combines technical knowledge, experience, and ability to interact with clients [Jazayeri 2004, Teel et al. 2012]. According to Nurkkala and Brandle [Nurkkala and Brandle 2011], traditional approaches of Software Engineering teaching present the following problems: no real products; short duration that causes an artificial time constraint and requires projects with low complexity; high turnover of the students; low projects complexity; no software maintenance; and lack of interaction with real customer. Furthermore, because of the large amount of theoretical concepts, Software Engineering is often considered by students as a boring subject [Teel et al. 2012].

In this context, educators must be continuously engaged in the creation of new teaching strategies that include practices compatible with the current software industry trends and that motivate students to appreciate the importance of this discipline for their careers. Some of these issues can be addressed by using interdisciplinarity to teach Software Engineering since it is a key topic in computing education [Teel et al. 2012] and it must be taught in an integrated way. Interdisciplinary teaching involves the interactions between two or more academic disciplines with a common goal. In this context, educators can bring to the classroom an environment that engages students and helps them to

develop knowledge, insights, problem solving skills and self-confidence. The use of interdisciplinary teaching contributes to a more complete and integrated academic formation of the students who will be better prepared and qualified for the industry.

This work presents an experience report of an interdisciplinary approach for teaching Software Engineering in a Computer Science major course of our University, during 2015 and 2016. The approach was conducted through an specific methodology designed to develop a software product around the teaching of four related disciplines: Software Engineering II, Software Architecture, Database Systems, and Object-Oriented Programming. The approach was proposed with the objective of bringing industry-related scenarios to the academy, allowing students to experience team work, share their knowledge and learn by experimentation [Ghezzi and Mandrioli 2005]. The approach adopts a problem-based learning process in which students can construct and acquire knowledge, enhance group collaboration and communication while develop a software project. In addition, technical capabilities, such as project management, requirements tracking, configuration management, software quality and test and collaboration tools can be experienced in a hands-on manner [Teel et al. 2012]. Our approach has been used to engage and empower students' learning in an undergraduate computer science course. In this paper, we summarize our experiences and lessons learned. The results reveal positive impacts in motivational as well as in learning aspects of the involved students.

The paper is organized as follows: section 2 discusses some related works. Section 3 describes the proposed approach, details its methodology, and presents the results of two consecutive years of its adoption. Finally, in section 4 we conclude the report.

2. Related Works

This section presents studies that address new strategies to teach Software Engineering which explore interdisciplinarity, experiential learning and non-technical skills. Marsicano et al. [Marsicano et al. 2016] present a teaching method that integrates the disciplines Requirements Engineering and Process modeling in an undergraduate course. The method was analyzed in two ways: grades and feedback on technical report analysis. Schaetter et al. [Schaetter et al. 2009] describe a multidisciplinary approach to teach Software Engineering based on teaching and learning methods. Through interdisciplinary projects, students are trained in software projects under realistic conditions. The results pointed that this approach has had significantly enhanced the students employability, according to feedback from their industrial partners. Chen and Chong [Chen and Chong 2011] present an study which introduces a meetings-flow approach to help in the instruction of student teamwork and to formalize stakeholder participation. The authors conducted a quantitative investigation which assessed the project and examined the numerical benefits that the approach brought to the project development. In addition, the study conducted group interviews to discuss the qualitative and educational effects of the approach. Bareiss and Griss [Bareiss and Griss 2008] discuss the use of teaching methods, coaching, and feedback in the Carnegie Mellon University and reports the positive impacts in the students formation such as competitive advantage and salary increases. The methods are based on student-centered learning where students are encouraged to discover knowledge themselves and to learn by doing and they are evaluated based on what they produce. Letouze et al [Letouze et al. 2016] propose a Problem-Based Learning approach to develop a web system for managing academic projects. In their ap-

proach, role-play scenarios were used in order to prepare students and to improve abilities within a role as development team member. Our approach is also based on interdisciplinarity since it involves the integrated teaching of Software Engineering in four disciplines in a Computer Science major course. However, we also propose a methodology to structure the approach in stages, allowing its replication in other academic institutions that offer, in the same academic semester, disciplines in the areas of Software Engineering, Database Systems and Programming. The proposed approach is not restricted to the application of Software Engineering concepts, but it aims to explore the interactions between the various roles of a software process, the collaborative production of artifacts and the use of computational tools, simulating situations commonly found in the real scenarios of the job market. Furthermore, the new realities of teaching Software Engineering [Jazayeri 2004] are covered by the proposed approach, as it tackles current issues such as software evolution, software quality, tools and environments. In addition, the approach provides a proper environment to develop non-technical skills such as communication and ability to work as a team.

3. Structuring of the Interdisciplinary Approach and Experience Report

This section presents an experience report of an interdisciplinary approach for Software Engineering teaching in the Computer Science undergraduate course at Federal University of Viçosa (UFV) - campus Florestal. The approach involved the integrated teaching of four related disciplines present in the course curriculum. All disciplines are offered each even semester, as shown in Table 1, and they are strongly related since their contents are complementary. It is important to note that Software Engineering I is a previous discipline, offered on the fifth semester of the course, and is responsible for the fundamentals of the area. However this discipline is not considered in our approach.

Table 1. Involved Disciplines

Discipline	Initials	Semester	Scope
Software Engineering II	SE	sixth	Process Management and Test
Object-Oriented Programming	OOP	fourth	Coding according to the O.O. paradigm
Database Systems	DB	sixth	Modeling, Design and Queries
Software Architecture	SA	eighth	Specification, Design and Code Integration

The scope of each discipline within the approach was defined based on their technical scope, as recommended by the course analytical program, but explored under an interdisciplinary perspective. Table 2 illustrates the interdisciplinary relations involved.

Table 3 illustrates the relation of students for each discipline (SE, OOP, DB and SA) in both editions (2015-2 and 2016-2) as well as the intersections of students enrolled in more than one discipline. In both editions, there were students who attended only one of these four disciplines, as well as others who attended two, three or four of them simultaneously. In 2015-2, 38 students and 2 instructors (1 professor was responsible for 3 disciplines) participated of the approach. In 2016-2, this number increased to 56 students and 3 instructors (1 professor was responsible for 2 disciplines).

To guide the interdisciplinary approach, we propose a methodology comprised by three stages: planning, execution and control, and closing (Figure 1). The planning

Table 2. Interdisciplinary Relations

From	To	Activities and Artifacts
SE	OOP	schedule, configuration management and software patterns
	DB	schedule, configuration management, quality (DB models and SQL scripts)
	SA	schedule, configuration management, GUI design, quality, tests (UML models and DB models)
SA	OOP	model specifications (UML use cases and class diagrams)
	DB	model specifications (UML class diagrams)
	SE	model specifications (UML use cases and class diagrams) and integrated code (Model-View-Controller architecture)
OOP	DB	specification of database queries
	SA	code of Model and Controller layers (to be integrated)
DB	SE	DB models, DB SQL scripts (creation and queries) to be tested
	SA	DB models and DB SQL queries (to be integrated).

Table 3. Students per discipline

Year	OOP	DB	SE	SA	$OOP \cap DB$	$OOP \cap SE$	$DB \cap SE$	$SA \cap DB$	$OOP \cap DB \cap SE$	Total
2015-2	20	10	10	5	2	1	5	-	1	38
2016-2	29	25	20	8	10	4	13	1	2	56

stage is responsible for the following definitions: case study, roles and responsibilities, and computational tools. This stage is also responsible for the development of the project schedule. In the execution and control stage, the schedule is controlled through the evaluation of the work products and the continuous feedback from the participants. At the end, in the closing stage, the approach is evaluated and the lessons learned are registered.

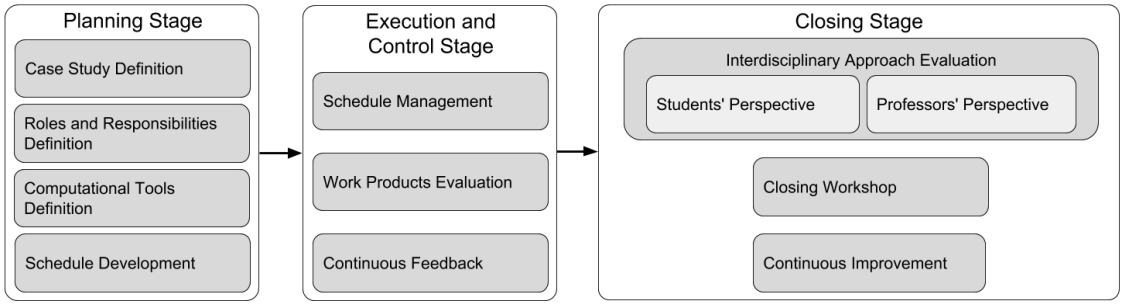


Figure 1. Methodology of the Interdisciplinary Approach

The following sections detail the proposed methodology stages and describe the results of both editions.

3.1. Planning Stage

The four management activities of this stage are conducted by the professors with the support of some students of SE and SA disciplines. The first activity of the planning stage is the *case study definition*. In order to cover the curricular content of the involved disciplines, this case study should address the development of a software product using the

object orientation paradigm, data persistence in relational databases, design patterns and frameworks. An academic system to control attendance at the university and a prototype web-based system to control denunciations of focus of *Aedes Aegypti* were the choices in the 2015-2 and 2016-2 editions, respectively.

The *roles and responsibilities definition* was based on the technical nature of each discipline, the activities planned in each one and the knowledge level of the students involved, with students from the more advanced periods (SA and SE) taking on more heterogeneous roles and of greater responsibility in the project. Because of that and considering that OOP and DB disciplines are offered in the 4th and 6th periods respectively, the activities of these two disciplines were uniform and so all the students of each one were divided into teams and each team performed the same tasks. Table 4 presents the defined roles and the corresponding responsibilities and work products for each discipline.

Table 4. Roles and Responsibilities

Role	Responsibility	Work Products	Discipline
Project Manager	Schedule and Workflow Control	Project Schedule	SE
Configuration Manager	Change and Version Control	SCM Plan	SE
GUI Designer	Design of Graphical User Interfaces	GUI prototypes	SE
Quality Analyst	Quality Assurance (models, codes, ...)	Quality Assurance Plan	SE
Tester	Software Tests	Test Plan, Test Cases, Test Reports	SE
Discipline Leader	Support to students and professors	Communication Plan, Meetings Reports	SE
Software Architect	Software specification and Design	Use Case, Classes and Components	SA
	Revision of database models	Improved Models	SA
Senior programmer	Revision of codes	Improved Codes	SA
	Coding of View layer	Boundary Classes	SA
	Integration code between layers (MVC)	Integrated code	SA
O.O. Programmer	Coding of Model layer	Entity Classes	OOP
	Coding of Controller Layer	Controller Classes	OOP
	Specification of database queries	Queries demands report	OOP
DB Designer	DB Logical Modeling	DB Logical Model	DB
	DB Physical Modeling	DB Physical Model	DB
DB Analyst	Database creation and data insertion	SQL Scripts	DB
	Development of SQL Queries	SQL Queries	DB

As shown in Table 4, in order to correct some workflow and communication problems, four SE students took on the role of leaders from each of the four disciplines to improve the teamwork. The discipline leaders were responsible for supporting the students in the execution of their tasks and also the professors in the monitoring of performed tasks and produced results. Compared with the 2015-2 edition, we noted that the performance of the discipline leaders in 2016-2 edition was central to improve teamwork and especially the relationships between disciplines. This is a very important aspect because students working in teams to complete software tasks is an effective method to learn necessary teamwork skills required in software industry[Shuto et al. 2016].

As the case study involves a software product development, the *Planning Stage* includes the *definition of computational tools* to support both process and collaborative work. Several tools were used during the project execution (Figure 2) and helped the work conduction by the professors in management and didactic support contexts. In addition, other tools supported the students in carrying out their tasks in the software process scope.

In the didactic support context, theoretical contents and task specifications of each

Context	Computational Support	Tools	
		2015-2	2016-2
Didactic Support	Virtual Learning Environment	System 1 (@OurUniversity)	System 1 (@OurUniversity)
	Control of Attendance and Grades	System 2 (@OurUniversity)	System 2 (@OurUniversity)
	Feedback	Google Forms, Padlet	Google Forms
Project Management	Schedule Management	Gantt Project	GanttPro
	Workflow Management	-	Trello
	Communication Management	Gmail, Hotmail, GoogleGroups	Gmail, Hotmail
Software Development	UML Modeling	Astah Community	Astah Community
	Database Modeling	MySQL Workbench	MySQL Workbench
	Design of Graphical User Interfaces	Moqups	Moqups
	Database Management	MySQL Server	MySQL Server
	Code development (IDE)	NetBeans	NetBeans
	Frameworks	JSF, PrimeFaces	JSF, Hibernate, SpringSecurity, ...
	Web Server or Application Server	Apache Tomcat	Apache Tomcat
	Unit Tests	-	JUnit
	Funcional Tests	Selenium	Selenium, Sikuli
	Security Tests	-	ZAP, IronWASP, SqlMap
	Configuration Management (Issue Tracking)	Mantis Bug Tracker	Mantis Bug Tracker
	Configuration Management (Version Control)	Subversion	Git

Figure 2. Support of Computational Tools

of the disciplines were available to students through a virtual learning environment. The students could view their grades in each of the participating disciplines through the official grading system. In the context of software development, tools were used for UML modeling, data modeling and system interfaces prototyping. The resultant artifacts of that were then used as inputs for coding and testing. All produced artifacts were submitted to configuration control, using issue tracking tools and version control. Finally, in the project management context, there was a small variation of the tools used. In the 2016-2 edition, the *Trello* tool was adopted, with the purpose of supporting the discipline leaders and professors in the assignment and monitoring of tasks in each discipline. The tool was also used by the discipline leaders to improve communication with the students/teams in each discipline. It should be noted that most of the adopted tools are widely used in software development projects, either in the academy or in the software industry.

Schedule Development is the last activity of this first stage, in which we developed a schedule, defining project activities, their deadlines and interdependence relationships. It was essential to workflow management in the *Execution and Control stage*.

3.2. Execution and Control Stage

At this stage, according to the developed schedule, the deadlines and results of the planned tasks were monitored and evaluated by the professors, supported by the students with the roles of project managers and discipline leaders. The planned tasks in the project schedule were executed by the students and monitored by the professors, with the support of project managers and discipline leaders. Professors, project managers and discipline leaders (2016-2) continuously interacted throughout the semester, through meetings and exchange of emails. Throughout the project some adjustments were made to the schedule, with readjustments in dates and time lengths of tasks that had not been correctly planned initially. In the 2016-2 edition, each discipline leader created a board in the *Trello* tool with the tasks of the teams in their respective discipline, which was continuously monitored, always informing project managers of the status of the tasks performed.

From the technical point-of-view, there were many interactions between the different roles during the academic semester. In both editions, the students of the SA presented the system requirements and models to the other students, in reserved classes of DB, OOP and SE. The students attended design technical meetings, some during the classes of the involved disciplines and others outside the classroom, in person or in a virtual way. Students from the same team (DB and OOP) or role (SA and SE) were also in constant interaction, exchanging information in e-mail systems and discussion groups.

In the 2015-2 edition, professors were responsible for direct interactions with students in their respective disciplines. This caused some schedule delays, since professors have several other assignments, and in some cases such interactions with students were only possible in a class in the following week. But in the 2016-2 edition, with the participation of the discipline leaders, we observed that the information arrived more efficiently to the students of each discipline.

To control the execution of the project tasks, in each edition was defined specific *work products evaluation* for each discipline. The evaluation was carried out by the responsible professors, based on the students pre-evaluation with roles of discipline leaders (SE), senior programmers (SA) and testers (SE). In cases where teams did the same task (OOP and DB), the professor selected the best result to follow to the next tasks. At the moment of the selection, the professor then assigned the grades to each team, using an ”award” strategy for the best results presented (class models, data models, source code). Professors then discussed these results with the students, in order to clarify all doubts of that task. In addition to this evaluation within the project itself, the content learned by the students could also be evaluated in other activities within each discipline, including written tests. The data stored in the repositories of the configuration control tools were also analyzed by the professors (2015-2 and 2016-2) and discipline leaders (2016-2), in order to verify the fulfillment of the deadlines of the schedule, the quality of the productions and the level of involvement of team members.

Professors, project managers and discipline leaders received *continuous feedback* from other participants throughout the semester, whether in the classroom, by e-mail or through computational support tools. In 2016-2, students reported their doubts and suggestions to the discipline leaders and so the students with this role reported to the professors who applied some adjustments throughout the semester.

3.3. Closing Stage

To evaluate the interdisciplinary approach itself, we have collected additional information at the ending of each edition. In the following we discuss the evaluation of the approach under the students' and professors' perspectives.

To evaluate the approach from a *students' perspectives*, every participating student answered a survey at the end of each edition. Each question covers a topic and contains five descriptive alternatives representing a level each: very high, high, medium, low, very low. The main results reveal that 71.4% of the students classify as *high* or *very high* their motivation to conclude the disciplines because of the interdisciplinary approach in 2015-2. We also observed an increasing of this measure from 71.4% to 76.8% in 2016-2. The survey also reveals that 89.3% and 79.1% of the students classified as *very high* or *high* the impact of the project in the learning during the editions of 2015-2 and 2016-2,

respectively, which means they considered the project as a motivator for their studies. On the other hand, some students have pointed out failures on the communication process in both editions. In 2016-2, this problem could be bigger due to increased number of participants, but we adopted *Trello* to organize the tasks and created the role of discipline leader, which was considered important to 67.4% of the students.

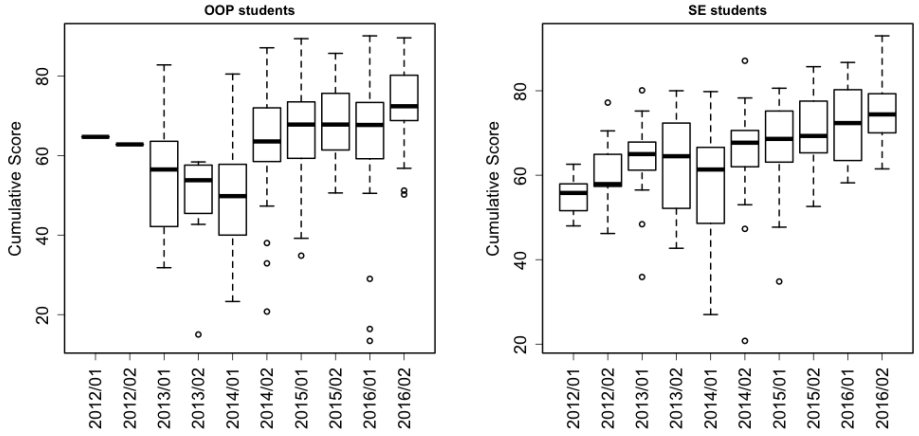
To assess how the interdisciplinary project affects the students' academic performance, from a *professors' perspective*, we analyze their grades during and after the period of the project. To this end, we consider as metric the cumulative score that is the grades' weighted average in which the weights are relative to each discipline's credits. This metric indicates well the students' performance and is officially adopted at the university where the project was conducted. Figure 3 depicts the cumulative scores for the students that were enrolled at Object-Oriented Programming discipline (Figure 3(a)) and Software Engineering II (Figure 3(b)). For both cases, it is clear the high trend in the scores, especially from 2015-2 term on, when the project was firstly adopted.

To reinforce this observation, we compute the Pearson's correlation coefficient of each student. The Pearson's coefficient varies from -1 to 1, indicating an increasing trend when positive. The assessed values were higher than 0.18 for more than half of the students that attend the Object-Oriented Programming discipline. The trend is even higher for the students that attend the Software Engineering II discipline, for whom the Pearson's coefficient was higher than 0.18 for over 75% of the students, and higher than 0.82 for more than half of them. This positive correlation reinforces that the interdisciplinary project was beneficial to improve the students' grades.

As described in Section 3.1, some students were enrolled in more than one discipline simultaneously. To assess whether this is beneficial or not, we evaluate the scores of these students separately. To avoid biased information, we ignore the student that was enrolled in the four disciplines. Figure 4 depicts the cumulative scores of the students that were enrolled in two (Figure 4(a)) or in three (Figure 4(b)) disciplines simultaneously. As can be noted, the students enrolled in three disciplines present an increasing trending in their scores, with a Pearson coefficient of more than 0.34 for over half of the students. The same observation is valid for the students that were enrolled in two disciplines simultaneously, which present a Pearson coefficient of more than 0.27 for over half of them.

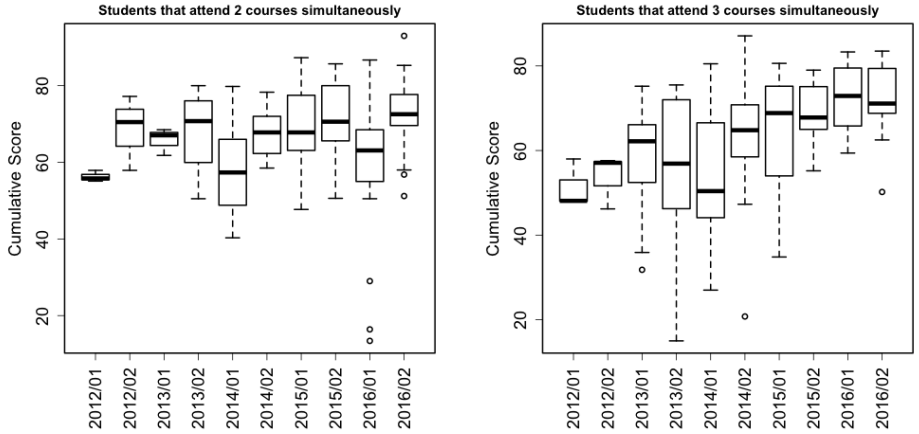
In summary, based on the cumulative scores and on the survey answers, it is possible to observe that the interdisciplinary project motivates the students to dedicate more to the studies, leading to better grades. At the end of the term, the students organized a *closing workshop* to discuss how the activities were conducted during the project. This way, the entire group get to know the overall project from different perspectives. The workshop is also used as an event where the students celebrate their accomplishments.

It is important to state that the methodology presented and adopted in this study should be adapted and improved constantly. To this end, all involved professors discuss the results and the students' feedback to detect potential improvements that would affect positively the upcoming editions. The changes adopted between the editions 2015-2 and 2016-2 in terms of roles and responsibilities (Table 4) and tools (Figure 2) are examples of results from the *continuous improvement*.



(a) OOP Students' Cumulative Scores (b) SE Students' Cumulative Scores

Figure 3. Students' cumulative scores increasing trend.



(a) Cumulative scores of students enrolled in 2 disciplines simultaneously (b) Cumulative scores of students enrolled in 3 disciplines simultaneously

Figure 4. Cumulative scores of students enrolled in 2 or 3 disciplines. It is possible to observe high trend in the grades of students enrolled in 3 disciplines.

4. Final Remarks

In this study, we present a case study of the application of an interdisciplinary project involving four software-engineering-related disciplines. The interdisciplinary approach was conducted in a way to replicate how software are developed in the industry. Thus, students could be familiar with processes, best practices and tools adopted in the industry. The survey results reveal that the project was considered as motivator to students, since they worked as a team and had to interact with other students of different disciplines. In addition, it was possible to observe a high trend in the students' grades that have attend the project, not only in the involved disciplines but in the overall computer science courses.

As future work, we plan to include other disciplines from different areas other than computer science, with the objective of integrating the computer science students with potential players in the software development process.

References

Bareiss, R. and Griss, M. L. (2008). A story-centered, learn-by-doing approach to software engineering education. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008*, pages 221–225.

Bass, M. (2016). Software engineering education in the new world: What needs to change? In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 213–221.

Chen, C.-Y. and Chong, P. P. (2011). Software engineering education: A study on conducting collaborative senior project development. *Journal of Systems and Software*, 84(3):479 – 491.

Ghezzi, C. and Mandrioli, D. (2005). The challenges of software engineering education. In *Proceedings. International Conference on Software Engineering - ICSE 2005 Education Track*, page 115 – 127. Springer.

Jazayeri, M. (2004). The education of a software engineer. In *Proceedings. 19th International Conference on Automated Software Engineering, 2004*, pages 18–27.

Letouze, P., d. Souza, J. I. M., and Silva, V. M. D. (2016). Generating software engineers by developing web systems: A project-based learning case study. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 194–203.

Marsicano, G., Mendes, F. F., Fernandes, M. V., and de Freitas, S. A. A. (2016). An integrated approach to the requirements engineering and process modelling teaching. In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 166–174.

Moreno, A. M., Sanchez-Segura, M.-I., Medina-Dominguez, F., and Carvajal, L. (2012). Balancing software engineering education and industrial needs. *Journal of Systems and Software*, 85(7):1607 – 1620. Software Ecosystems.

Nurkkala, T. and Brandle, S. (2011). Software studio: Teaching professional software engineering. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 153–158, New York, NY, USA. ACM.

Schaetter, A., Koeglmayr, H.-G., Blankenbach, K., and Nippa, M. (2009). Interdisciplinary approach to software engineering education. *Journal of Systematics, Cybernetics and Informatics)*, 7(5):29–36.

Shuto, M., Washizaki, H., Kakehi, K., Fukazawa, Y., Yamato, S., and Okubo, M. (2016). Learning effectiveness of team discussions in various software engineering education courses. In *Proceedings. 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 227–231.

Teel, S., Schweitzer, D., and Fulton, S. (2012). Teaching undergraduate software engineering using open source development tools. *Issues in Informing Science and Information Technology (IISIT)*, 9:63 – 73.

Zeidmane, A. and Cernajeva, S. (2011). Interdisciplinary approach in engineering education. In *Proceedings. IEEE Global Engineering Education Conference (EDUCON), 2011*, pages 1096–1101.

TuPy Online - Programação em Português com Visualização de Execução e Abstrações de Estruturas de Dados na Web

Giancarlo F. Roberto[1], **Fabiano S. Oliveira**[1]*,
Paulo Eustáquio D. Pinto[1]*, **Igor M. Coelho**[1]

[1]Instituto de Matemática e Estatística
Universidade do Estado do Rio de Janeiro (UERJ)
Rio de Janeiro – RJ – Brazil

giandroberto@gmail.com,

{fabiano.oliveira, pauloedp, igor.machado}@ime.uerj.br

Abstract. *There have been efforts to create tools that support the teaching of programming logic, such as program visualizers. One of them is Online Python Tutor which, despite having features for the representation of the state of variables for each step of the execution, does not allow convenient visualization of abstractions of more complex data structures, like graphs or trees. To address this problem, we present the TuPy Online tool, an adaptation of Online Python Tutor, proposing a lean syntax pseudolanguage (TuPy), commands in Portuguese and customizable display of data structures, being usable not only for program debugging but also for the preparation of courseware.*

Resumo. *Existem esforços para a criação de ferramentas de apoio ao ensino de lógica de programação, entre elas os visualizadores de programas. Um deles é o Online Python Tutor que, embora possua funcionalidades de representação do estado de variáveis a cada passo de execução, não permite a visualização conveniente de abstrações de estruturas de dados mais complexas, como grafos ou árvores. Para endereçar esse problema, apresentamos a ferramenta TuPy Online, uma adaptação do Online Python Tutor, propondo uma pseudolinguagem (TuPy) de sintaxe enxuta, comandos em português e exibição customizável para estruturas de dados, podendo ser utilizada tanto para depuração de programas quanto para preparação de material didático.*

1. Introdução

O estudo e o desenvolvimento de técnicas e ferramentas de suporte ao ensino da programação, sobretudo destinadas a estudantes da graduação, teve seu interesse aumentado durante a última década no âmbito da produção científica [Cunha et al. 2017]. Acredita-se que esse aprendizado é um esforço que deve ser capaz de se renovar em curtos intervalos de tempo para acompanhar o rápido ciclo evolutivo da tecnologia que ele tange. Esse processo de renovação pode ser ilustrado, por exemplo, pela adoção e o subsequente ganho de popularidade da linguagem Python para disciplinas de programação nas universidades do Brasil e do mundo [Barbosa et al. 2014, Guo 2014].

*Projeto parcialmente financiado por FAPERJ.

Além da escolha da linguagem de programação para o ensino, a facilidade de uso e o conjunto de funcionalidades do ambiente de desenvolvimento utilizado também são fatores que podem influenciar os resultados obtidos. Ambientes de desenvolvimento destinados a projetos de grande porte oferecem numerosas opções avançadas que, apesar de úteis para desenvolvedores profissionais, tornam mais difícil a navegação e o uso dessas ferramentas por parte do programador iniciante. Uma plataforma de aprendizado deve oferecer um escopo de funcionalidades apropriadamente limitado, tendo em vista que a facilidade de uso se dá através da simplicidade do campo de interação [Kölling 2010].

A ferramenta *Online Python Tutor* (OPT) [Guo 2013] foi projetada com o objetivo de providenciar um ambiente simples para visualização e depuração de pequenos programas escritos em Python, que fosse acessível através do navegador sem necessidade de instalação prévia. Desde então, recebeu um número considerável de usuários, tendo inclusive seu uso incentivado por cursos de graduação como o da University of California, Berkeley. Além disso, sua arquitetura foi projetada para ser extensível, viabilizando a gradual implementação do suporte a outras linguagens de programação, incluindo Javascript, C e Ruby. Entretanto, o OPT possui algumas limitações, como a ausência de recursos para se visualizar uma estrutura de dados de maneira personalizada, mais apropriada à abstração subjacente. Por exemplo, para depurar um programa em que um vetor é usado como um *heap*, é desejável que este possa ser exibido no formato de árvore binária, que evidencia as ligações entre nós pai/filho ao contrário de sua representação vetorial.

Este trabalho introduz o TuPy Online[1], uma ferramenta construída com base na arquitetura de OPT com os seguintes objetivos:

1. introduzir e utilizar uma pseudolinguagem (o TuPy) que seja o mais compacta e expressiva possível, de modo que com um número reduzido de palavras-chave seja possível expressar algoritmos gerais encontrados nos livros-texto;
2. servir ao público brasileiro, por ser uma linguagem definida com comandos em português, de forma a incluir a parcela da população que não é proficiente em outros idiomas, em particular o inglês;
3. permitir que algoritmos possam ser estudados com execução passo-a-passo e que estruturas de dados possam ser visualizadas de maneira apropriada conforme a abstração que representam.

2. Trabalhos Relacionados

Desde a concepção do sistema interativo BALSA [Brown e Sedgewick 1984], que gerava representações de alto nível de estruturas de dados de um programa Pascal, a área da visualização de *software* vem apresentando um número crescente de desenvolvimentos.

A plataforma de educação ViLLE começou como uma ferramenta para visualização passo-a-passo de programas elaborados por professores [Rajala et al. 2007]. Além de elaborar os programas-exemplo, o professor também podia configurar questões de múltipla escolha que eram exibidas ao aluno após passos específicos da visualização. O aluno, por sua vez, tinha a capacidade de modificar os programas recebidos para observar como as suas mudanças impactavam a execução. Os autores reforçam a necessidade do foco no raciocínio lógico em detrimento das particularidades das sintaxes das linguagens

[1]Disponível em https://gvirtu.github.io/tupy. Acessado em 30 mar. 2018.

de programação. A plataforma pode ser configurada para aceitar qualquer linguagem e oferece um pseudocódigo limitado que pode ser usado no visualizador.

O JavaTool permite que o aluno construa seu programa usando uma simplificação da sintaxe tradicional Java. Dessa forma, alavanca o ensino das disciplinas iniciais de programação através de animações da sequência de comandos executados e de explicações geradas automaticamente [Mota et al. 2008].

O uso de IDEs de baixa complexidade também é sustentado por [Noschang et al. 2014] que oferecem, através da ferramenta Portugol Studio, uma notação do Portugol[2] baseada em linguagens como C e PHP, além de bibliotecas que possibilitam a construção de programas com interfaces gráficas e reprodução de sons. Acompanha o *software* um extenso conjunto de demonstrações de funcionalidades, assim como exemplos de jogos de duas dimensões criados com ele.

Ao contrário das abordagens citadas até agora, que fazem uso de linguagens imperativas, a sintaxe encontrada em Potigol [Lucena e Lucena 2016] dispõe de construções multiparadigma, com ênfase no paradigma funcional.

Sorva et al. revisam mais de 40 sistemas de visualização de programa (incluindo OPT, ViLLE e JavaTool) e defendem o engajamento do usuário como fator fundamental para a eficácia de uma ferramenta didática de visualização [Sorva et al. 2013]. Propõem uma taxonomia bidimensional de engajamento que chamam de 2DET, com alicerce em taxonomias existentes [Naps et al. 2002, Myller et al. 2009]. Em uma dimensão, 2DET classifica o nível de envolvimento direto do aprendiz com a visualização, levando em consideração o controle que ele pode exercer sobre a ferramenta. A segunda dimensão diz respeito à autoria do conteúdo, classificado quanto à possibilidade de ser predefinido, parametrizável, modificável, ou elaborado pelo usuário. Ademais, os autores afirmam que existe um problema de disseminação no domínio do desenvolvimento de ferramentas de visualização. Segundo eles, em sua maioria os sistemas revisados aparentam ter sido protótipos de pesquisa descartados após sua elaboração ou após a condução de um estudo de avaliação, e comunidades de código aberto dedicadas ao trabalho em tais sistemas são "quase não existentes".

Vale destacar que os projetos Portugol Studio[3], Potigol[4] e OPT[5] seguem o modelo de código aberto na plataforma pública GitHub, estimulando o desenvolvimento colaborativo a longo prazo. Entretanto, na data de acesso aos respectivos repositórios, nenhum havia registrado um número expressivo de contribuidores, corroborando com as conclusões de Sorva et al.

As demais ferramentas mencionadas constituem visualizadores de programas. Existe uma outra classe de ferramenta, os visualizadores de algoritmos, que enfatiza a ilustração conceitual de certos algoritmos sem a necessidade de estarem rigorosamente alinhados a uma linguagem ou código, ao invés de evidenciar a interpretação da sequência de comandos de um programa para o acompanhamento de seu estado [Diehl 2007]. Um exemplo desse tipo de ferramenta é VisuAlgo [Halim 2015], que oferece uma coleção de

[2]Uma pseudolinguagem para a escrita de algoritmos em português.
[3]Disponível em https://github.com/UNIVALI-LITE/Portugol-Studio. Acessado em 13 fev. 2018.
[4]Disponível em https://github.com/potigol/Potigol. Acessado em 13 fev. 2018.
[5]Disponível em https://github.com/pgbovine/OnlinePythonTutor. Acessado em 13 fev. 2018.

algoritmos com visualizações animadas e individualizadas de acordo com as abstrações de suas estruturas, permitindo ainda que o usuário parametrize suas entradas.

Como será discutido adiante, o TuPy Online é uma iniciativa que busca unir a flexibilidade dos visualizadores de programa, mantendo a possibilidade de construir e depurar programas, à expressividade dos visualizadores de algoritmo, providenciando recursos para criar visualizações personalizadas.

3. Arquitetura do TuPy Online

Para o melhor entendimento de como o TuPy Online foi estruturado, dois componentes relevantes devem ser discutidos: o OPT e o interpretador da linguagem TuPy.

3.1. Online Python Tutor

Como foi mencionado, o OPT recebeu suporte a novas linguagens além de Python ao longo dos anos. Um aspecto de sua arquitetura que favoreceu tal implementação é o fato da visualização não ser construída diretamente a partir da interpretação do programa Python, mas sim da interpretação de uma estrutura de representação intermediária, a qual é gerada a partir da execução íntegra do programa. A questão de estender o suporte a novas linguagens se dá, portanto, pela implementação de um mecanismo para geração dessa representação intermediária. Para as linguagens atualmente suportadas, existem componentes separados encarregados de executar o código com restrições de privilégios e implementar alguma estratégia para produzir a estrutura desejada.

A representação utilizada é um arquivo JSON que descreve o rastro da execução, isto é, contém uma sequência de informações acerca do estado de variáveis e da pilha de ativação para cada passo da execução do programa. Como essa execução acontece integralmente de forma não interativa, o usuário pode, em posse do arquivo retornado, navegar livremente pela visualização da execução, em passos para a frente ou para trás.

No OPT, o rastro da execução de um programa Python é obtido através da análise de dados fornecidos pelo depurador `bdb` ao executá-lo. No TuPy Online, a concepção de um interpretador permitiu a integração de um módulo que acompanha a execução dos programas e é capaz de produzir o arquivo de rastro de execução no formato desejado.

3.2. O Interpretador da Linguagem TuPy

Foi utilizada a ferramenta ANTLR4 [Parr 2013] para que, a partir de uma gramática EBNF[6], houvesse geração de código para cumprir as etapas de análise léxica e sintática do interpretador, gerando uma árvore de sintaxe abstrata (AST).

Cabe ao interpretador, em seguida, percorrer a AST e registrar a evolução de estado do programa. Para isso, foram implementadas abstrações para a tabela de símbolos e a pilha de ativação, que contém as informações necessárias não somente à execução mas também à produção do rastro de execução.

4. Análise da Ferramenta

A interface do TuPy Online é totalmente baseada no navegador e herda diversos aspectos do OPT. Conforme mostra a Figura 1, a página principal consiste do campo de edição

[6]Uma extensão do Formato Backus–Naur (BNF) que, dentre outras características, possui notações que facilitam opcionalidade e repetições em regras.

de texto e dos botões de ação. A parte inferior da página também conta com um manual da linguagem TuPy e uma coleção de exemplos representativos de diferentes algoritmos e estruturas de dados. Durante a visualização, um painel de código situa-se à esquerda com destaque às linhas atual e seguinte da sequência de execução, junto aos botões de navegação. Assim como no OPT, é possível configurar pontos de parada por meio de um clique na linha de código desejada. Algumas funcionalidades do OPT, no entanto, ainda não estão disponíveis no TuPy Online, como colaboração em tempo real e a possibilidade de embutir um visualizador em outra página Web através de código HTML.

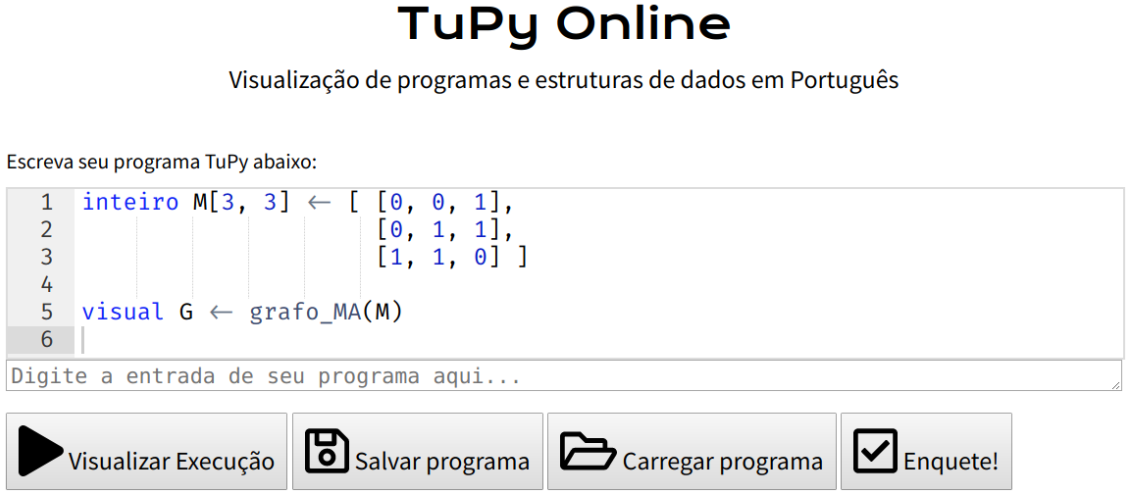


Figura 1. A interface de edição de código do TuPy Online.

Entre os diferenciais que foram implementados está a possibilidade de fornecer dados de entrada do usuário a serem lidos pelo programa, assim como botões para salvar e carregar o código-fonte de um arquivo de texto. O editor de código também foi adaptado para habilitar conveniências como preenchimento automático, expansão ou contração de blocos (*code folding*) e uma fonte[7] com suporte a ligaduras voltadas à programação, isto é, a capacidade de unir símbolos consecutivos de um código em um novo símbolo coerente automaticamente. A digitação dos símbolos correspondentes ao comparador de não-igualdade de operandos (`!=`), por exemplo, resulta na exibição do símbolo $\neq$. Com isso, espera-se que o código TuPy possa se aproximar visualmente de notações de pseudocódigo manuscrito. Além disso, foi disponibilizada uma versão que pode ser executada localmente, sem necessidade de conexão com a internet, minimizando problemas de escalabilidade oriundos do compartilhamento de recursos do servidor.

4.1. A Sintaxe da Linguagem TuPy

Assim como a documentação e todos os menus e textos do ambiente de desenvolvimento do TuPy Online, os comandos e as palavras reservadas presentes na linguagem estão em português, com suporte a caracteres *Unicode*, incluindo letras acentuadas. Essa deliberação se deu em vista do obstáculo linguístico encontrado em ferramentas existentes [Anido 2015]. A intenção é permitir o foco do programador aprendiz na estruturação

[7]Fira Code. Disponível em https://github.com/tonsky/FiraCode. Acessado em 18 fev. 2018.

e escrita do código, de forma a minimizar a carga cognitiva adicional oriunda da necessidade de mapear símbolos, vocábulos estrangeiros ou mnemônicos a um conjunto de comportamentos esperados.

Um dos objetivos originais do OPT era facilitar o processo de elaboração de conteúdo expositivo com o qual o docente precisa arcar, que requer esforço e é propenso a erros [Guo 2013]. Alinhada a esse objetivo, a linguagem TuPy foi derivada, após sucessivos refinamentos, de material que é aplicado a estudantes dos primeiros períodos do curso de Bacharelado em Ciência da Computação, e que foi elaborado por professores da Universidade do Estado do Rio de Janeiro. Possui, portanto, um conjunto restrito de funcionalidades que abrange o conteúdo tradicionalmente abordado por cursos introdutórios, incluindo comandos de repetição, suporte a recursividade e tipos compostos.

Em TuPy, o agrupamento de comandos em um mesmo bloco de código não necessita de marcadores de início e fim: foi adotado um estilo de sintaxe baseado em indentação similar ao encontrado em Python. O uso do ponto-e-vírgula como terminador de comando também não é obrigatório. Outras características importantes incluem atribuições paralelas, um sistema de tipos estáticos e a definição de tipos compostos com recursos do paradigma de orientação a objeto, como atributos e métodos, herança e polimorfismo.

Destinada ao uso integrado com o visualizador, a sintaxe de TuPy também permite ocultar certos elementos da sequência de visualização, incluindo valores de variáveis arbitrárias ou linhas de código inteiras. Dessa forma, um expositor pode destacar elementos específicos, filtrando funções, variáveis e estruturas auxiliares que não sejam essenciais ao entendimento de um algoritmo.

4.2. Visualização de Estruturas de Dados

Uma das limitações de OPT, nas palavras do autor, consiste na falta de visualização para "estruturas de dados sofisticadas". O projeto do TuPy Online buscou endereçar essa restrição providenciando uma biblioteca de funções para criar representações visuais extensíveis a partir de estruturas de dados.

É possível gerar visualizações de grafos (direcionados ou não), árvores, *heaps*, vetores, matrizes, listas encadeadas, pilhas e filas em qualquer passo da sequência de execução através de um único comando. A Figura 2 ilustra um caso de uso para essa funcionalidade: um grafo cujas arestas são descritas por uma matriz de adjacências. Note que a visualização OPT, embora visualize fielmente as variáveis que compõem o estado interno da estrutura (uma matriz, na forma de uma lista de listas), não sintetiza de forma compreensiva a abstração que ela representa (o grafo). O conjunto de funções de visualização que foram predisponibilizadas com a linguagem pode ser encontrado na Tabela 1. Em TuPy, uma mesma estrutura pode ser traduzida para diferentes formatos, como pode ser evidenciado também pela Figura 2, que ilustra a mesma matriz exibida como tabela e como grafo. As visualizações customizadas devem ser explicitamente especificadas pelo programador ao atribuir o resultado de uma das funções geradoras a uma variável, caso contrário a exibição OPT é mostrada por padrão.

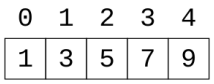
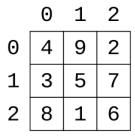
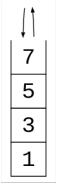
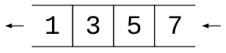
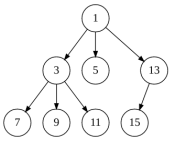
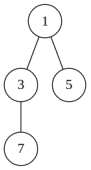
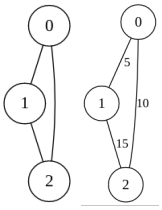
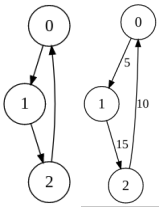
Estrutura	Função	Resultado
Vetor	`vetor(V)`	
Matriz	`matriz(M)`	
Lista Encadeada	`lista_encadeada(...)`	
Pilha	`pilha(V)`	
Fila	`fila(V)`	
Árvore	`árvore(...)`	
Heap	`heap(V)`	
Grafo	`grafo_MA(M)`	
	`grafo_LA(M)`	
	`grafo_valorado_MA(M)`	
	`grafo_valorado_LA(M)`	
Digrafo	`digrafo_MA(M)`	
	`digrafo_LA(M)`	
	`digrafo_valorado_MA(M)`	
	`digrafo_valorado_LA(M)`	

Tabela 1. Funções predefinidas para gerar visualizações de estruturas em TuPy. Foi utilizado V para simbolizar um parâmetro vetorial, M para um parâmetro em forma de matriz, e "..." para um parâmetro de tipo composto.

Internamente, as funções encapsulam lógica para percorrer as estruturas convertendo-as para descrições textuais, que são consumidas e transformadas em imagens durante a execução no navegador do usuário pelo Graphviz [Ellson et al. 2003], uma ferramenta para visualização de grafos e produção de diagramas estáticos descritos pela linguagem DOT [Gansner et al. 2015].

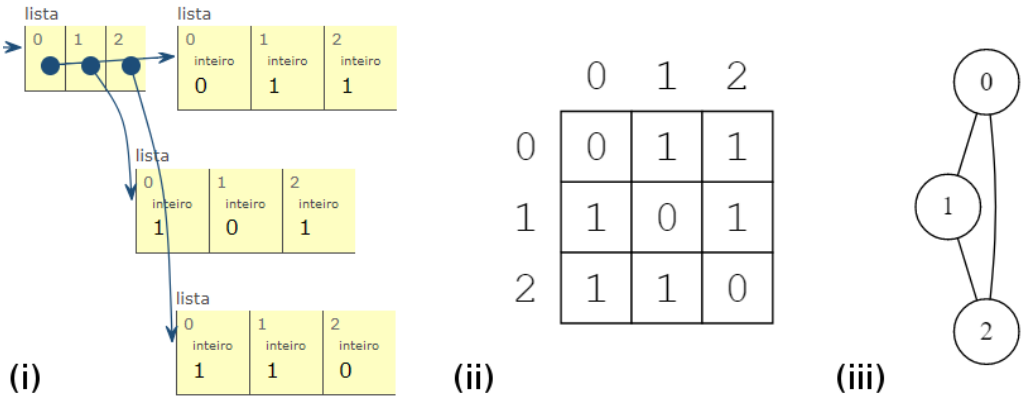


Figura 2. Visualização de uma matriz de adjacência de três formas no TuPy Online: a visualização tradicional do OPT (i), a visualização com a função de abstração `matriz` (ii), e com a função de abstração `grafo_MA` (iii).

O uso de descrições textuais na linguagem DOT, que possui uma extensa gama de opções para a customização dos formatos, das cores e do posicionamento dos elementos visuais, permite gerar visualizações parametrizadas pelo estado do programa. Como as funções de visualização geram programas DOT armazenados em cadeias de caracteres, é possível usar operações básicas como concatenação para manipular ou construir qualquer subprograma DOT válido em tempo de execução. Isso significa que, além das funções predefinidas para estruturas de dados convencionais, o programador pode criar novas funções capazes de exibir imagens com as características que desejar, complementando a visualização de forma integrada. Essa funcionalidade é observada na demonstração do problema das N rainhas [Weisstein 2002], incluído como um dos exemplos da ferramenta: uma função construída em TuPy é chamada pelo programa principal para criar iterativamente uma descrição de uma tabela cuja imagem resultante se assemelha a um tabuleiro de xadrez. Conforme se dá a execução do algoritmo, que usa a técnica de *backtracking* para posicionar N rainhas de xadrez em um mesmo tabuleiro N×N, as células do tabuleiro são enfatizadas através da mudança de cor na ordem de seus acessos pelo programa, e caracteres *Unicode* são usados para representar as peças atualmente posicionadas. Esse resultado pode ser visto na Figura 3.

5. Conclusões e Trabalhos Futuros

As finalidades do desenvolvimento do TuPy Online incluem a provisão de um ambiente para o ensino de programação que possa beneficiar tanto o professor quanto o aluno, oferecendo simplicidade e versatilidade suficientes para integrar o estudo de estruturas de dados a visualizações automáticas em um nível adequado de abstração. Tal recurso não substitui o depurador tradicional, mas pode ter um papel complementar na otimização de processos de construção de algoritmos até mesmo fora da sala de aula, tornando intuitivo o entendimento e a identificação de problemas em estruturas complexas. Adicionalmente, o

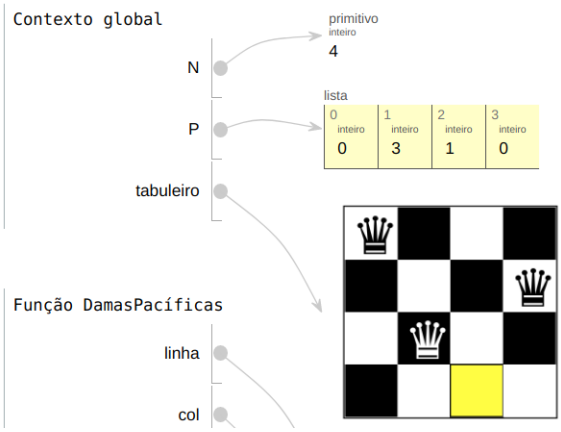


Figura 3. Tabuleiro de xadrez gerado por uma função TuPy durante a execução do algoritmo de backtracking para o problema *N* Rainhas (N=4)

projeto de código aberto, ao adaptar e integrar outras ferramentas consolidadas e testadas como OPT e Graphviz, visa promover um desenvolvimento estável e fomentar o reuso e o aprimoramento colaborativo de ferramentas existentes.

Entre os aspectos do projeto que podem ser futuramente aperfeiçoados estão: expressividade de mensagens de erro, ausência de explicações em linguagem natural para os comandos e interatividade na customização de visualizações.

Tem se fortalecido a crença de que os obstáculos para a adoção convencional de ferramentas de visualização não estão somente relacionados à abrangência de suas funcionalidades. Fatores como engajamento e facilidade de uso têm tido sua importância evidenciada na literatura. Espera-se que o TuPy Online possa, além de atender esses requisitos em algum nível, ser compatível com as necessidades curriculares e infraestruturais das universidades brasileiras, podendo assim ser uma contribuição catalisadora do ensino de algoritmos e estruturas de dados. Para que possa ser conduzido um estudo qualitativo da eficácia da ferramenta, o seu uso já está sendo implantado nos cursos introdutórios de computação do Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro.

Referências

Anido, R. (2015). Saci–ainda outro ambiente para o ensino de programação. *Anais do XXIII WEI - Workshop sobre Educação em Computação*.

Barbosa, A. d. A., Ferreira, D. Í., e Costa, E. B. (2014). Influência da linguagem no ensino introdutório de programação. Em *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 25, página 612.

Brown, M. H. e Sedgewick, R. (1984). *A system for algorithm animation*, volume 18. ACM.

Cunha, L. S., Tonetti, P., e Sanavria, C. Z. (2017). O ensino de informática no brasil: Uma análise da produção científica em eventos da sbc (2010–2014). *Anais do Computer on the Beach*, páginas 31–40.

Diehl, S. (2007). *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media.

Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., e Woodhull, G. (2003). Graphviz and dynagraph – static and dynamic graph drawing tools. Em *GRAPH DRAWING SOFTWARE*, páginas 127–148. Springer-Verlag.

Gansner, E., Koutsofios, E., e North, S. (2015). Drawing graphs with dot.

Guo, P. (2014). Python is now the most popular introductory teaching language at top us universities. *BLOG@ CACM, July*, página 47.

Guo, P. J. (2013). Online python tutor: Embeddable web-based program visualization for cs education. Em *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, páginas 579–584, New York, NY, USA. ACM.

Halim, S. (2015). Visualgo–visualising data structures and algorithms through animation. *OLYMPIADS IN INFORMATICS*, página 243.

Kölling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):14:1–14:21.

Lucena, L. R. e Lucena, M. (2016). Potigol, a programming language for beginners. Em *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, páginas 368–368. ACM.

Mota, M. P., Pereira, L. W. K., e Favero, E. L. (2008). Javatool: Uma ferramenta para o ensino de programação. Em *Congresso da Sociedade Brasileira de Computação. Belém. XXVIII Congresso da Sociedade Brasileira de Computação*, páginas 127–136.

Myller, N., Bednarik, R., Sutinen, E., e Ben-Ari, M. (2009). Extending the engagement taxonomy: Software visualization and collaborative learning. *ACM Transactions on Computing Education (TOCE)*, 9(1):7.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., et al. (2002). Exploring the role of visualization and engagement in computer science education. Em *ACM Sigcse Bulletin*, volume 35, páginas 131–152. ACM.

Noschang, L. F., Pelz, F., e Raabe, A. e. a. (2014). Portugol studio: Uma ide para iniciantes em programação. *Anais do CSBC/WEI*, páginas 535–545.

Parr, T. (2013). *The definitive ANTLR 4 reference*. Pragmatic Bookshelf.

Rajala, T., Laakso, M.-J., Kaila, E., e Salakoski, T. (2007). Ville: a language-independent program visualization tool. Em *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88*, páginas 151–159. Australian Computer Society, Inc.

Sorva, J., Karavirta, V., e Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, 13(4):15:1–15:64.

Weisstein, E. W. (2002). Queens problem. *MathWorld–A Wolfram Web Resource*. Disponível em http://mathworld.wolfram.com/QueensProblem.html. Acessado em 22 fev. 2018.

Performance Analysis of Computer Science Students in Programming Learning

Air Rabelo[1], **Luiz Claudio Gomes Maia**[1], **Fernando Silva Parreiras**[1]

[1]Department of Computer Science - FUMEC University
Av. Afonso Pena 3880 - 30130-009 - Belo Horizonte - MG - Brazil - +55 31 3269-5230

air@fumec.br, luiz.maia@fumec.br, fernando.parreiras@fumec.br

Abstract. *The difficulties faced by lecturers and students in order to teach and learn programming on computer science courses have been a research topic over the years. The hardship to understand the abstract and logic concepts and consequent demotivation has been resulting in high rates of novices' failure and class abandonment. This study adopted statistical concepts to analyze students' final grades in programming subjects and compare their performance. Data were gathered from a computer science course at a Brazilian University. The period analyzed was from 2010 to 2015 including six programming subjects from the first and second academic year. The results pointed a significant number of student failure (43%) and abandonment (25%). It was also discovered that even with different teachers, semesters and programming subjects, the students' performance mean were nearly equal. The discoveries of this work contributed to point the hardship faced by students and teachers to learn and teach programming.*

1. Introduction

The learning programming process is a challenge activity to novice programmers [Lahtinen et al. 2005]. The students from programming courses face hardship to understand and apply the abstract and logic concepts required [Medina et al. 2013]. The difficulties refers to syntax and semantics structures relating to programming languages [Xinogalos et al. 2015]. Piteira and Costa conduct a study and identified the greater difficulties perceived by students: designing a program to solve a task, understanding programming structures and learning the programming language syntax [Piteira and Costa 2013]. The learning process suggests use of practices to enable the students to deal with situations and problems in which an algorithm could be a solution [Medina et al. 2013]. Students identify the practical sessions as one of the most useful learning methods [Piteira and Costa 2013].

The issues faced by novice programming students, who are not able to cope with the natural difficulties associated, commonly leads many to demotivation [Gomes and Mendes 2014]. Some individual factors contribute to student engagement such as achievement, motivation and goals [Eltegani and Butgereit 2015]. There are frustration when the teaching does not meet students' expectations, which reduce engagement and affects the success or failure rates in programming [Eltegani and Butgereit 2015]. A research conduct by Eltegani and Butgereit found the following attributes required in programming learning: a)time to study; b)receiving feedback and support; c)intensive practice; d)reduce fear that is defined as lack of interest in programming; e)promoting self

confidence and problem solving skills; f)availability of devices and resources to practice outside class; g)feeling of pressure/no pressure; g)language used in teaching should be easy [Eltegani and Butgereit 2015].

The objective of this research is to perform a statistical analysis regarding the performance of computer science students in programming learning. The data were gathered from a Brazilian University, including first and second academid years' grades on programming subjects from 2010 to 2015.

The research questions are: a)What was the general performance of the students in the analyzed period? b)What was the failure and success rates? c)Was there a performance variation among students from first and second year, or from distinct teachers?

2. Methodology

This paper adopted statistical concepts to analyze students' grades of introductory programming subjects in a computer science course. The main object was to compare rates of failure and success. The data were gathered from a computer science course at a Brazilian University. The period analyzed was from 2010 to 2015 including six different programming subjects applied on the first and second academic year. In this Brazilian University the terms of programming subject takes six months and they are called as semester. There are two terms per year, one in the first semester ane other in the second semester. The data were organized and grouped by semester, such as first semester of 2010 and second semester of 2010.

The database had 4396 rows covering 8 semesters and six different programming topics. The data were disposal in the following columns: year and semester, names of subject, teacher and student, first test grade, second test grade, third test grade and final grade. The data were placed in a CSV (Coma Separated Value) file. The R programming language and software environment for statistical computing and graphics, and the R Studio IDE (Integrated Development Environment) was adopted to data manipulation and graphics generation. The CSV file was imported to R Studio and some R programs were developed to reach the results presented in this paper.

In the first graph was presented a histogram with final grades of all students and terms from the database. In second graph, full length database were used to present the following items: a)a global mean of students grades, b)a success mean rate, c)a failure mean rate, d)a abandon mean rate. From the third to fifth analysis the same items were demonstrated, but at that time the objective was to compare a performance analysis between different teachers, terms and topics. In the following analysis, a data sample imported to R Studio was used for the tests ANOVA (Analysis of Variance), Shapiro-Wilk, Levene and Tuckey. The object of those tests was to verify if there was any relevant performance difference between students from distinct semester and teacher.

3. Results of the Analysis

Including all data in the database, 4396 student grades in the first and second years on computer science programming subjects from 2010 to 2015, a histogram was presented in the figure 1 considering only the final grades. The grades go from 0 to 100, and students have to reach 60 at least to success. The highest bar displayed in the figure 1 represents

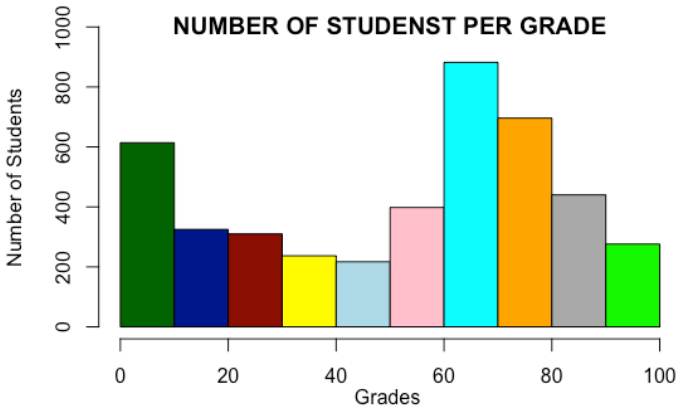


Figure 1. Histogram of Grades

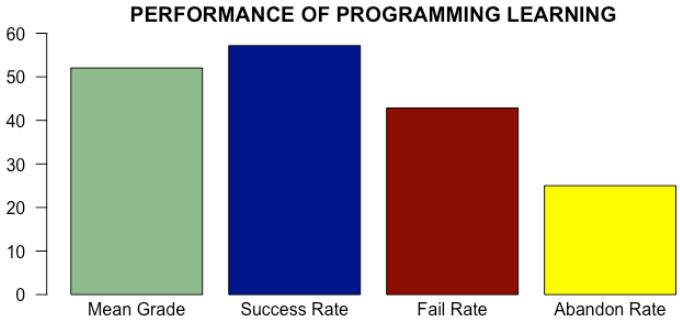


Figure 2. Students performance in programming learning

grades between 60 and 70, the next highest bar 70 to 80, they both were success grades. There were a significant number of students' grades from 0 to 59, which represent failure. This figure also revealed in the first left bar, grades from 0 to 10. There were around 600 students (in a total of 4396) in this situation of very poor outcome, which represent 13.6%.

In the figure 2 was presented the mean grade, rates of success, failure and abandoning. The expressive number of students with grades below 60% shown in figure 1 reduced the mean grade to 52%. The success rate was 57% and 43% of failure. Despite the success had been higher than fail, they were very near to each other. The abandoning rate was 25%, which means from those 43% of failure, 25% were students who had abandoned the programming subject before the end of semester. In this case those students quitted the class because they concluded themselves would not be possible to reach the minimal grade to success. The other 18% (43% - 25% = 18%) remained in the class until finish but failed at the end.

To verify if there were any difference on students performance considering the teachers involved in those programming subjects, the figure 3 exposed the mean grade, rates of success, failure and abandon, separated for each of the ten professors who taught programming classes in the analyzed period. The students of professors number 2, 5 and 9 had a failure rate higher than success. In the other hand, the students from the other seven professors' classes had just the opposite performance. The worst case scenario was presented in the professor 9 class, which the students had 32% of success and 68% of fail,

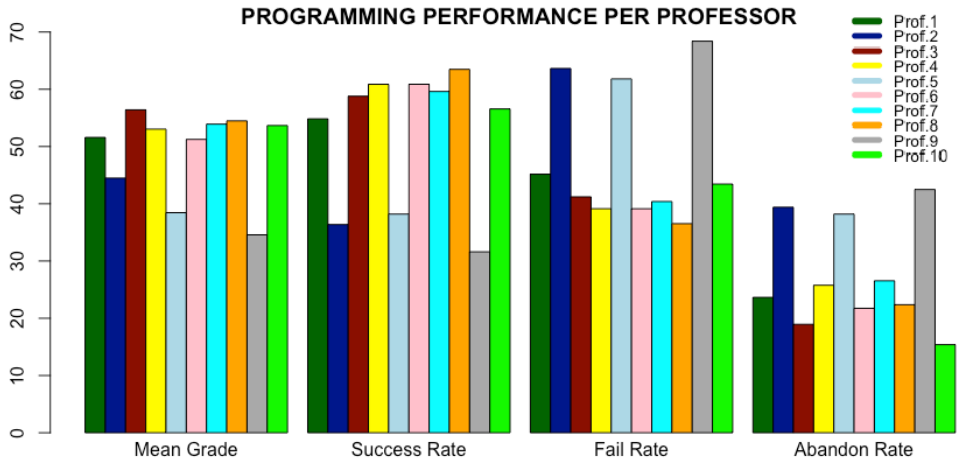


Figure 3. Students performance in programming learning per professor

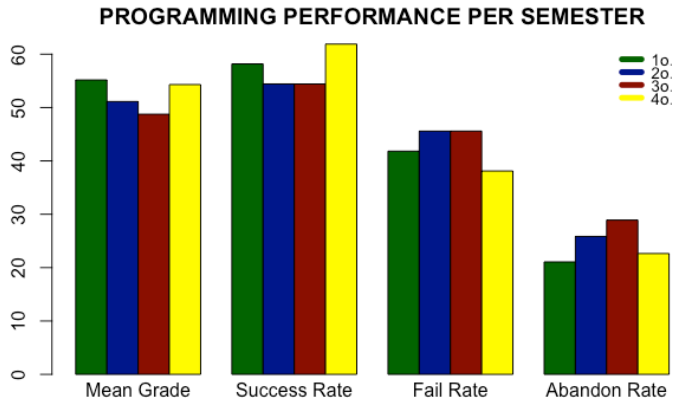


Figure 4. Students performance in programming learning per semester

42% of abandoning. The best case scenario was in the professor 8 class, which had 63% of success and 38% of fail, 22% of abandoning. In those cases the outcomes were almost the opposite each other. The reasons for that could be adoption of different assessment methods or criteria to assign grades. Despite those professors number 2, 5 and 9, in which students had more fail than success, in the others seven, which means 70% of professors, the students had similar performance.

A similar analysis of figure 3 was presented in figure 4. In this case the mean grade, rates of success, failure and abandon were presented per semester. The objective was to verify if students had different performance among the fourth semesters in which included the first and second years of the computer science course. Therefore, the figure 4 revealed a similar performance in all four semesters. Consequently, there was no indication that novice students from first semester had worst performance than others with some programming experience such as those from fourth semester.

In the figure 5 the mean grade, rates of success, failure and abandon were presented per subject. The subject Prog.I was from first semester, Prog.II was from second semester, Prog.III and Prog.V were from third semester, Prog.IV and Prog.VI were from fourth semester. The objective was to verify if students had different performance among those six programing subjects structured on the first two academic years of the computer

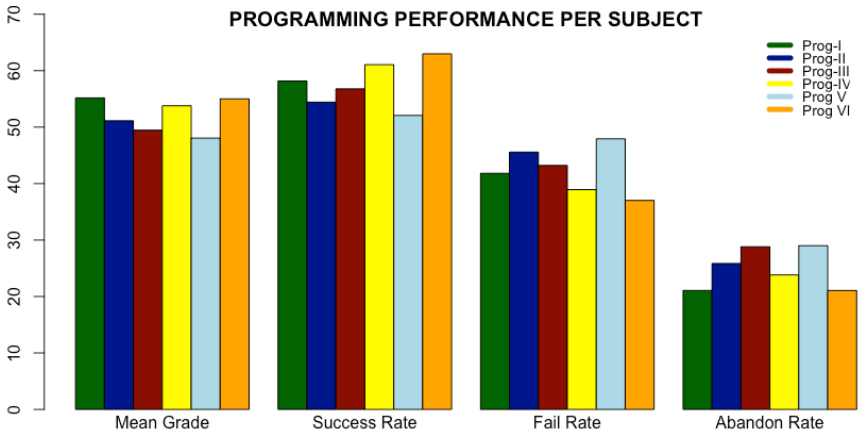


Figure 5. Students performance in programming learning per course subject

science course. In the same way of figure 4, the figure 5 shows a very similar performance of students between the six programming subjects analyzed. Despite the initial programming subjects such as Prog.I and Prog.II, in which programming concepts had less complexity than subjects Prog.IV and Prog.VI from fourth semester, students' performance was very similar among them. This indicated that as long as students gain more programming experience and apprenticeship, as time goes by, the complexity of the subjects also grew up, keeping the performance level similar between the six programming subjects analyzed.

To certify if the students performance were indeed similar considering different semesters and subjects, as it was presented in figure 4 and figure 5, a sample of database was isolated to perform an ANOVA test (Analysis of Variance). The ANOVA was used to compare means across groups of semesters and subjects. The ANOVA test was used to determine whether the data provided strong evidence of difference between the means. Otherwise, the null hypothesis is considered, indicating that all means are equal [Diez et al. 2012]. In the first sample was performed an ANOVA two-way. When an ANOVA is performed between groups using two factors it is called ANOVA two-way [Diez et al. 2012]. In this case, semester was one factor and subject the other. This sample gathered the six programming subjects in three sequential semesters: 2014 second, 2015 first and 2015 second. The figure 6 presented the box plot graph of the students' final grades mean on the six programming subjects across the three groups of semesters. The graph exhibits different mean grades among each group, 2015/1o was the lower one whilst 2014/2o and 2015/2o were similar.

The figure 7 presented the box plot of students final grade on the tree semesters across the six groups of programming subjects. The graph also exhibited different mean grades among each group, subject Prog I was the lower one and Prog VI the higher within more than ten percent points of difference. The ANOVA summary showed whether that differences were statistically significant.

The figure 8 exhibits the ANOVA summary for testing whether the mean of student final grade differs across semesters and programming subjects. The summary last right column showed the p-value. According to [Diez et al. 2012] whether the p-value is larger than 0.05, denotes that the evidence is not strong enough to reject the null hypoth-

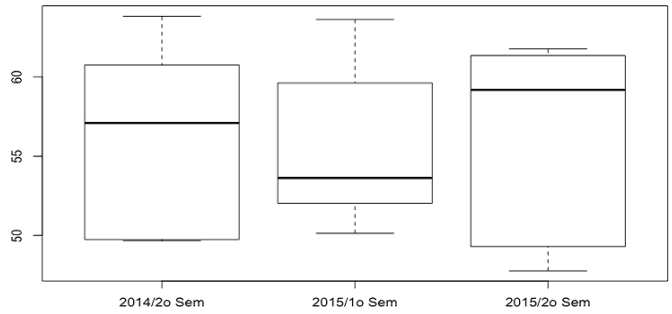


Figure 6. Side-by-side box plot of the students final grades on the six programming subjects across the three groups of semesters

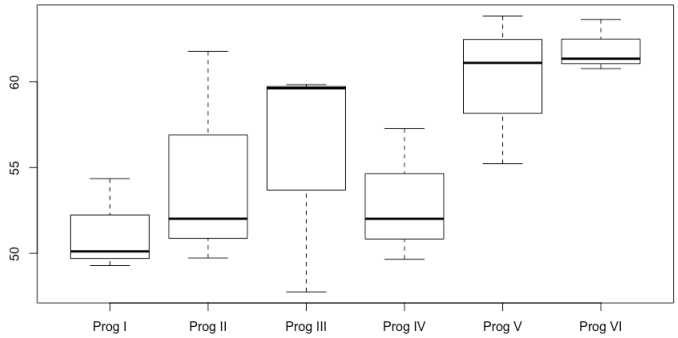


Figure 7. Side-by-side box plot of students final grade on the tree semesters across the six groups of programming subjects

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
tabela$Semester	2	3.66	1.83	0.07	0.933
tabela$Subject	5	256.13	51.23	1.95	0.172
Residuals	10	262.63	26.26		

Figure 8. The ANOVA summary for testing whether mean of student final grade differs across semesters and programming subjects

esis. The semester p-value was 0.933, which indicates a probability of 93.3% for null hypothesis, and subject p-value was 0.172, then the probability was 17.2%. Therefore, can be concluded that the data do not provide strong evidence indicating the mean of student final grade varied by semester or programming subject and the difference presented were due to chance.

As the first sample indicated an equivalent performance among students, in a second one was performed another ANOVA two-way also with semester and programming subject factors. This second sample gathered four subjects from first semester of 2012 to second of 2015. The figure 9 presented the box plot graph of the students' final grades mean on the four programming subjects across the eight groups of semesters. The graph also exhibited different mean grades among each group, 2013/1o was the lower one whilst 2014/1o the highest.

The figure 10 presented the box plot of students' final grade on the eight semesters across the four groups of programming subjects. The subject Prog II was the lower one and Prog VI the higher.

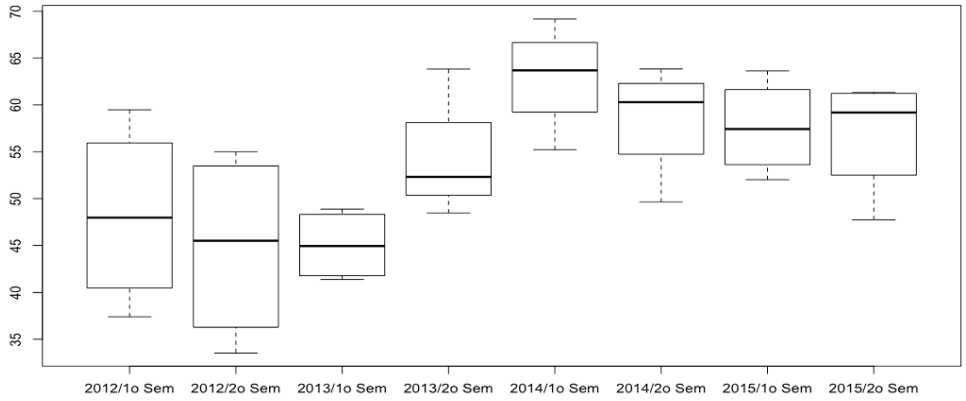


Figure 9. Side-by-side box plot of the students' final grades on the four programming subjects across the eight groups of semesters

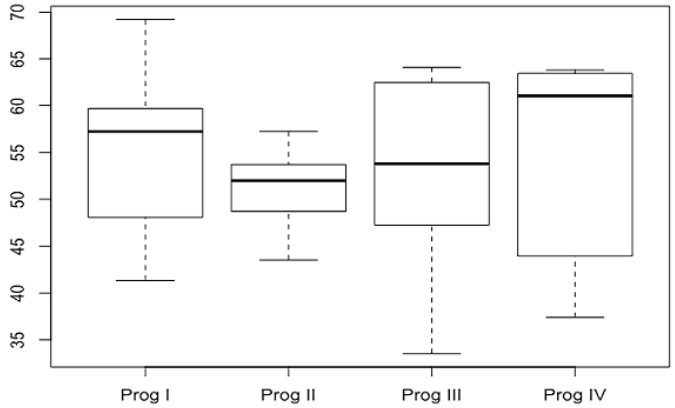


Figure 10. Side-by-side box plot of students' final grade on the tree semesters across the six groups of programming subjects

```
                Df Sum Sq Mean Sq F value Pr(>F)
tabela$Subject   3   76.3   25.42   0.480 0.6996
tabela$Semester  7 1267.8  181.11   3.421 0.0134 *
Residuals       21 1111.9   52.95
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 11. The ANOVA summary for testing whether mean of student final grade differs across semesters and programming subjects

The ANOVA summary exhibited in figure 11 showed whether that differences were statistically significant. The p-value for subject was 0.6996, which indicated a probability of 69.96% for null hypothesis. The semester p-value was 0.0134, so less than 0.05, then the null hypothesis has to be discarded and there was strong evidence that the different means in each of the eight semesters was not simply due to chance. In this case, in at least two semesters had been occurred different performance between students.

According to [Diez et al. 2012], there are three conditions to be checked for an ANOVA analysis: all observations must be independent, the data must be nearly normal, and the variance within each group must be homogeneous. To verify normality, a Shapiro-Wilk test was performed. The normality is reached when the p-value is higher than 0.05

```
        Shapiro-Wilk normality test

data:  resid(dados.anova)
W = 0.96787, p-value = 0.4427
```

Figure 12. The Shapiro-Wilk summary for testing data normality

```
> leveneTest(tabela$Performance ~ tabela$Semester, data=tabela)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  7  1.1467 0.3684
      24
```

Figure 13. The Levene summary for testing homogeneity of variance in semester

```
> leveneTest(tabela$Performance ~ tabela$Subject, data=tabela)
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  3  1.0768 0.3749
      28
```

Figure 14. The Levene summary for testing homogeneity of variance in subject

[Shaphiro and Wilk 1965], as presented in figure 12 p-value was 0.4427, so the data were normal.

To verify whether the variance within each group were homogeneous, a Levene test was performed. The homogeneity is reached when the p-value is higher than 0.05 [Brown and Forsythe 1974]. As presented in figure 13 the semester p-value was 0.3684, and figure 14 presented the subject p-value 0.3749, so the variance of semester and subject groups were both homogeneous.

Having confirmed the normality of data and homogeneity of variance, to discover in which pair of semesters had a statistical significant difference, a post-hoc Tukey test were performed [Tukey 1977]. The Tukey test performed all possible pairwise comparisons within both factors semester and programming subject . The figure 15 presented the Tukey test summary with six pairwise tests of subjects and twenty eight of semesters. All the comparison between each programming subjects displayed the p-value higher than 0.05, pointing that this differences were statistically insignificant. On the other hand, from the twenty eight pairwise semester comparison, two had presented p-value lower than 0.05: 2014/1o - 2012/2o and 2014/1o - 2013/1o. It was a statistical significant evidence that the students from semester 2014/1o had a significant better performance in programming subjects comparing to 2012/2o and 2013/1o.

4. Conclusions and future work

This research pointed a significant number of student's failure (43%), which include an also expressive rate of subject abandonment (25%). The first ANOVA test performed analyzed means of student's final grades from three semesters and six programming subjects. The summary showed no statistical significant difference between those student's performance. Then, a second ANOVA and Tukey tests were performed including eight semesters and four programming subjects. It was pointed a difference of student's performance in two of twenty eight pairwise comparisons between semesters, which represents 7.14%. So, 92.86% of the students were considered to have had a nearly equal perfor-

```
Tukey multiple comparisons of means          95% family-wise confidence level
   Subject              diff        lwr       upr     p adj
PROG II -PROG I    -3.8789871 -14.019935  6.261960 0.7132303
PROG III -PROG I   -1.9883564 -12.129304  8.152591 0.9464616
PROG IV -PROG I    -0.3201528 -10.461100  9.820795 0.9997475
PROG III -PROG II   1.8906307  -8.250317 12.031578 0.9534272
PROG IV -PROG II    3.5588344  -6.582113 13.699782 0.7631697
PROG IV -PROG III   1.6682037  -8.472744 11.809151 0.9672029

   Semester                 diff         lwr      upr     p adj
2012/2o Sem-2012/1o Sem -3.3095244 -20.5673949 13.94835 0.9976730
2013/1o Sem-2012/1o Sem -3.1683387 -20.4262093 14.08953 0.9982301
2013/2o Sem-2012/1o Sem  6.0254251 -11.2324454 23.28330 0.9315474
2014/1o Sem-2012/1o Sem 14.7376129  -2.5202576 31.99548 0.1313938
2014/2o Sem-2012/1o Sem 10.3166160  -6.9412545 27.57449 0.5016241
2015/1o Sem-2012/1o Sem  9.4178425  -7.8400281 26.67571 0.6081465
2015/2o Sem-2012/1o Sem  8.6631753  -8.5946952 25.92105 0.6969499
2013/1o Sem-2012/2o Sem  0.1411856 -17.1166849 17.39906 1.0000000
2013/2o Sem-2012/2o Sem  9.3349495  -7.9229210 26.59282 0.6180338
2014/1o Sem-2012/2o Sem 18.0471373   0.7892668 35.30501 0.0362584
2014/2o Sem-2012/2o Sem 13.6261404  -3.6317302 30.88401 0.1936296
2015/1o Sem-2012/2o Sem 12.7273668  -4.5305037 29.98524 0.2591516
2015/2o Sem-2012/2o Sem 11.9726997  -5.2851708 29.23057 0.3252506
2013/2o Sem-2013/1o Sem  9.1937639  -8.0641067 26.45163 0.6348321
2014/1o Sem-2013/1o Sem 17.9059517   0.6480811 35.16382 0.0384239
2014/2o Sem-2013/1o Sem 13.4849547  -3.7729158 30.74283 0.2029840
2015/1o Sem-2013/1o Sem 12.5861812  -4.6716893 29.84405 0.2707499
2015/2o Sem-2013/1o Sem 11.8315141  -5.4263565 29.08938 0.3387161
2014/1o Sem-2013/2o Sem  8.7121878  -8.5456827 25.97006 0.6913087
2014/2o Sem-2013/2o Sem  4.2911909 -12.9666797 21.54906 0.9888466
2015/1o Sem-2013/2o Sem  3.3924173 -13.8654532 20.65029 0.9972853
2015/2o Sem-2013/2o Sem  2.6377502 -14.6201203 19.89562 0.9994537
2014/2o Sem-2014/1o Sem -4.4209969 -21.6788675 12.83687 0.9867605
2015/1o Sem-2014/1o Sem -5.3197705 -22.5776410 11.93810 0.9634448
2015/2o Sem-2014/1o Sem -6.0744376 -23.3323081 11.18343 0.9288251
2015/1o Sem-2014/2o Sem -0.8987735 -18.1566441 16.35910 0.9999996
2015/2o Sem-2014/2o Sem -1.6534407 -18.9113112 15.60443 0.9999758
2015/2o Sem-2015/1o Sem -0.7546671 -18.0125377 16.50320 0.9999999
```

Figure 15. The Tukey test summary

mance. Therefore, regardless whether the programming subject is from first or second year of the course, or the semester analyzed, the means of student's final grades in programming learning had no statistical significant difference. It was analyzed whether there was any difference among mean grades considering different teachers, the majority of results also indicated a similar student performance.

The difficulty faced by novices on programming learning results in high rates of failure and abandonment. This has been addressed in several researches from different countries, such as [Tucker 1993], [Holland et al. 2009], [Stankov et al. 2015], [Guzdial and Guo 2014], [Robins et al. 2003], [Lahtinen et al. 2005], [Medina et al. 2013], [Xinogalos et al. 2015], [Miller et al. 1994]. Some of those researches also propose a methodology, computer tool or framework to optimize the effectiveness of teaching and learning programming. They also recommend a continuity of works in this area aiming to find better solutions to improve the student's performance in programming learning.

This work gathered data of students' final grades from a computer science course in a Brazilian university. Despite the data involved ten different teachers, the teaching methods were very similar and the assessment criteria were nearly equals. This fact favored the comparative analysis presented in this research and place reliability to discoveries reached, which corroborates the hardship of programming learning.

The data include eight semesters, six programming subjects from first to second year and ten distinct teachers. However, this study is limited by the period of the data analysed from 2010 to 2015 and by the one computer sicence course from one Brazilian university. Future works including undergraduate students from other courses and universities, covering a wide range of years, adopting different teaching methods, frameworks and other computing learning resources, would contribute to discover other results and complete the conclusions reached in this work.

References

Brown, M. B. and Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346):364–367.

Diez, D. M., Barr, C. D., and Cetinkaya-Rundel, M. (2012). *OpenIntro statistics*. CreateSpace.

Eltegani, N. and Butgereit, L. (2015). Attributes of students engagement in fundamental programming learning. In *Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference on*, pages 101–106. IEEE.

Gomes, A. and Mendes, A. (2014). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–8. IEEE.

Guzdial, M. and Guo, P. (2014). The difficulty of teaching programming languages, and the benefits of hands-on learning. *Commun. ACM*, 57(7):10–11.

Holland, J., Mitrovic, A., and Martin, B. (2009). J-latte: a constraint-based tutor for java. In *17th International on Conference Computers in Education (ICCE 2009)*, pages 142–146.

Lahtinen, E., Ala-Mutka, K., and Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *SIGCSE Bull.*, 37(3):14–18.

Medina, C. F., Pérez, J. R. P., Álvarez Garc'ıa, V. M., and del Puerto Paule Ruiz (2013). Assistance in Computer Programming Learning Using Educational Data Mining and Learning Analytics. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 237–242, New York, NY, USA. ACM.

Miller, P., Pane, J., Meter, G., and Vorthmann, S. (1994). Evolution of novice programming environments: The structure editors. In *of Carnegie Mellon University*, pages 140–158.

Piteira, M. and Costa, C. (2013). Learning computer programming: Study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ISDOC '13, pages 75–80, New York, NY, USA. ACM.

Robins, A., Rountree, J., and Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172.

Shaphiro, S. and Wilk, M. (1965). An analysis of variance test for normality. *Biometrika*, 52(3):591–611.

Stankov, E., Jovanov, M., Kostadinov, B., and Madevska Bogdanova, A. (2015). A new model for collaborative learning of programming using source code similarity detection. In *Global Engineering Education Conference (EDUCON), 2015 IEEE*, pages 709–715.

Tucker, S. A. (1993). Evaluation as feedback in instructional technology: The role of feedback in program evaluation. *Interactive instruction and feedback*, pages 105–132.

Tukey, J. W. (1977). Exploratory data analysis.

Xinogalos, S., Malliarakis, C., Tsompanoudi, D., and Satratzemi, M. (2015). Microworlds, games and collaboration: Three effective approaches to support novices in learning programming. In *Proceedings of the 7th Balkan Conference on Informatics Conference*, BCI '15, pages 39:1–39:8, New York, NY, USA. ACM.

Formação Docente na Pós-Graduação *Stricto Sensu* em Ciência da Computação: um recorte das regiões Norte e Nordeste

Pauleany S. Morais[1], Jean C. S. Rosa[2], Anna Raquel S. Marinho[3], Ecivaldo Matos[2]

[1]*Campus* de Educação a Distância - Instituto Federal do Rio Grande do Norte (IFRN)
Natal - Rio Grande do Norte - Brasil

[2]Departamento de Ciência da Computação - Universidade Federal da Bahia (UFBA)
Salvador - Bahia - Brasil

[3]*Campus* Zona Norte - Instituto Federal do Rio Grande do Norte (IFRN)
Natal - Rio Grande do Norte - Brasil

pauleany.morais@ifrn.edu.br, jean.rosa@ufba.br,
raquelmarinho.linfor@gmail.com, ecivaldo@ufba.br

Resumo. *A formação docente necessária aos professores que atuam/atuarão no sistema de ensino superior brasileiro está presente em inúmeras discussões acadêmicas. Nesse sentido, este artigo apresenta resultados de uma investigação sobre as iniciativas de formação docente nos cursos de pós-graduação stricto sensu em Ciência Computação de instituições federais de ensino superior das regiões Norte e Nordeste do Brasil. Para isso foi realizada a análise de documentação dos cursos de mestrados e doutorados acadêmicos. Identificou-se que determinados cursos de pós-graduação tentam sanar essa limitação de formação docente por meio de disciplinas sobre didática no ensino superior, alguns implementam atividades como estágio docente e outros não possuem quaisquer iniciativas voltadas à formação docente dos seus estudantes.*

Abstract. *There are several scientific studies about the Brazilian teacher training for higher education. In this sense, this paper presents results of an investigation about the initiatives of teacher training in Computer Science graduate programs of federal higher education institutions of North and Northeast of Brazil. For this, one documentary analysis on pedagogical documentos of that programs was carried out. It was identified that a few graduate courses (master's and doctoral) have some targeted actions for teacher training, with disciplines on didactics in higher education or activities of supervised teaching internship. The most programs do not have any initiative aimed at teacher training of their students.*

1. Introdução

A formação de professores tem enfrentado grandes desafios ao longo dos anos, principalmente no que se refere ao exercício da docência por bacharéis e tecnólogos nas diversas modalidades de ensino. Pimenta e Anastasiou (2014) apresentam estudos sobre

as especificidades da Docência no Ensino Superior com concreta vivência da ensinagem. Destacam a relevância da compreensão do processo de ensinagem e seleção de determinadas estratégias necessárias à atuação docente, particularmente, no Ensino Superior.

Para Imbernón (2012), o professor do Ensino Superior deve conscientizar-se da necessidade de dominar conhecimentos didáticos em sua área de atuação e sua interação com a disciplina, a própria docência e as interações vivenciadas junto aos discentes. No Brasil, a formação docente para o Ensino Superior ocorre na formação em nível de pós-graduação *stricto sensu* (mestrados e doutorados acadêmicos) (BRASIL, 1965).

No entanto, na área de Ciência da Computação, de modo geral, os programas de pós-graduação (PPG) em Ciência da Computação no Brasil não possuem políticas de formação de professores para além de ações pontuais, como a instituição de estágio (ou tirocínio) docente, ou disciplinas optativas de Metodologia do Ensino Superior (ou Didática do Ensino Superior) (MASSA, 2015).

De maneira geral, observa-se que os professores bacharéis e tecnólogos condicionam sua atuação profissional docente às experiências vivenciadas como alunos. Essa atitude é típica dos professores sem formação pedagógica específica, assim como ocorre no ensino de computação na Educação Profissional (SILVA, MATOS e MASSA, 2018). Tal condição pode desqualificar o ato educativo diante da incompletude das exigências necessárias à docência. De acordo com Reche e Vasconcelos (2014), as situações educacionais necessitam de uma tomada de posição do professor. Nesse contexto, o professor não licenciado possivelmente não vivenciou possibilidades de formação inicial que exigissem posicionamentos em ensaios de práticas educacionais.

Este artigo apresenta uma reflexão sobre a necessidade de formação docente nos PPG que atendam às exigências específicas do Ensino Superior e práticas curriculares condizentes ao ensino de Ciência da Computação com qualidade. Além disso, também são apresentados resultados de análise documental sobre os objetivos dos cursos de pós-graduação em Ciência da Computação e as suas iniciativas de formação docente. Dada a dimensão territorial do Brasil, bem como o número de PPG na área, realizou-se um recorte nas Regiões Norte e Nordeste do país.

Na próxima seção, serão tratados os significados da formação profissional docente no Ensino Superior. Na Seção 3 enfatiza-se as diretrizes normativas e os mecanismos para formação docente no Ensino Superior nos PPG de Ciência da Computação em Instituições Federais de Ensino Superior do Norte e do Nordeste. A Seção 4 é destinada à apresentação da metodologia de pesquisa e as Seções 5 e 6, respectivamente, tratam sobre os resultados identificados e as considerações finais.

2. Significados da Formação Profissional Docente no Ensino Superior

A formação profissional docente representa um processo em que o sujeito constrói um conhecimento pedagógico especializado que deve ser vivenciado no espaço das instituições formadoras ou mesmo orientado por elas. Os estudos que investigam essa formação mostram que o foco do exercício profissional docente deve estar na realidade social, devido à intensa relevância do processo ação-reflexão-ação (RAMALHO; NUÑEZ; GAUTHIER, 2003).

Para alcançar a formação profissional, o projeto de formação de professores deve articular de maneira integrada o exercício da profissão do magistério, a formação continuada, as atividades de pesquisas pedagógicas e as atividades de pesquisas colaborativas, enquanto processos constantes, necessariamente refletidos e questionáveis que possibilite a formação do professor como agente social (VEIGA, 2002). De maneira geral, o pensamento que se tenta realçar até essas linhas pode ser traduzido por meio da Figura 1, pois ilustra essa proposição de acordo com os estudos de Veiga (2002, p. 30):

Figura 1. Modelo de formação profissional do professor como agente social (fonte: VEIGA, 2002).

Nessa compreensão, pode-se dizer que a formação centrada na práxis envolve todas as condições necessárias ao desenvolvimento de programas de formação, pois é o campo empírico que por excelência proporciona respostas às demandas da própria instituição, como a elevação da qualidade do ensino e da aprendizagem em sala de aula, bem como deve ser o campo empírico para responder às necessidades das instituições superiores (IMBERNÓN, 2004). Isto é, formação profissional com foco nos saberes da experiência no próprio fazer pedagógico.

Esse profissional deve refletir sobre a sua prática didática, pesquisando os elementos da ação docente nos próprios contextos nos quais ocorrem, no sentido de qualificar o ato de ensinar e de aprender (PIMENTA; GHENDIN, 2002). Defende-se que a formação docente possa preparar os sujeitos em suas múltiplas dimensões para atuação profissional qualificada, assim fazê-los vivenciar o termo "dodiscência" (docência-discência), no sentido de representar a relação dialética entre ensinar e aprender (FREIRE, 1996).

O repensar da práxis do professor requer espaços para que ele possa experienciar o conhecimento em suas várias possibilidades, tornando-se pesquisador de sua própria prática, construindo um olhar crítico e intencional reflexivo sobre ela (PIMENTA; GHEDIN, 2002). Esse processo de fortalecimento da prática de ensino, considera que o fazer pedagógico do professor-aluno não é linear, mas gradual, envolvendo avanços, recuos, conflitos, desequilíbrios, encontros e desencontros. A constituição dos saberes necessários à docência no Ensino Superior precisa ser vivenciada na formação em nível

pós-graduação, pois as universidades têm intensa responsabilidade em oferecer à sociedade profissionais que tenham habilidades e competências para utilizar o conhecimento abstrato em situações do concreto real em atuação profissional (IMBERNÓN, 2012).

Sendo assim, é imprescindível que o sujeito mesmo sem formação pedagógica específica e que se dispõe a ser professor, tenha possibilidades de compartilhar momentos de discussões teóricos e experienciais em processos de ensinagem (PIMENTA; ANASTASIOU, 2014); sejam em formação inicial ou continuada. Dessa maneira, o sujeito que pretende atuar como professor, ainda que não tenha formação em curso de licenciatura, precisa ser sensibilizado a compreender as nuances do fazer didático-pedagógico exigido para e na educação.

No Ensino Superior, conforme sinalizam Pimenta e Anastasiou (2014), pode-se utilizar diversas estratégias para condução da ensinagem, tais como: a aula expositiva dialogada, o estudo de texto, o portfólio, a tempestade cerebral, o mapa conceitual, o estudo dirigido, a lista de discussão por meios informatizados, a solução de problemas, o grupo de verbalização e de observação, a dramatização, o seminário, o estudo de caso, o júri simulado, o simpósio, o painel, o fórum, a oficina, o estudo do meio, o ensino com pesquisa, dentre outros. Apesar da utilização das referidas estratégias, não se pode deixar de definir objetivos de ensino para cada uma delas, pois essas não podem operacionalizadas como fins em si mesmas. As autoras apresentam essas possibilidades de estratégias no sentido de viabilizar o processo de ensinagem e superar perspectivas tradicionais de ensino.

No caso da atuação profissional no ensino superior de Ciência da Computação, a formação docente tem a necessidade de refletir sobre o desenvolvimento do *computational thinking* (pensamento e/ou raciocínio computacional) (FERREIRA *et al.*, 2015) atrelado a processos educacionais nos diversos níveis e modalidades da educação. Na discussão sobre *computational thinking*, Wing (2006) refere-se às diversas possibilidades analíticas da Computação. Desse modo, o conceito defendido pela autora relaciona-se ao potencial da Ciência da Computação diante das necessidades de resolução de problemas, abstração, decomposição, automação, simulação, modelação, pensamento recursivo, seqüencial e paralelo. Mostra-se a relevância dos estudos sobre raciocínio computacional significativo à resolução de problemas do cotidiano.

Com intenso potencial de articular ações, discussões e debates das diversas áreas do conhecimento, o desenvolvimento do raciocínio computacional (FERREIRA et al, 2015) deveria ser uma busca constante em processos educacionais orientadores da prática docente em Ciência da Computação. Isso significa inferir que a prática docente necessita de encaminhamentos pedagógicos, didáticos, metodológicos e conceituais na sistematização dos conhecimentos científicos próprios da Ciência da Computação. No entanto, o percurso profissional docente dos que atuam na Ciência da Computação em diversos níveis ou modalidades de ensino não contempla a formação em cursos de licenciatura, com restrita ou total ausência de significativas possibilidades de formação pedagógica que discutam ou os orientem a adequação comportamental, conceitual ou metodológica voltadas ao processo educacional (MASSA, 2015).

Oliveira e Silva (2012, p. 196) concluem que o "[...] bacharel torna-se professor sem nenhum tipo de formação pedagógica e os licenciados, por sua vez, assumem a

profissão com lacunas em seu processo formativo". Portanto, a formação ao Ensino de Ciência da Computação, vivência problemáticas que se baseiam na ausência de qualquer orientação didático-pedagógica mais ampla sobre o ato de ensinar e aprender. Matos (2013) analisa que a Computação ao longo de sua história possui restrita tradição em estudar aspectos identitários de seus profissionais. Por consequência, tem-se exigido formação cada vez mais especializada dos profissionais, e dos que atuam na docência (bacharéis, tecnólogos e/ou licenciados) uma qualificação pedagógica e formação técnica condizente à área da Ciência da Computação (MATOS, 2013).

O percurso profissional docente dos que atuam na Computação geralmente não está articulado à uma formação pedagógica e conhecimentos conceituais das áreas específicas exigidos à docência (SANTOS; SANTOS, 2012). A formação de professores de uma maneira geral precisa mobilizar conhecimentos epistemológicos de determinada área científica, didático-pedagógico e situações de aprendizagem reais que podem consolidar a constituição dos saberes docentes - formação profissional, disciplinares, curriculares e experienciais (TARDIF, 2012).

Massa (2015) trata da formação pedagógica para o ensino de Ciência da Computação e constata que muito se tem pesquisado sobre o tema da Informática na Educação quando se refere à utilização de novas Tecnologias de Informação e Comunicação (TIC). No entanto, alerta para a existência de poucas pesquisas brasileiras sobre a educação relacionada às questões específicas do processo de ensino e de aprendizagem de Ciência da Computação. Mostra-se, portanto, cada vez mais a necessidade dos PPG em Ciência da Computação se preocuparem com a formação docente de seus alunos.

3. Diretrizes para formação docente na pós-graduação em Ciência Computação

No sentido de compreender as possibilidades de formação docente nos PPG em Ciência da Computação se faz necessário analisar as orientações normativas que amparam as ações pedagógicas dos PPG no Brasil. A formação dos sujeitos que tomam posição pela docência mesmo não sendo formados em cursos de licenciatura em computação (a maioria dos profissionais docentes na/da Ciência da Computação), confere às instituições educacionais em nível superior intensa responsabilidade de desenvolver formação continuada para exercício do magistério.

O artigo 66 da Lei de Diretrizes e Bases da Educação Nacional – LDB (BRASIL, 1996) prescreve que a preparação para o exercício do magistério no ensino superior deve ser realizada em nível de pós-graduação, prioritariamente, em programas de mestrado e doutorado.

No ano de 2010, instituiu-se o Plano Nacional de pós-graduação – PNG (2011-2020), oriundo de comissões compostas por representantes de universidades públicas brasileiras com notoriedade nacional para constituir eixos norteadores aos PPG. O sistema de pós-graduação brasileiro foi implantado a partir dos anos 1970, amparado no arcabouço jurídico da Reforma Universitária de 1968. Esse sistema seguiu o modelo de universidade estadunidense, com a missão de promover formação de professores e de pesquisadores (BRASIL, 2010). Desse modo, desde sua origem, os programas deveriam instituir ações que contemplassem a formação para docência e para pesquisa.

O PNG (2011-2020) foi articulado em cinco eixos que deveriam conduzir as ações dos PPG em todo o país, como segue abaixo:

> 1 – a expansão do Sistema Nacional de pós-graduação (SNPG), a primazia da qualidade, a quebra da endogenia e a atenção à redução das assimetrias; 2 – a criação de uma nova agenda nacional de pesquisa e sua associação com a pós-graduação; 3 – o aperfeiçoamento da avaliação e sua expansão para outros segmentos do sistema de C,T&I; 4 – a multi- e a interdisciplinaridade entre as principais características da pós-graduação e importantes temas da pesquisa; 5 – o apoio à educação básica e a outros níveis e modalidades de ensino, especialmente o ensino médio (BRASIL, 2010, p. 15).

Observa-se na delimitação dos eixos, a inexistência de um item que contemple a formação docente ao Ensino Superior, tendo maior ênfase na formação para a pesquisa. Certamente essa ausência da discussão da formação de professores destinada ao Ensino Superior necessária aos PPG, gera fundamentos para possíveis estudos científicos, conforme afirma Massa (2015).

Isto posto, tem-se na pós-graduação brasileira ênfase na formação de desenvolvimento da pesquisa no país. No entanto, como formar bom pesquisadores sem bons professores? Algo bastante contraditório. Portanto, há intensa necessidade de o professor universitário conscientizar-se da importância dos estudos didáticos para melhorar sua relação com os conhecimentos que devem ser sistematizados, sua relação com a disciplina, com os alunos e com sua própria docência (IMBERNÓN, 2012).

Aspecto significativo destacado no PNG é a necessidade de a CAPES (BRASIL, 2010) articular ações educacionais de apoio às diversas modalidades de ensino, particularmente, a educação básica. Na abertura da terceira semana de Avaliação Trienal no ano de 2013, a diretora de Formação de Professores da Educação Básica da CAPES, Carmem Moreira de Castro Neves, falou sobre a urgente necessidade de articulação entre a pós-graduação e a educação básica, segundo a Coordenação de Comunicação Social da Capes (BRASIL, 2014). Ela avalia, ainda, que a pós-graduação no Brasil não pode ter números consideráveis se a qualidade da educação básica for insatisfatória (BRASIL, 2014). Essas notícias evidenciam a preocupação e as orientações normativas que indicam a considerável relevância de articulação sistêmica entre os níveis de educação do nosso país para elevação do padrão de qualidade como um todo do sistema educacional.

O Documento de Área de Ciência da Computação da CAPES do ano quadriênio 2013-2016 (BRASIL, 2016) evidencia a preocupação com a natureza interdisciplinar da área e suas possíveis contribuições ao Ensinos Fundamental e Médio. Nesse caso, os PPG em Ciência da Computação precisam oferecer formação para docência de modo interdisciplinar, bem como subsídios de pesquisa e extensão em outros níveis de ensino para alcance das diretrizes estabelecidas nos documentos oficiais divulgados pela própria CAPES.

Compreende-se que as diretrizes instituídas nos documentos não contempla de maneira incisiva a formação docente, mas a condução de pesquisas científicas. Neste artigo, não se busca desvalorizar a formação para/em pesquisa na pós-graduação, mas refletir sobre a relevância da formação docente no Ensino Superior, particularmente na área de Ciência da Computação. Desse modo, os documentos oficiais são primordiais para a análise dos objetivos dos PPG, pois orientam as diretrizes postas e evidencia a ausência de políticas de formação de professores para compreensão de suas especificidades.

A ausência de formação pedagógica dos professores no Ensino Superior também pode ser relacionada ao tratamento opcional dos componentes curriculares de formação docente, conforme observado nos documentos. Nesse caso, o professor universitário sem formação pedagógica restringe suas possibilidades de reflexão e ação sobre aspectos sociais, psicológicos e pedagógicos do seu trabalho profissional específico ao ensino superior (IMBERNÓN, 2012).

De acordo com a CAPES[1], o Parecer MEC/CESu nº 977/65 (BRASIL, 1965) é o marco conceitual e regulatório da pós-graduação brasileira. O parecer define e confere os objetivos dos cursos de pós-graduação *stricto sensu* no Brasil. De acordo com o parecer, é imperativo que os cursos de pós-graduação *stricto sensu* formem professores universitários, haja vista que o primeiro objetivo dos PPG, segundo o parecer, é "*formar professorado competente que possa atender à expansão quantitativa do nosso ensino superior garantindo, ao mesmo tempo, a elevação dos atuais níveis de qualidade*" (BRASIL, 1965; p. 3). Essa necessidade de formação de professores do nível superior evidenciada pelo parecer, está relacionada à falta de profissionais docentes qualificados. Portanto, conforme o parecer, o instrumento fundamental para a formação dos professores do ensino superior são os cursos de pós-graduação *stricto sensu* (BRASIL, 1965).

Com bases nessas reflexões e regulações oficiais sobre a formação docente na pós-graduação *stricto sensu* em Ciência da Computação, questiona-se sobre *quais são as iniciativas de formação docente nos PPG em Ciência da Computação no Brasil e como essas iniciativas se relacionam com os objetivos desses PPG?* Essa questão norteou a pesquisa apresentada nesse artigo, que apresenta resultados sobre a ações voltadas à formação docente para o ensino superior nos PPG de Ciência da Computação de instituições federais de ensino superior das Regiões Norte e Nordeste.

4. Metodologia

Para identificar as iniciativas de formação docente nos cursos de pós-graduação *stricto sensu* em Ciência Computação foi realizada uma pesquisa documental nos documentos oficiais dos programas de pós-graduação de instituições federais de ensino superior das Regiões Norte e Nordeste do Brasil avaliados pela CAPES na área de Ciência da Computação.

A pesquisa documental consiste na análise de documentos ou dados na busca de informações e de padrões que possam contribuir para a resolução de um determinado

[1] Disponível em: http://capes.gov.br/avaliacao/sobre-a-avaliacao/legislacao-especifica. Acesso em: 31/03/2018

problema (WAZLAWICK, 2010). Portanto, foram investigados os regimentos e matrizes curriculares dos respectivos cursos.

Para guiar a análise foram concebidas sete questões a serem respondidas com base na análise dos documentos:

Q1 - Qual é o objetivo do programa de pós-graduação?

Q2 - O programa de pós-graduação oferece alguma disciplina voltada à formação docente?

Q2.1 - Qual é o título dessa disciplina?

Q2.2 - Essa disciplina é de caráter obrigatório?

Q3 - O programa de pós-graduação oferece alguma atividade curricular voltada à formação docente (ex.: estágio docente)?

Q3.1 - Quais são os tipos de atividades?

Q3.2 - Essas atividades são de caráter obrigatórias?

De acordo com o Documento de Área da Ciência da Computação, publicado pela CAPES em 2016, há 109 cursos de pós-graduação *stricto sensu* na área de Ciência da Computação (BRASIL, 2016). Desse quantitativo, a Região Nordeste possui 29 programas de pós-graduação e a Região Norte possui apenas três[2].

Dentre os 32 programas de pós-graduação das Regiões Norte e Nordeste, foram excluídos os programas de instituições privadas e estaduais de ensino superior e programas/cursos de mestrado profissional, reduzindo a amostra para a análise à quantidade de 15 programas. O Quadro 1. apresenta os nomes dos programas, as instituições federais de ensino superior vinculadas e os níveis dos cursos ofertados (mestrado/doutorado), *corpus* de investigação dessa pesquisa.

Quadro 1. Programas de pós-graduação investigados.

Nome	Instituição de Ensino Superior	Tipo
Informática	Universidade Federal de Alagoas (UFAL)	M[3]
Informática	Universidade Federal do Amazonas (UFAM)	MD[4]
Ciência da Computação	Universidade Federal da Bahia (UFBA)	MD
Ciência da Computação	Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)	M
Ciências da Computação	Universidade Federal do Ceará (UFC)	MD
Ciência da Computação	Universidade Federal do Maranhão (UFMA)	M
Ciência da Computação	Universidade Federal do Pará (UFPA)	MD
Ciência da Computação	Universidade Federal de Campina Grande (UFCG)	MD

[2]Esse dados foram coletados na Plataforma Sucupira. Disponível em: https://sucupira.capes.gov.br/sucupira/ Acesso em: 19/02/2018.
[3] Mestrado.
[4] Mestrado e Doutorado.

Informática	Universidade Federal da Paraíba (UFPB)	M
Ciências da Computação	Universidade Federal de Pernambuco (UFPE)	MD
Informática Aplicada	Universidade Federal Rural de Pernambuco (UFRPE)	M
Ciência da Computação	Universidade Federal do Piauí (UFPI)	M
Ciência da Computação	Universidade Estadual do Rio Grande do Norte & Universidade Federal Rural do Semi-Árido[5] (UERN-UFERSA)	M
Sistemas e Computação	Universidade Federal do Rio Grande do Norte (UFRN)	MD
Ciência da Computação	Universidade Federal de Sergipe (UFS)	M

5. Resultados

Foram identificados e analisados os regimentos e as matrizes curriculares dos 15 PPG da área de Ciência da Computação de instituições federais de ensino superior das Regiões Norte e Nordeste do Brasil.

Para favorecer a compreensão dos objetivos dos PPG foi criada a *tagcloud* apresentada na Figura 2. A *tagcloud* foi composta pelas palavras que compõem os objetivos dos PPG *corpus* desta pesquisa. Acredita-se que a evidência das palavras em conjunto podem apontar a associação de termos e fornecer os interesses e experiências das palavras investigadas; impressão geral do conjunto de conteúdo (Rivadeneira *et al.*, 2007).

Figura 2. *Tagcloud* de palavras que compõem os objetivos dos PPG em Ciência da Computação das Regiões Norte e Nordeste.

Nessa *tagcloud* as palavras "computação", "ciência" e "área" estão em evidência com relação às demais, isso acontece porque houve 16 ocorrências da palavra "computação" e 14 das palavras "ciência" e "área". Todavia, o que interessa a esta

[5] O PPG em Ciência da Computação da UERN-UFERSA é multi-institucional com associação ampla entre as duas instituições e por uma delas ser federal, o programa foi incluído (*cf.* https://ppgcc.ufersa.edu.br/apresentacao/).

pesquisa é o destaque das palavras relacionadas à atividade docente (*i.e.* ensino, docentes, docência, professores). Dentre as relacionadas à formação docente, a palavra "ensino" é a de maior destaque. Todavia, várias outras palavras estão mais destacadas que às palavras relacionadas a atividade docente e isso expressa a importância dada pelos PPG, por exemplo, à "pesquisa" quando comparada ao "ensino".

A palavra "pesquisa" está mais evidente que a palavra "ensino", isso acontece porque somente nove dos 15 PPG tratam sobre a formação de professores em seus objetivos. Portanto, é possível reconhecer que os objetivos dos PPG estão mais alinhados à formação e desenvolvimento de pesquisadores e/ou pesquisas na área de Ciência da Computação, deixando de lado o primeiro objetivo da pós-graduação apresentado pelo Parecer MEC/CESu nº 977/65, a formação do professor do ensino superior (BRASIL, 1965).

O Quadro 2 apresenta a sumarização dos dados coletados quanto à formação docente dos pós-graduandos na área de Ciência da Computação. Por meio da análise documental foi possível identificar que somente três PPG oferecem disciplinas relacionadas à docência no ensino superior: UFRN, UFCG e IFCE.

Quadro 2. Sumarização dos dados coletados[6].

Instituição	Q2	Q2.1	Q2.2	Q3	Q3.1	Q3.2
UFAL	Não	--	--	Sim	Estágio Docência	Somente para bolsistas CAPES
UFAM	Não	--	--	Sim	Estágio Docência	Para todos alunos
UFBA	Não	--	--	Sim	Estágio Docente Supervisionado	Para todos alunos
IFCE	Sim	*Docência no Ensino Superior*	Não	Não	--	--
UFC	Não	--	--	Não	--	--
UFMA	Não	--	--	Não	--	--
UFPA	Não	--	--	Sim	Estádio de Docência	Para mestrandos bolsistas e para todos doutorandos
UFCG	Sim	*Metodologia do Ensino Superior*	Não	Sim	Estágio Docência	Não
UFPB	Não	--	--	Sim	Estágio Docência	Não
UFPE	Não	--	--	Não	--	--
UFRPE	Não	--	--	Sim	Estágio de Docência	Não
UFPI	Não	--	--	Sim	Estágio Docência	Para todos alunos
UERN-UFERSA	Não	--	--	Não	--	--

[6] Cada coluna faz referência à respectiva questão apresentada na metodologia.

UFRN	Sim	*Iniciação a Docência no Ensino Superior*	Somente para bolsistas do Programa de Bolsas REUNI	Sim	Estágio Docência em Computação	Somente para bolsistas Programa de Bolsas REUNI
UFS	Não	--	--	Sim	Estágio Docência	Para bolsistas

Na UFRN a disciplina oferecida é nominada de *Introdução a Docência no Ensino Superior*. A disciplina é de natureza obrigatória somente aos alunos bolsistas do Programa de Bolsas de Assistência ao Ensino do REUNI[7]. Por sua vez, na UFCG, a disciplina correlata é chamada de *Metodologia do Ensino Superior* e tem caráter optativo. No IFCE, a disciplina oferecida possui o nome *Docência no Ensino Superior* e também é de natureza optativa.

Apesar de muitos PPG não possuírem disciplinas relacionadas à formação docente em suas estruturas curriculares, dez PPG instituíram a atividade de *estágio docente*[8]. Os PPG da UFAM e UFBA, que possuem cursos de mestrado e doutorado, estabeleceram o estágio docente como atividade obrigatória para ambos os cursos. A UFPA[9] institui o estágio docente como atividade obrigatória apenas para o doutorado, enquanto a UFPI apenas para o curso de mestrado.

Por sua vez, os PGG da UFAL, da UFS, ambos apenas com curso de mestrado, e o da UFRN tornaram obrigatório o estágio docente, mas apenas para os seus alunos bolsistas. O PPG da UFS ainda determinou que o aluno que comprovar o exercício de atividades de docência no ensino superior realizadas na própria UFS receberá créditos e, consequentemente, terá a quantidade de disciplinas a serem cursadas diminuída. Não foram encontrados dados sobre disciplinas ou atividades relacionadas à formação docente no ensino superior nos regulamentos e matrizes curriculares dos PPG da UFC, da UFMA, da UERN-UFERSA e da UFPE.

Dentre os programas estudados, o PPG da UFPE é o único com conceito 7 (excelência internacional) na última avaliação da CAPES (Quadriênio 2013-2016). Entretanto, esse programa não apresenta qualquer ação/atividade voltada à formação docente do ensino superior em seu regimento, nem na sua matriz curricular. Isso, de certo modo, reflete a desatenção dada pelo Comitê de Área da Ciência da Computação (BRASIL, 2016) à formação de professor para o ensino superior, conforme o objetivo do Parecer MEC/CESu nº 977/65 (BRASIL, 1965).

Como afirma D'Ávila (2013), a atividade docente deve ser constituída em uma prática social ampla, com a combinação de conhecimentos, habilidades e atitudes, expectativas e percepções de mundo relacionadas a diversas histórias de vida dos professores. Nesse caso, os processos de ensinagem no Ensino Superior exigem reflexão sobre a prática social e coletiva diante da construção de um partilhar das próprias incertezas e dificuldades do ato de ensinar (PIMENTA e ANASTASIOU, 2014). Certamente, as restritas discussões promovidas no sentido de oferecer formação didático-pedagógica no Ensino Superior nos PPG de Ciência da Computação pouco

[7] Programa de Bolsas de Assistência ao Ensino do REUNI tem por objetivo elevar a qualidade do ensino superior público prestando subsídio aos cursos de graduação por meio dos cursos de pós-graduação.
[8] O nome da atividade difere entre algumas instituições.
[9] No PPG da UFPA o estágio docente é obrigatório para bolsistas de mestrado.

contribuirão para os professores compreenderem seu papel de agente social em sua atuação profissional (VEIGA, 2002). Por isso, as reflexões sobre a prática docente poderiam ocorrer em formações no transcurso dos mestrados e doutorados, de modo que principalmente os professores universitários bacharéis e tecnólogos (sem formação em licenciatura) tivessem melhores condições de refletir sobre suas práticas docentes.

6. Considerações Finais

A ausência de formação docente nos PPG em Ciência da Computação, nas regiões Norte e Nordeste do Brasil, tem repercussão não somente no Ensino Superior, mas nas demais modalidades de ensino. Sabendo que os profissionais formados em nível de pós-graduação poderão atuar e articular pesquisas em diversas modalidade de ensino, mesmo com intenso foco no ensino superior.

O intenso foco na pesquisa tem deixado a formação à docência em muitos casos na dimensão da opcionalidade, quando pelo menos isso é permitido. É contraditório o professor atuante no ensino superior apenas ter sua formação de pós-graduação direcionada à pesquisa. Mesmo porque a atuação docente exige a constituição de diversos saberes necessários à docência. O exercício profissional docente necessita ter movimentos que promovam processos de ação-reflexão-ação (RAMALHO; NUÑEZ; GAUTHIER, 2003) sobre o processo de ensinagem no Ensino Superior.

A pesquisa de Massa (2015), diante dos restritos estudos nessa área, mostra a necessidade de busca por formações de docência relacionadas ao ensino superior de Ciência da Computação. Precisa-se promover, sejam em disciplinas ou estudos, as reflexões necessárias sobre a atividade docente para ensinagem no ensino superior de modo mais amplo (PIMENTA e ANASTASIOU, 2014). E, para além disso, atender aos objetivos da pós-graduação *stricto sensu* no Brasil, conforme Parecer MEC/CESu nº 977/65 (BRASIL, 1965).

Como enfatizado, na Figura 2 fica evidente a palavra “pesquisa” em detrimento do “ensino”; tal constatação é um dos indicativos das limitadas atuações de natureza didático-pedagógica destinadas ao Ensino Superior. Ainda que a formação docente necessária ao Ensino Superior exija uma diversidade de estratégias didático-pedagógicas na condução da ensinagem, os PPG em Ciência da Computação apresentam restritas orientações de formação à docência; sendo, portanto, necessária a reflexão sobre as possibilidades de viabilizar a formação docente para atuação no Ensino Superior. Desse modo, pode dizer que ainda é preciso constituir os saberes docentes nas formações de mestrado e doutorado em Ciência da Computação.

Como trabalho futuro será ampliado o *corpus* da análise para os demais PPG em Ciência da Computação do Brasil, haja vista que esse estudo se limitou aos regimentos e às matrizes curriculares dos 15 PPG de instituições federais de ensino superior das Regiões Norte e Nordeste do Brasil. Além disso, serão realizadas entrevistas com mestrandos, doutorandos e recém[10] mestres e doutores da área de Ciência da Computação para investigar se e como ocorre(u) sua formação docente na pós-graduação.

[10] Serão considerados recém mestres e doutores os diplomados nos último cinco anos a partir de 2018.

Agradecimentos

Agradecemos ao Grupo de Pesquisa e Extensão em Informática, Educação e Sociedade - Onda Digital (UFBA/CNPq) e à CAPES pelo financiamento parcial desta pesquisa.

Referências

Anastasiou, L. C. (2007) Ensinar, aprender, apreender e processos de ensinagem. In: ANASTASIOU, L. C.; ALVES, L. P. (Orgs.). *Processos de ensinagem na universidade: pressupostos para as estratégias de trabalho em aula*. 7. ed. Joinville: Univille, 2007. p. 15-43.

Brasil. (1965) Parecer MEC/CESu nº 977/65. *Definição dos Cursos de Pós-Graduação*. 1965 Disponível em: http://portal.mec.gov.br/cne/arquivos/pdf/2007/parecer%20cfe%20977-1965.pdf. Acesso em: 13 mar. 2018.

_____. (1996) Senado Federal. *Lei de Diretrizes e Bases da Educação Nacional: nº 9394/96*. Brasília.

_____. (2007) Lei n.º. 11. 502, de 11 de julho de 2007. Modifica as competências e a estrutura organizacional da fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, [...] e autoriza a concessão de bolsas de estudo e de pesquisa a participantes do programa de formação inicial e continuada de professores para a educação básica. *Diário Oficial da União*, Brasília, DF, n. 133, 12 jul. 2007.

_____. (2010) *Plano Nacional de Pós-graduação (PNPG) 2011-2020.* Disponível em: https://www.capes.gov.br/images/stories/download/PNPG_Miolo_V2.pdf. Acesso em: 07 jan. 2016.

_____. (2014) Coordenação de Comunicação Social da Capes. *Articulação entre pós-graduação e educação básica foi abordada na abertura da terceira semana de Avaliação Trienal.* Disponível em: http://www.capes.gov.br/36-noticias/6581-articulacao-entre-pos-graduacao-e-educacao-basica-foi-abordada-na-abertura-da-terceira-semana-de-avaliacao-trienal. Acesso em: 07 jan. 2016.

_____. (2015) *Portaria nº. 086, de 03 de julho de 2015. Aprovar o Regulamento do Programa Nacional de Pós-Doutorado – PNPD.* Disponível em: https://www.capes.gov.br/images/stories/download/legislacao/Portaria_86_2013_Regulamento_PNPD.pdf. Acesso em: 07 jan. 2016.

_____. (2016) *Avaliação Quadrienal da CAPES. Documento de área da Ciência da Computação.* Disponível em: http://www.capes.gov.br/images/stories/download/avaliacaotrienal/Docs_de_area/Ci%C3%AAncia_da_Computa%C3%A7%C3%A3o_doc_area_e_comiss%C3%A3o_att08deoutubro.pdf. Acesso em: 07 jan. 2018.

D'ávila, C. M. (2008) Formação Docente na Contemporaneidade: Limite e Desafios. *Revista da FAEEBA: Educação e Contemporaneidade*, Salvador, v. 17, n. 30, jul/dez. p. 103 - 112.

Ferreira, A. C. C.; Melhor, A.; Barreto, J. S.; Paiva, L. F.; Matos, E. (2015) Experiência prática interdisciplinar do raciocínio computacional em atividades de computação desplugada na Educação Básica. In: *Anais do XXI Workshop de Informática na Escola*. Maceió/AL. p. 256 - 265.

Freire, P. (1996) *Pedagogia da Autonomia: saberes Necessários à Prática Educativa.* São Paulo: Paz e Terra.

Imbernón, F. (2004) *Formação docente e profissional: formar-se para a mudança e a incerteza*. 2º ed. São Paulo: Cortez.

_____. (2012) *Inovar o ensino e a aprendizagem na universidade*. Trad. Silvana Cobucci Leite. São Paulo: Cortez.

Massa, M. S. (2015) A formação didático-pedagógica do docente da área de computação: um estudo de caso em uma Universidade Brasileira. In: *Anais do XXXV Congresso da Sociedade Brasileira de Computação*. Porto Alegre: SBC - Sociedade Brasileira de Computação, 2015.

Matos, E. (2013) *Dialética da Interação Humano-Computador: tratamento didático do diálogo midiatizado*. Tese (Doutorado em Educação). Universidade de São Paulo, São Paulo/SP.

_____. Identidade profissional docente e o papel da interdisciplinaridade no currículo de licenciatura em computação. *Revista Espaço Acadêmico*, Maringá, PR, v. 13, n. 148, set. 2013. p. 26 - 34.

Oliveira, V, S.; Silva, R. F. (2012) Ser bacharel e professor: dilemas na formação de docentes para a Educação Profissional e Ensino Superior. *Revista HOLOS*, ano 28, v. 2. p. 193 - 205.

Pimenta, S. G.; Anastasiou, L. G. C. (2014) *Docência no Ensino Superior*. 5 ed. São Paulo, Cortez.

Pimenta, S. G.; Ghedin, E. (2002) *Professor Reflexivo no Brasil*: Gênese e Crítica de um Conceito. São Paulo: Cortez.

Ramalho, B. L.; Nuñez, I. B.; Gauthier, C. (2003) *Formar o professor, profissionalizar o ensino*: perspectivas e desafios. Porto Alegre: Sulina.

Reche; B. D.; Vasconcellos, M. M. M. (2014) A construção da carreira docente por bacharéis: considerações iniciais. In: *Anais da X ANPED SUL*, Florianópolis. p. 1 - 20.

Rivadeneira, A. W.; Gruen, D. M.; Muller, M. J.; Millen, D. R. (2007). Getting our head in the clouds: Toward evaluation studies of tagclouds. In: *Proceedings of 25th SIGCHI Conference on Human Factors in Computing Systems*. p. 995 - 998.

Santos, E.; Santos, R. (2012) A formação pedagógica é necessária ao docente de computação? - análise dos currículos de referência e das diretrizes curriculares dos cursos da área de computação. In: *Anais do XVI Encontro Nacional de Didática e Práticas de Ensino – ENDIPE*. Unicamp/Campinas. p. 895 - 908.

Silva, R. S.; Matos, E. S.; Massa, M. (2018) O desenvolvimento da identidade docente por professores de Computação não licenciados atuantes na Educação Profissional de Nível Médio. XXVI Workshop sobre Educação em Computação. In: *Anais do XXXVIII Congresso da Sociedade Brasileira de Computação*. Natal/RN. s/p.

Tardif, M. (2012) *Saberes docentes e formação profissional*. 14 ed. Petrópolis, RJ: Vozes.

Vianna, W. B.; Ensslin, L.; Giffhorn; E. (2011) A integração sistêmica entre pós-graduação e educação básica no Brasil: contribuição teórica para um "estado da arte". *Ensaio: aval. pol. públ. educ.*, Rio de Janeiro, v. 19, n. 71, abr./jun. p. 327-344.

Veiga, I. P. A. (2002) Professor: tecnólogo do ensino ou agente social? In: VEIGA, I. P. A.; AMARAL; A. L. (Orgs.) *Formação de professores: políticas e debates*. Campinas, SP: Papirus. p. 65 - 96.

Wazlawick, R. S. (2010) Uma Reflexão sobre a Pesquisa em Ciência da Computação à Luz da Classificação das Ciências e do Método Científico. *Revista de Sistemas de Informação da FSMA*, n. 6. p. 3 - 10 .

Wing, J. M. (2006) Computational Thinking. *Communications of the ACM*. March, v. 49, n.. 13. p. 33 - 35.

Programação para Administração de Redes de Computadores – Uma Experiência com Estudantes de Computação

Silmar Antonio Buchner de Oliveira[1,2], Andréa Pereira Mendonça[1]

[1]Mestrado Profissional em Ensino Tecnológico. Instituto Federal do Amazonas (IFAM) Manaus – AM – Brasil.
[2]Instituto Federal de Rondônia (IFRO). Porto Velho – RO – Brasil.

silmar.oliveira@ifro.edu.br, andrea.mendonca@ifam.edu.br

Abstract. *In this paper we report an experience on conception and execution of a programming course on computer network administration. This course is aimed at students of undergraduate technology courses in Computing related areas. Organized according to a flipped classroom approach, the course focused on solving practical problems by using the Python programming language. The results obtained evidence student's progress in automatising solutions for network equipment and user actions monitoring, log analysis, data synchronizing as well as automatic alerts issuing. The teaching materials and the dynamics of the course will be integrated into a guideline to aid other teachers of computer network affined courses.*

Resumo. *Neste artigo relatamos uma experiência de criação e execução de um curso de programação para administração de redes de computadores, voltado para estudantes de Cursos Superiores de Tecnologia na área de Computação. Implementado no formato de sala de aula invertida, o curso focou na resolução de problemas práticos utilizando a linguagem Python. Os resultados evidenciaram progresso dos estudantes na automatização de soluções para monitoramento de equipamentos de redes e ações de usuários, análise de logs, sincronismos de dados e emissão automática de mensagens de alertas. O material de ensino e dinâmica do curso serão integrados em um guideline para auxiliar outros professores de redes de computadores.*

1. Introdução

Os currículos dos Cursos Superiores em Computação no Brasil, independente da categoria (bacharelado, tecnologia ou licenciatura), possuem em comum a presença de duas disciplinas: (i) **programação** que integra a área de formação básica; e (ii) **redes de computadores** que integra a área de formação tecnológica (BRASIL, 2012).

Tratando especificamente sobre os Cursos Superiores de Tecnologia (CST), observamos que a disciplina de redes de computadores é lecionada priorizando aspectos teóricos, tais como: topologia de redes; protocolos de comunicação, em especial, o modelo de referência OSI (*Open System Interconnection*) e a pilha de protocolos TCP/IP (*Transfer Control Protocol/Internet Protocol*); tipos de redes e aspectos básicos de interconexão de redes (WANG; BLUM; MCCOEY, 2014). Por outro lado, a disciplina de programação é ensinada dentro de uma perspectiva prática, voltada para a resolução de problemas com a adoção de uma linguagem de programação.

Ocorre que, ao ingressarem na vida profissional, os estudantes dos Cursos Superiores de Tecnologia são demandados para a resolução de problemas práticos. No caso de atuação na área de redes de computadores, são comuns demandas voltadas para administração de redes de computadores, as quais envolvem automatização de procedimentos para monitoramento, controle e gerenciamento de serviços, como execução de *backups*, sincronismo de dados, verificação de consumo de recursos, envio de mensagens de alerta via *e-mail*, etc. Estas demandas, por sua vez, não podem ser atendidas com base em uma formação de redes pautada apenas em aspectos teóricos e dissociada dos conhecimentos de programação.

Dada esta lacuna na formação, relatamos neste artigo a proposta e execução de um curso de programação para administração de redes de computadores, voltado para estudantes de CST. O curso focou na resolução de problemas práticos, adotou Python como linguagem de programação, foi administrado na modalidade de ensino híbrido, mais especificamente, no modelo de sala de aula invertida, combinando estudo autônomo e presencial, os quais foram conduzidos por meio de roteiros de aprendizagem (RA).

2. Ensino de redes de computadores nos CSTs

Os Cursos Superiores de Tecnologia (CSTs) possuem regulamentação específica e uma delas é o Catálogo Nacional de Cursos Superiores de Tecnologia, disponibilizado pela Secretaria de Educação Profissional e Tecnológica do Ministério da Educação (SETEC/MEC). Os Catálogos são organizados por eixos, sendo o eixo Informação e Comunicação aquele que integra os cursos relacionados à área de Computação/Informática, como CST em Análise e Desenvolvimento de Sistemas, CST em Sistemas para Internet, CST em Redes de Computadores, CST em Jogos digitais, entre outros (BRASIL, 2016).

Por sua característica direcionada à prática profissional, os CSTs têm duração mais curta (entre 2 e 3 anos), em comparação com os cursos de bacharelado. Seus egressos possuem um perfil profissional dotado de compreensão sobre o processo produtivo, fundado no saber tecnológico e no ato de fazer, com autonomia para tomada de decisões profissionais, indo além do simples domínio operacional de técnicas de trabalho. Geralmente, os CST são oferecidos pelos Institutos Federais de Educação, pertencentes à rede Federal de Educação Profissional, Científica e Tecnológica, criada em 2008. Há também a oferta de CST em universidades públicas e Instituições de Ensino Superior particulares (BRASIL, 2006; BRASIL, 2002).

A fim de caracterizar o perfil das disciplinas de redes de computadores e de programação em CST, realizamos uma pesquisa *ad hoc* em cursos do eixo Informação e Comunicação, oferecidos pelos Institutos Federais de Educação. Como resultado desta pesquisa, identificamos uma relação de 60 (sessenta) cursos, distribuídos em 40 (quarenta) instituições, conforme informações obtidas no sistema e-Mec[1] do Ministério da Educação.

A partir da identificação dos cursos, analisamos as ementas das disciplinas iniciais de redes de computadores e identificamos que as mesmas eram essencialmente teóricas, baseadas em alguma edição das obras de Andrew S. Tanenbaum (Redes de

[1] http://emec.mec.gov.br

Computadores) e de James F. Kurose (Redes de Computadores e a Internet), centradas no modelo OSI e na pilha de protocolos TCP/IP.

A qualidade das obras é indiscutível. Contudo, os resultados da pesquisa *ad hoc* apontam um distanciamento entre teoria e prática na forma como a disciplina é ensinada, no tocante à resolução de problemas de redes de computadores, quando considerado o contexto profissional no qual atuam os egressos dos CST.

Na pesquisa, identificamos 9 (nove) CST que possuíam em sua matriz curricular uma disciplina cuja denominação (por exemplo, Programação de Scripts e Programação Aplicada à Gerência de Redes de Computadores) indicava a junção de redes de computadores e programação. Entretanto, não identificamos informações sobre como as mesmas são aplicadas, de forma que se outro professor quiser repetir a experiência, terá dificuldades, pois não encontrará planejamento, metodologias detalhadas ou problemas práticos abordados nestas disciplinas.

Destacamos ainda que, nos CSTs do Eixo Informação e Comunicação há pelo menos uma disciplina de programação no currículo, ensinada no primeiro ano do curso, antes da disciplina de redes de computadores. Tal organização do currículo poderia resultar no aproveitamento dos conhecimentos adquiridos em programação, dentro da disciplina de redes de computadores, possibilitando aos alunos a resolução de problemas práticos com o auxílio de programação.

Corroborando com os resultados de nossa pesquisa *ad hoc*, Chen (2014) aponta que os cursos de redes de computadores possuem muito conteúdo teórico, abstrato e de difícil entendimento, muitas vezes prejudicando o interesse na aprendizagem e resultando em uma capacidade prática deficitária. O autor sugere como solução a adoção de atividades práticas para a resolução de problemas reais. Kurose et al. (2002) também fazem críticas ao ensino essencialmente teórico nos cursos de redes de computadores e citam a importância de utilizar laboratórios com aplicações práticas. Inclusive, levantam perspectivas sobre quando e quanto programar em redes de computadores. Enfatizam ainda que os estudantes de redes de computadores que aprendem a codificar programas podem ter mais facilidade para internalizar e entender conceitos avançados, além de desenvolver um conhecimento "comercializável". Wang, Blum e McCoey (2014) ao discutirem o ensino de redes de computadores para iniciantes, sugerem que, além da introdução geral e tradicional, devem proporcionar atividades práticas que envolvam programação para resolução de problemas.

Frente a esta lacuna de formação, desenvolvemos a proposta de um curso de programação para administração de redes de computadores, focado na resolução de problemas práticos, que adota a linguagem de programação Python e organiza as práticas de ensino-aprendizagem de acordo com o ensino híbrido, mais especificamente, o modelo de sala de aula invertida, como será detalhado na seção seguinte.

3. Metodologia do curso

O curso está estruturado em dois módulos: o primeiro, abrangendo assuntos referentes ao monitoramento de redes e de ações de usuários; o segundo, integrando assuntos sobre a preservação das informações e comunicação de eventos via e-mail e Telegram. Cada assunto está associado a uma temática, cujo desdobramento se dá pela integração de conteúdos sobre redes, sistemas operacionais e programação, conforme pode ser

observado no Quadro 1. Além disso, para cada assunto, há um conjunto de problemas propostos, nos quais os estudantes devem integrar teoria e prática, propondo soluções automatizadas com uso de programação. Os assuntos abordados em cada temática são cumulativos e suas complexidades evoluem conforme o andamento do curso.

Quadro 1. Módulos, temas, assuntos e conteúdos.

Módulo 1 - Monitoramento de redes de computadores
Tema 01: Identificando a movimentação da vizinhança
Assunto: Monitoramento de disponibilidade de equipamentos e serviços em rede. **Conteúdos principais**: Redes: Comando "ping"; Protocolo SNMP. Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados. S.O. Linux: Estruturas de arquivos e diretórios; - Monitoramento de carga e de estados de componentes. **Problemas propostos**: Detecções de problemas de lentidão ou perda de acesso, através de monitoramentos via SNMP e "ping", relacionados ao estado de máquinas e componentes.
Tema 02: Cuidando do *Habitat*
Assunto: Monitoramento de estado de discos e partições, em rede. **Conteúdos principais**: Redes: Protocolo NFS; Protocolo SNMP. Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados. S.O. Linux: Compartilhamento de dados; Estruturas de arquivos e diretórios; **Problemas propostos**: Monitoramentos remotos de discos e partições como prevenção de estouros de espaços de armazenamento.
Tema 03: De olho nas intenções
Assunto: Monitoramento de ações de usuários. **Conteúdos principais**: Redes: Protocolo SSH, Acesso remoto; Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados; Operações sobre arquivos; Operações com *strings* e datas; Laços; Decisão. S.O. Linux: Editores de textos; Acesso remoto; Arquivos de configuração; Hierarquia de usuários; Históricos de ações; Data e hora; **Problemas propostos**: Monitoramentos de ações de usuários através de verificação automatizada de *logs* do sistema operacional.
Módulo 2 - Preservação de informações e comunicação de eventos
Tema 04: Construir é difícil. Reconstruir pode ser impossível
Assunto: Compactação de dados, cópias de segurança em rede e registros de atividades (*logs*). **Conteúdos principais**: Redes: *Backup*; Compartilhamento de arquivos; Protocolo NTP; Protocolo NFS. Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados; Operações com *strings* e datas; Laços; Decisão. S.O. Linux: Empacotamento e compactação de dados; Agendamento de tarefas; Sincronismo de data e hora; Arquivos de *logs*. **Problemas propostos**: Agendamentos de execuções de *backup* em rede, com verificação de etapas de execução e gravação em arquivos de *log*.
Tema 05: Um é pouco. Então tenha dois (ou mais!)
Assunto: Sincronismo de dados em rede. **Conteúdos principais**: Redes: *Backup*; Políticas de *backup*; Sincronismo de dados remotos; Compartilhamento de arquivos; Protocolo SSH; Protocolos NFS. Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados; Operações com *strings* e datas; Decisão. S.O. Linux: Sincronismo de dados; Agendamento de tarefas; Arquivos de *logs*. **Problemas propostos**: Agendamentos de execuções de sincronismo de dados em rede, com verificação de etapas de execução e gravação em arquivos de *log*.
Tema 06: Seja o primeiro a saber
Assunto: Automatização de mensagens de eventos e de alertas via *e-mail* e mensageria instantânea (Telegram). **Conteúdos principais**: Redes: Serviços de e-mail; Protocolo SMTP; Protocolos SSL/TLS e StartTLS. Programação: Bibliotecas e métodos; variáveis; entrada e saída de dados; Envio de *e-mails*; Operações com *strings*. S.O. Linux: Envios de *e-mails*; Envio de mensageria instantânea; Sincronismo de dados; Agendamento de tarefas; Configuração de serviços. **Problemas propostos**: Automatizações de procedimentos com verificações de etapas de execuções e, em função dos resultados, envio de avisos e alertas via e-mail e/ou Telegram.

Fonte: Autoria Própria.

Para a definição dos conteúdos, realizamos uma pesquisa em livros nacionais e internacionais que tratam sobre redes de computadores, conforme apresentado no Quadro 2. Com base nesta pesquisa, selecionamos os assuntos mais frequentes e identificamos também os tipos de problemas associados aos conteúdos.

Quadro 2. Relação livro x conteúdo

Legenda dos conteúdos: 1-Monitoramento de Redes; 2-Compartilhamento de dados; 3-Envio de *e-mails*/relatório; 4-Gerenciamento de sistemas de arquivos; 5-Gerenciamento de usuários/acesso; 6-*Login*; 7-Monitoramento de sistemas; 8-Gerenciamento de Arquivos e Diretórios; 9-*Backup*; 10-SSH/acesso remoto; 11-Análise de *logs*; 12-Compressão de dados; 13-Autenticação segura; 14-Expressões regulares; 15-Manipulação de *strings*.

	Relação Livro x Conteúdo		Conteúdo														
	Livro	**Ano**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Administração de redes com scripts	2007	■		■	■	■	■	■		■			■			
2	Automating Linux and Unix System Administration	2009		■							■	■					
3	Automating System Administration with Perl	2009	■		■	■	■	■	■	■			■				
4	Automating Windows Administration	2004			■		■	■		■			■	■		■	
5	Automating Windows Server 2008 Administration with Windows PowerShell	2008		■		■	■	■		■							
6	Core Python Applications Programming	2012		■	■											■	
7	Essential System Administration-Tools and Techniches for Linux and Unix	2002	■		■	■	■		■		■	■			■		
8	Foundations of Python Network Programming	2014	■	■	■						■				■		
9	Learning Python Network Programming	2015	■	■	■			■			■						
10	Linux Command Line and Shell Scripting Bible	2015				■	■		■	■							
11	Linux System Administration Recipes	2009	■	■		■		■			■	■			■		■
12	Pro Python System Administration	2014	■						■								
13	Python for Unix and Linux System Administration	2008	■	■	■	■		■		■		■	■	■		■	■
14	Shell Scripting Recipes	2015				■				■							■
15	Unix and Linux System Administration Handbook	2011					■		■		■						
16	Wicked Cool Shell Scrips – 101 scripts	2004	■	■			■		■	■	■		■	■			
	Total de ocorrências		9	8	8	8	8	7	7	7	6	6	4	4	3	3	3

Fonte: Autoria Própria.

A dinâmica para administração dos conteúdos e resolução dos problemas foi organizada obedecendo ao modelo de ensino híbrido de sala de aula invertida, criado por Bergmann e Sams (2012). De acordo com este modelo, uma parte da aprendizagem é feita de forma autônoma pelo estudante, fora da sala de aula, utilizando recursos tecnológicos. A outra parte é realizada em sala de aula, com a presença de um professor. Assim, a carga horária do curso de 60 (sessenta) horas, foi distribuída em 40 (quarenta) horas para estudos presenciais e 20 (vinte) horas para estudos autônomos.

Para guiar o aluno tanto no estudo presencial quanto no autônomo, baseados nos conteúdos das obras elencadas no Quadro 2, definimos os Roteiros de Aprendizagem (RA) que são instrumentos que conduzem o aluno por um itinerário de estudo a fim de que ele atinja os resultados pretendidos da aprendizagem. Para cumprir os assuntos descritos no Quadro 1, doze (12) roteiros foram elaborados, sendo 6 (seis) roteiros de aprendizagem para estudo autônomo[2] e 6 (seis) roteiros para estudo presencial[3]. Os RAs para estudo autônomo e presencial se assemelham na sua estrutura geral, a diferença reside na seção de tarefas, conforme pode ser verificado no Quadro 3.

[2] Um exemplo de RA autônomo (RA Autônomo 01) está disponível em https://goo.gl/t6wTj5.
[3] Um exemplo de RA presencial (RA Presencial 01) está disponível em https://goo.gl/qeTGDu.

Para auxiliar na realização dos estudos autônomos, foram criadas 27 vídeoaulas, com tempo médio de 7 minutos cada, num total de 3 horas e 28 minutos, postadas no Youtube. Também foram utilizados vídeos de terceiros como os que tratavam sobre Python e redes, tais como: Python para Zumbis[4], Curso em vídeo[5] e Nic.br[6].

Quadro 3. Estrutura geral dos Roteiros de Aprendizagem (RAs).

Cabeçalho
Identifica o módulo corrente, o número do roteiro, o tema abordado, o tipo de estudo (presencial ou autônomo) e a carga horária prevista.
Descrição geral
Apresenta orientações sobre como seguir o RA.
Resultados pretendidos da aprendizagem
Descreve os objetivos da aprendizagem a serem atingidos pelos estudantes.
Conteúdos envolvidos
Apresenta os conteúdos abordados, referentes a redes de computadores, programação e sistemas operacionais (Linux).
Tarefas
Nos **estudos autônomos**, orientam a evolução do estudante sobre os conteúdos preparatórios para a resolução dos problemas que serão apresentados nos estudos presenciais. Tais tarefas são dotadas de questões cujas resoluções são guiadas por vídeos, tutoriais *online* e *links* de páginas *web* para a aplicação prática em um laboratório virtual, criado especificamente para o curso. Os **estudos presenciais** são compostos de dinâmicas para resolução de problemas, inspiradas em Bender (2014) e em Biggs e Tang (2011), divididas em: Aquecimento - Durante os primeiros 30 minutos de aula, os estudantes expõem suas dúvidas sobre o estudo autônomo, em uma discussão mediada pelo professor. Apresentação do contexto - É apresentado um contexto envolvendo um problema com características reais de redes de computadores. Discussão - São dispostas questões que instiguem o estudante a tentar entender melhor a solução do problema e fomentem uma discussão sobre formas de solucioná-lo. Resolução - O estudante descreve como será a solução para o problema e quais recursos de programação utilizará. Depois constrói um programa em Python consonante com a descrição e o executa no laboratório virtual.

Fonte: Autoria Própria.

O material do curso foi disponibilizado de forma virtual no Google Classroom[7]. Outros canais virtuais também foram utilizados, como os que serviam para elucidação de dúvidas sobre o estudo autônomo, como e-mails e mensagens Whatsapp. De forma complementar, uma vez por semana havia duas horas e meia dedicadas ao atendimento aos alunos, cuja presença era facultativa. Estes atendimentos tinham por objetivo dirimir dúvidas e explicar conceitos, antes de cada estudo presencial. Também foram disponibilizados horários extras no laboratório para que os estudantes pudessem realizar

[4] https://www.pycursos.com/python-para-zumbis/
[5] http://www.cursoemvideo.com/course/curso-python-3/
[6] http://www.nic.br/videos/
[7] Google Classroom (no Brasil, Google Sala de Aula) é uma plataforma web da Google que reúne um conjunto de serviços (e-mail, drive, gerenciador de tarefas, entre outros) para auxiliar e promover atividades educacionais.

estudos autônomos e sanar dúvidas sobre os estudos presenciais, dispondo de computadores e Internet na própria instituição de ensino.

Três avaliações foram planejadas para o curso: (i) avaliação diagnóstica, realizada antes de iniciar o curso e que tinha por objetivo identificar os conhecimentos prévios dos alunos; (ii) avaliação intermediária, planejada para o final do primeiro módulo; e, (iii) avaliação final, a ser realizada após o término do segundo módulo. Cada avaliação possuía estrutura semelhante aos roteiros de aprendizagem para estudo presencial, com tarefas direcionadas à resolução de problemas de redes com uso de programação.

A primeira experiência de aplicação do curso foi realizada no período de novembro/2017 a janeiro/2018, na forma de um curso de extensão, ofertado no Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, com 30 vagas para estudantes de cursos de nível superior na área de Informática. O pré-requisito era tão somente o estudante ter cursado previamente alguma disciplina de programação. Para este curso, houve 53 (cinquenta e três) inscrições, sendo 30 (trinta) estudantes classificados, conforme o coeficiente de rendimento. Como o período de realização do curso coincidiu com o encerramento do semestre, concorrendo com as avaliações finais dos cursos regulares nos quais os alunos estavam matriculados, e com as férias de final de ano, 15 (quinze) alunos desistiram, mesmo tendo assinando termo de compromisso. Desta forma, os resultados apresentados na próxima seção serão reportados com base nos alunos que concluíram o curso (15 alunos).

4. Resultados

Dos alunos que concluíram o curso, 80% eram do sexo masculino e 20% do sexo feminino. A média de idade do grupo era de 24,2 anos. Identificamos também que 80% dos alunos haviam cursado, no mínimo, uma disciplina de redes na graduação.

Como falamos anteriormente, antes de iniciarmos o curso, os alunos realizaram uma avaliação diagnóstica contendo: (i) questões objetivas sobre a teoria básica de redes de computadores, envolvendo classificação de redes por distribuição geográfica, principais protocolos e modelos de pilhas de protocolos; e, (ii) uma questão prática envolvendo a automatização com verificação de disponibilidade de equipamentos em uma rede, com envio de alerta por e-mail.

Com respeito às questões teóricas objetivas, mais de 84% dos estudantes não acertaram questões básicas como associar o comando *ping*[8] ao protocolo ICMP (*Internet Control Message Protocol*). Dificuldade semelhante foi percebida quando quase 70% dos estudantes tampouco demonstraram conhecimentos sobre a utilidade do protocolo SNMP[9] (*Simple Network Management Protocol*). Com respeito à questão prática, 30% dos estudantes deixaram de resolvê-la e, dos que tentaram resolver, seus desempenhos não ultrapassaram 35% de acertos.

Considerando a característica do grupo, no qual 80% dos alunos já haviam cursado uma disciplina de redes de computadores, era esperado melhores resultados nas questões teóricas. Os resultados desta avaliação diagnóstica ratificam as dificuldades de

[8] O comando *ping* faz parte do protocolo ICMP e é utilizado para testes básicos de comunicação entre computadores.

[9] SNMP é um protocolo utilizado para gerenciamento de redes de computadores e seus componentes.

aprendizagem em redes, conforme contextualizamos na Seção 2.

Com respeito à realização dos roteiros de aprendizagem administrados ao longo do curso, os estudantes concluíram 86% dos RA para estudo autônomo e 87% dos RA para estudo presencial. Embora a evolução dos estudantes tenha sido evidenciada ao longo do curso, percebemos na maioria dos alunos uma dificuldade inicial em desenvolver seus estudos autônomos, pois a experiência de "estudar sozinho" parecia inédita e/ou desconfortável para muitos. Em virtude disso, houve entregas posteriores ao prazo estabelecido e também ausências de entregas, conforme constatado pelos números apresentados acima. Cabe destacar que, a não entrega dos RA para estudo autônomo acabava impactando na entrega dos RA para estudo presencial, pois faltava aos alunos a fundamentação conceitual necessária para a resolução dos problemas propostos.

No decorrer do curso, os estudantes foram capazes de automatizar soluções para 12 (doze) problemas, distribuídos em 6 (seis) RA para estudo presencial. Estes problemas evoluíam desde situações simples envolvendo monitoramento de conexões de redes com o comando "*ping*", até automatização de análise remota de *logs*, monitoramento e emissão de alertas sobre a ocupação de partições de discos de máquinas remotas, sincronismo de dados entre servidores, cópias de segurança em rede, além de envio de alertas automáticos via e-mail e Telegram, orientados à criticidade de cada situação. Os alunos também tiveram que montar e administrar seus laboratórios virtuais, utilizando o sistema operacional Linux sobre a plataforma de virtualização Oracle VM Virtualbox, ferramentas até então desconhecidas pela maioria.

No processo de resolução de problemas, identificamos dificuldades dos alunos com respeito à programação, mais especificamente, com a manipulação de *strings,* arquivos e estrutura de laços de repetição. Também identificamos desconhecimento sobre o sistema operacional Linux, mitigados pela intensificação de atendimento a alunos de forma presencial e remota, indicação de vídeos de terceiros e pela produção de uma apostila envolvendo estes assuntos.

A fim de proporcionar um *feedback* contínuo aos estudantes, ao final de cada estudo presencial, os estudantes recebiam de maneira individual, por e-mail ou Whatsapp informações sobre seu desempenho, com indicação de quais aspectos teóricos e/ou práticos deveriam melhorar. Estes *feedbacks* foram muito importantes e refletiram em melhorias no desempenho individual ao longo do curso.

Na avaliação intermediária, ocorrida ao final do primeiro módulo, a média da turma foi 6,8 pontos (em uma escala de 0 a 10), acima da nota mínima para aprovação no curso e também acima da média aceita para aprovação nos CST regulares que os estudantes cursavam, ambos de 6,0 pontos. Na avalição final, realizada após o término do segundo módulo, a média da turma foi de 8,7 pontos (em uma escala de 0 a 10), representando uma melhora significativa no desempenho do grupo.

Com base nesta aplicação do curso foi possível identificar aspectos a serem melhorados, como por exemplo: (i) ampliar o material de apoio sobre Python no que diz respeito à manipulação de *strings*, arquivos e estrutura de repetição, com exercícios específicos para consolidar a aprendizagem; (ii) incluir conteúdos básicos sobre o sistema operacional Linux, a fim de proporcionar aos alunos melhor familiaridade com seus conceitos e operações; e, (iii) melhorar as vídeoaulas para esclarecer termos e conceitos básicos de redes, que foram mencionados pelos alunos ao longo do curso.

Com respeito ao estudo autônomo, a aplicação deste curso revelou algumas demandas para professores e gestores de Instituições de Ensino Superior (IES). Primeiro, é necessário que as IES disponibilizem infraestrutura adequada para os alunos, permitindo-lhes, por exemplo, acesso aos laboratórios ou outras dependências fora do horário de aula. Para os professores, é necessário estabelecer com a turma uma "cultura de estudo autônomo" e para isso, os alunos precisam ser orientados. No caso deste trabalho, identificamos os roteiros de aprendizagem como instrumentos adequados para orientar os alunos no estudo. Porém, a preparação dos roteiros e dos materiais de suporte demandam um tempo considerável de planejamento. Em contrapartida, possuem a vantagem de poder ser reutilizados em outras ofertas do curso.

Considerando a plataforma web utilizada, o Google Classroom teve alta aceitação entre os alunos, isto porque todos eram usuários de e-mail da Google e podiam facilmente aproveitar os recursos disponibilizados, com destaque para o compartilhamento de materiais, vídeos e submissão das respostas dos alunos aos roteiros de aprendizagem. Contudo, para as comunicações virtuais e trocas de informação, o tempo de resposta era mais efetivo com a adoção do Whatsapp.

5. Considerações Finais

Neste artigo reportamos o planejamento de um curso de programação para administração de redes de computadores, assim como os resultados de sua aplicação com estudantes de Computação. O curso reuniu conteúdos de redes, sistemas operacionais e programação, em uma perspectiva prática, pautada na resolução de problemas e tendo no modelo de sala de aula invertida a dinâmica para condução do processo de ensino-aprendizagem.

O planejamento do curso demandou seis meses de trabalho e incluiu a elaboração da ementa, conteúdos, organização dos roteiros, elaboração dos problemas práticos, criação dos recursos de apoio (vídeos, textos, códigos, laboratórios virtuais, etc.), além de identificação de recursos disponíveis na Internet que pudessem ser reutilizados no curso para auxiliar os alunos no estudo autônomo.

Como trabalhos futuros, desenvolveremos um *guideline* para professores, disponibilizando todo o planejamento e recursos produzidos para o curso. Acreditamos que isso possa contribuir com outros docentes que ministram a disciplina de redes de computadores. Almejamos também desenvolver uma análise focada no nível de complexidade dos problemas propostos, a fim de identificarmos pontos de correção e redimensionamento dos problemas e dos próprios conteúdos associados.

Adicionalmente, realizaremos novos estudos com a aplicação da proposta de ensino-aprendizagem descrita neste artigo, pois compreendemos que a quantidade de estudantes envolvida nesta primeira experiência representa uma limitação do trabalho. Assim, em aplicações futuras, pretendemos oferecer o curso como uma disciplina do currículo, administrada durante um semestre acadêmico, o que permitirá uma avaliação mais criteriosa do desempenho dos estudantes, suas dificuldades, assim como das potencialidade e limitações do próprio curso, a fim de que possamos ampliar as discussões sobre Educação em Computação, mais especificamente neste trabalho, sobre o ensino de redes de computadores.

Referências

BENDER, W. N. **Aprendizagem baseada em projetos** – Educação diferenciada para o Século XXI. Porto Alegre: Penso, 2014.

BERGMANN, J; SAMS, A. **Flip YOUR Classroom** – Reach Every Student in Every Class Every Day. Alexandria: ISTE, 2012.

BIGGS, J.; TANG, C. **Teaching for Quality Learning at University**. 4. ed. Berkshire, England: Society for Research into Higher Education & Open University Press, 2011.

BRASIL. Ministério da Educação (2002). **Parecer Conselho Nacional de Educação – Conselho Pleno nº 29/2002**. Institui as Diretrizes Curriculares Nacionais Gerais para a organização e o funcionamento dos cursos superiores de tecnologia. Disponível em: <http://portal.mec.gov.br/cne/arquivos/pdf/cp29.pdf>. Acesso em: 23 ago. 2016.

BRASIL. **Parecer CNE/CES No 136/2012**. Institui Diretrizes Curriculares Nacionais para os cursos de graduação em Computação.

BRASIL. Ministério da Educação (2006). **Decreto n.º 5.773, de 9 de maio de 2006**. Dispõe sobre o exercício das funções de regulação, supervisão e avaliação de instituições de educação superior e cursos superiores de graduação e sequenciais no sistema federal de ensino. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/decreto/d5773.htm>. Acesso em: 01 set. 2016.

BRASIL. Secretaria de Educação Profissional e Tecnológica / Ministério da Educação (2016). **Catálogo Nacional de Cursos Superiores de Tecnologia.** 3a edição. Disponível em: < http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=44501-cncst-2016-3edc-pdf&category_slug=junho-2016-pdf&Itemid=30192 >. Acesso em: 10 fev. 2018.

CHEN H, Discussion on the Teaching Methods of Applied Undergraduate Computer Network Courses. 3rd International Conference on Science and Social Research (ICSSR 2014), 2014, Paris. **Proceedings...** Paris: Atlantis Press, 2014, p. 127-130.

KUROSE, J. F.; et al. Workshop on computer networking: curriculum designs and educational challenges. **ACM SIGCOMM Computer Communications Review**, v. 32, n. 5, p. 1–9, Nov. 2002.

WANG, Y; BLUM, T; McCOEY, M. Teaching a networking class for freshmen: course design and lessons learned. In: Proceedings of the 15th Annual Conference on Information technology education, 2014, Atlanta. **Proceedings...** Atlanta: 2014, p. 9-14.

Kids Block Coding Game: A game to introduce programming to kids

Luís E. T. Forquesato[1], Juliana F. Borin[2]

[1] SIDI – Samsung Instituto de Desenvolvimento para a Informática
Rua Aguaçu, 171, Lot. Alphaville Campinas – 13098-321 Campinas, SP – Brasil

[2] IC – Instituto de Computação – UNICAMP
Av. Albert Einstein, 1251, Barão Geraldo – CEP 13083-852 – Campinas, SP – Brasil

luisforque@gmail.com, juliana@ic.unicamp.br

Abstract. *In the near future, many areas will need at least a basic knowledge of computer science. Nonetheless, studies have shown that professors are currently having trouble teaching those same concepts in universities. In consequence, the idea of introducing programming theory to children is trending. This article presents a game that provides problems for the user to solve using blocks. The game targets the six to eleven age range, although it supports other ages as well. User trials affected the product through the development phase, changing user interaction and game design. A few children have already tested the final product and the results were positive: the tool was regarded as being fun, educational, and engaging.*

1. Introduction

Computational thinking education is a topic that has acquired a great deal of attention after a publication by Jeannete Wing [Wing 2006] in which she argues the importance for people in the job market to know basic concepts of computer science, such as abstraction, automation, and complex problem solving.

Teaching programming is a common choice to introduce computational thinking concepts. When learning programming, students face several challenges, problem solving and debugging among them. These activities are central to the concept of computational thinking [Flórez, Casallas, Hernández, Reyes, Restrepo and Danies 2017]. Previous research shows that teaching logic and programming to undergraduates is a hard task [Bromwich, Masoodian and Rogers 2012] [Liu, Cheng and Huang 2011]. This finding lead us to believe that the earlier a person is accustomed with those abstract concepts, the easier it will be for her to learn and use that knowledge when necessary. According to Piaget, kids of age approximately seven to eleven are on the concrete operational stage [Piaget and Cook 1952], in which they already can construct abstract ideas and logical structures on their mind.

Children are spending more time than ever[1] using mobile devices. At the same time, it is becoming common practice to teach children a very basic form of

[1] The Common Sense Census: Media use by kids age zero to eigth 2017
https://www.commonsensemedia.org/research/the-common-sense-census-media-use-by-kids-age-zero-to-eight-2017

programming or robotics, either in school or in extra-curricular classes. This scenario has driven research in educational games and applications. However, most of the research done on this topic focuses on older children [Bromwich, Masoodian and Rogers 2012] [Denner, Werner and Ortiz 2012] [Werner and Campe 2012] [Liu, Cheng and Huang 2011] [Werner, Denner and Campe 2012] [Webb 2010] [Brennan and Resnick 2012]; there are not so many examples of studies analyzing the usage of those educational software for children younger than 10 years old.

Kids Block Coding Game was developed to be a game that introduces programming concepts by using blocks to resolve a pathing problem. The focus is on the 6 to 10 age range. Besides the educational goal of the project, another motivation is to investigate the possibility of using stealth assessment [Shute 2011] to determine whether the kids are indeed learning computational thinking and programming concepts while playing the game. This analysis is not present in this work but will be discussed in the section containing future work. This paper describes in further detail the game's development, user trials, and final product.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 explains how the application was developed: listing important requirements and noteworthy implementation details. Section 4 presents the preliminary user tests and the acquired results. Finally, Section 5 contains the conclusion and the future work for the study.

2. Related work

During the research for computational thinking tools and applications, the authors noticed that there are many alternatives for the older audience; in particular, Scratch[2] is very well known. However, there are fewer good options for younger children. Kids in age 6 to 10 can already understand basic logic and abstract concepts, meaning they are valid targets for elementary logic applications.

Duncan et al. produced a theoretical study about teaching computational thinking in different ages [Duncan, Bell and Tanimoto 2014]. The authors ask a valid question in "Should your 8-year-old Learn Coding?", discussing historical elements, developmental psychology, school curriculum, among others. The authors then conclude that it is important for kids to learn this skill, as long as some important restrictions are met. Last year, during the Workshop of Education in Computation (WEI), the Brazilian Computing Society (SBC) proposed a scenario[3] in which a basic sort of computational thinking is taught at schools to children as early as four years old.

There are many tools used to teach computational thinking. Commonly, computational thinking tools can be categorized into games, robotics, or programming [Duncan, Bell and Tanimoto 2014]. Duncan et al. lists 47 tools graded across difficulty

[2] https://scratch.mit.edu/

[3] http://computacaonaescola.com.br/proposta-de-inclusao-de-computacao-na-educacao-basica/ (in portuguese)

and age ranges. Within those, the authors found 20 applicable to kids between 3 and 10 years old.

Similar to the application proposed in this paper, there is the Box Island[4]. It is well known, has received awards and it is considered for the Hour of Code[5] certificate. Its age target is six plus years old. Box Island includes complex concepts like functions and has 100 levels.

Another similar application is Lightbot[6]. Lightbot uses different types of blocks and visual, but the concepts involved are similar. Lightbot is said to be all-ages friendly and has 40 puzzles. Gouws et al. did a case study in which they analyzed the game for which computational thinking concepts are being passed to the user in each puzzle [Gouws, Bradshaw and Wentworth 2013]. The authors concluded that "models and abstraction" are exercised in 92% of the levels, while "inference and logic" is practiced in 50%. By extrapolation, they concluded that the game teaches computational thinking well, presenting 74% of the content of computational thinking in some way.

Game Logic aims to help teaching programming logic [Netto, Medeiros, de Pontes and de Morais 2017]. The game is organized in levels, and the user needs to use programming blocks to solve each problem. The central difference between Game Logic and Kids Block Coding Game is that the latter has a younger age target.

3. Development of the game

Samsung has an ecosystem[7] with a handful of mobile applications dedicated to children, and the proposed game was developed to be a part of that context. The characters can be, for that reason, already known to the user. The game's commercial name is Crocro's Adventure.

Kids Block Coding Game works in all sorts of devices and tablets, of multiple screen sizes and hardware. Despite originally targeting children of ages 6 to 11, the game can be played by younger kids because it does not have text, and can be challenging to adults.

3.1. Requirements

At the first moment, the idea was to create an application that was fun and challenging for children, but could also teach basic programming. When targeting young children, we knew it could not be too hard or complicated. To that end, the choice to use block programming was natural, seeing that they are intuitive and not punishing by grammar

[4] https://boxisland.io/

[5] *"The Hour of Code started as a one-hour introduction to computer science, designed to demystify "code", to show that anybody can learn the basics, and to broaden participation in the field of computer science."* https://hourofcode.com/

[6] http://lightbot.com/

[7] http://www.samsung.com/global/galaxy/apps/kids-mode/

mistakes. Trying to appeal to children′s spatial knowledge, the game was thought up as a puzzle, in which the main character is lost in a simple maze and the user needs to use blocks to show him the way out. New blocks are unlocked with new levels. Using blocks was a natural decision because they visually represent a command that is going to execute and are easy to use and to understand by kids, without the onus of having a complex syntax.

The application has some elements that closely relate to programming, in order to complete its initial goal of being educative. The order that the user places blocks in the command line is the same as the one in which commands will be executed, for example. This is analogous to how programming works, where code is entered and executed in a specific order. Similarly, the blocks are executed only after a ′play′ button is selected, which is comparable to running a program after coding. During execution, the player can see exactly what action each command has the character do. When the execution fails, the software places a red sign on the wrong block and the user receives the option to debug his solution.

To make the game fun and engaging, the game design has a storytelling element. Upon entering the game, the user sees a video showing why he should help the character by solving his puzzles. Users can also earn prizes and unlock more content after finishing a task. The story shows that the character lives in a boat and has a collection of beautiful candies kept in his fridge, but after a huge storm the boat crashes and he loses all his cherished candy, spread all around the world. Every level is a chance to recollect one of his candies or unlock ways to help him in future tasks, by moving the character from one island to the next.

The game′s structure consists of 41 stages with increasing difficulty, and the user can only proceed to the next level by finishing the current one. The first few challenges presents the basics of the game, with a hands-on tutorial on how the gameplay works. Characters are unlocked throughout the game, up to four by the end. Each character has a special ability based on its personality that helps him cross a specific type of obstacle. Following the common, directional blocks, players unlock the ′jump′ block, that can allow them to jump over a small piece of water; and then multipliers, that can be attached to another block and have that command execute repeatedly, equivalently to a loop. In order to support creativity and increase replay value, many puzzles can be solved with more than one solution.

Using directional blocks provide the opportunity for users to practice their abstraction, because they need to visualize how that block will affect the character without actually seeing it happen. Likewise, the ‘jump’ block represents a very basic logic condition, considering that kids will need to think, “If there is an empty space, then I need to ‘jump’”. The multipliers embody a primitive control flow, and every character’s special powers allow for the design of harder puzzles, which in turn can lead to better problem solving skills.

The application has some secondary use cases. One important requirement was that the application should provide a possibility for multiplayer local interaction,

between child and parent, or siblings, for example. This feature was implemented by creating a screen where one user can create puzzles and another can solve it. Finally, the application also provides a lounge where kids can have fun by interacting with unlocked content, without needing to solve puzzles.

Since this game is part of a research project which aims to investigate the use of stealth assessment to assess computational thinking skills it was important to have every relevant event made by the user logged and saved. Collecting analytics data is crucial to analyze the difficulty of the levels, how long it usually takes a kid to solve it, when he/she grows weary, among others. Logged information includes, for example, adding or removing a block, selecting the 'play' button, winning or losing, entering and exiting the application or each screen.

Finally, while brainstorming the application, the idea to produce a game with as little text as possible appeared. There are no menus and explanations on how to play the game. This not only enables the game to be played by younger kids who cannot read well, but also facilitates the internationalization part of the commercialization.

3.2. Architecture and implementation

The application was implemented using Unity3D. Unity3D is a game engine that facilitates game development immensely. The programming language used was C#. Most of the game was implemented in 3D mode, with an isometric camera that moves while the user swipes the screen.

The game does not need an internet connection to be played; all of its content can be accessed locally and is stored locally. The game is available for downloading in Android[8] and iOS[9] devices.

The architecture used was based upon Clean Architecture[10]. Clean Architecture is a collection of design patterns that appeared specifically for Android native development, but can be used in other types of applications, such as Unity3D.

3.3. User interface

The user interface is composed of realistic backgrounds with cartoon-like elements to increase engagement by the kids. There are four types of screens: the screen where the user can choose the next level; the screen where he can see the content he has unlocked; the screen where he can build a custom level; and the screen where he solves the level.

The screen where the player chooses the level is a horizontal-scrolling screen with only a couple of levels visible. Only by finishing those, the next few challenges appear. The authors took this decision to increase the gamification of the game; the player can have an achievement effect upon unlocking a new play section. Treasure

[8] https://play.google.com/store/apps/details?id=com.sec.kidsplat.kidsbcg
[9] https://itunes.apple.com/us/app/crocro-adventure/id1364966911?l=pt&ls=1&mt=8
[10] https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html

chests and bosses challenges were scattered to make the game funnier and less repetitive. Treasure chests contain either new blocks or new characters. The screen in Figure 1 is the base; it redirects the user to the others.

Figure 1. The first screen, where users choose the level to play

The puzzle screen (Figures 2 and 3) is where the game actually happens. The user sees one island on the left, where the character awaits, and another island on the right, where the level objective is. On the bottom part of the screen, the player can see the available blocks, the command line with empty spaces where blocks are added, and the 'play' button used to start the execution after the command line is filled accordingly. Once the player taps the 'play' button, the character starts moving in the direction represented by the commands, trying to cross to the other island. If the character reaches the goal and the solution is correct, congratulations appears and the user is taken to the previous screen. Otherwise, the incorrect block renders red, the character goes back to the start and the child can try again. By showing the incorrect block and allowing the player to try again, the authors expect to stimulate trial and error and debugging, both useful programming techniques. The game does not happen live, meaning that first, the kid does something, and only after clicking 'play', the solution runs. This promotes planning, organization and decomposition of the solution in steps.

Figure 2. One of the first levels, while the character is animating the solution

Figure 3. A harder level showing a wrong solution and the visual feedbacks

The third screen is where the child can see what he has unlocked. The characters are animating, the candies are listed, and the videos are watchable. This screen encourages the kid to boast his achievements by interacting with others, either his family or his friends, for instance.

In addition, to improve social interaction and collaboration, the application has a screen where the user creates and plays new puzzles. This feature allows, for example, a parent to create a custom level specific to his child difficulties.

There are also a few videos scattered between puzzles, in specific moments when the player is going to unlock important elements. Videos are used to capture and increase a child's attention.

4. Tests and results

During the development phase, we had user trials with 15 kids in order to find design problems, validate concepts and discover how they felt about the game. The kids were placed in a closed meeting room with their parents and given the game to play. There was no introduction on how to play the game, and interaction was kept to a minimum.

Early in development, the application had a 'reset' button that would clear the command line, removing all blocks already added. This seemed natural to the authors, being adults, but in the user trial, not a single child used it; they would just simply remove blocks one of a time by dragging and dropping. Similarly, in the beginning of the project, correctly solving problems would give the player stars; he would get three stars for an optimal solution and less whenever he used more blocks than necessary. At the user trial, the authors noticed that most kids were not interested in the stars, did not care how many stars they got, they just wanted to go further in the game faster. In both cases, the user trial affected the game design for the user interaction.

In the first release version, the game play was similar to the present, but the application did not have unlockable characters, videos and sections. In the trial, kids would often run tedious and stop before completing the game. This caused a major rethink of features and how to keep them engaged, which led to the current product.

Furthermore, some kids enjoyed seeing the videos so much that the authors implemented a way that they can watch again all videos that they had already unlocked.

Regarding the final product, the user trial reviews were majorly positive; the kids enjoyed the game new looks and features a lot better. They had more interest in solving the puzzles and continued to practice their logic and problem solving for a larger amount of time. With that said, we did not have a single child that managed to solve all levels and complete the game, possibly because the game is too long to be completed in one play session. Consequently, we had not been able to validate the difficulty of the more advanced levels.

Tests were mainly validating the user experience, and did not focus on learning, which is a harder appraisal. To do that, the authors will use usage statistics and a strict play session with a group of children from different age ranges, on another study.

The authors believe that the final game product turned out fun and engaging as well as challenging and informative. The product received international attention in the event Samsung Developers Conference 2017, in San Francisco[11].

5. Conclusion and future work

This paper presented the application Kids Block Coding Game, an educative game proposed to teach logic and basic computational thinking to children using block programming.

The impact of user trials on the development of the current version of the game was discussed. It is interesting to note that many of the choices made by experienced programmers when developing a tool to teach programming had to be rethought during this process. The authors believe that this discussion may greatly contribute to other researchers/developers interested in developing tools for computational thinking and programming education.

In the future, the authors ought to validate the difficulty of the whole game against kids of age six to eleven. Additionally, the application is part of another study that intends to validate the use of stealth assessment [Shute 2011] to determine if the player is learning logic and computational thinking concepts. For this study, a strict user trial will be conducted, and log usage from participants will be analyzed quantitatively to conclude if the application is indeed educational, and thereafter if this assessment technique is viable in this scenario.

11 *http://www.samsungmobilepress.com/stories/sdc-2017:-our-commitment-to-helping-kids-gain-digital-literacy*

Acknowledgements

The authors would like to thank the support from SIDI and Samsung in providing the possibility and time for developing the application and writing this article. The authors would also like to thank the whole team that helped create Crocro's Adventure.

Finally, a special thanks to the kids that participated in the voluntary user trials, providing valuable feedback that resulted in this fun and educational game.

References

Bromwich, K., Masoodian, M. and Rogers, B. (2012) "Crossing the Game Threshold: A System for Teaching Basic Programming Constructs", Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest Group of Human-Computer Interaction, 56-63.

Brennan, K., Resnick, M. (2012) "New frameworks for studying and assessing the development of computational thinking", American Educational Research Association 2012, 1-25.

Denner, J., Werner, L., Ortiz, E. (2012) "Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?", Computers & Education Volume 58 Issue 1, 240-249.

Duncan, C., Bell, T. and Tanimoto, S. (204) "Should Your 8-year-old Learn Coding?", Proceedings of the 9th Workshop in Primary and Secondary Computing Education, 60-69.

Flórez, F. B., Casallas R., Hernández M., Reyes A., Restrepo S., Danies G. (2017) "Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming", Review of Educational Research Vol 87, 834-860.

Gouws, L., Bradshaw, K. and Wentworth, P. (2013) "Computational Thinking in Educational Activities", Proceedings of the 18th ACM conference on Innovation and Technology in Computer Science Education, 10-15.

Liu, C., Cheng, Y. and Huang, C. (2011) "The effect of simulation games on the learning of computational problem solving", Computers & Education 57, 1907-1918.

Netto, D., Medeiros, L. M., de Pontes, D. and de Morais, E. (2017) "Game Logic: Um jogo para auxiliar na aprendizagem de lógica de programação", 25º Workshop sobre Educação em Computação.

Piaget, J. and Cook, M. (1952) "The origins of intelligence in children", New York: International University Press.

Shute, V. J (2011) "Stealth assessment in computer-based Games to support learning", Computer Games and Instruction, 503-523.

Webb, D. (2010) "Troubleshooting assessment: an authentic problem solving activity for it education", Procedia - Social and Behavioral Sciences Volume 9, 903-907.

Werner, L., Campe, S. (2012) "Children learning computer science concepts via Alice game-programming", Proceedings of the 43rd ACM technical symposium on Computer Science Education, 427-432.

Werner, L., Denner, J., Campe, S. (2012) “The fairy performance assessment: measuring computational thinking in middle school”, Proceedings of the 43rd ACM technical symposium on Computer Science Education, 215-220.

Wing, J. M. (2006) “Computational Thinking”, Communications of the ACM 49.3, 33-35.

Relato de Experiência da Aplicação da Metodologia Ativa de Ensino com Pesquisa na Disciplina de Sistemas de Informação

Felipe M. Sampaio[1], Jefferson P. de Almeida[1]

[1]Instituto Federal do Rio Grande do Sul (IFRS) - Campus Farroupilha
Farroupilha - RS - Brasil

{felipe.sampaio, jefferson.almeida}@farroupilha.ifrs.edu.br

Resumo. *Este artigo apresenta um relato da experiência da aplicação da metodologia ativa de Ensino com Pesquisa como instrumento da prática docente, com o objetivo de aprimorar o processo de ensino-aprendizagem. Os princípios básicos desta metodologia foram utilizados como base para o planejamento da prática docente em uma turma de ensino técnico da área de informática, na disciplina de Sistemas de Informação. Como base para as atividades de sala de aula, dois projetos de pesquisa foram elaborados como elementos norteadores do processo de ensino-aprendizagem segundo a metodologia utilizada. Como reflexões sobre a experiência da aplicação da estratégia de Ensino com Pesquisa, notou-se um aumento na participação os dos alunos, tanto na realização quanto na proposição de atividades. O professor teve seu papel perante os estudantes modificado, atuando agora como um orientador das atividades de pesquisa desenvolvidas em sala de aula. No que tange o processo de acompanhamento das atividades e de avaliação da aprendizagem, ocorreu uma valorização maior ao percurso investigativo percorrido pelo aluno, dado pela organização metodológica das atividades de acordo com as etapas do método científico.*

Abstract. *This paper presents an experience report of the active methodology "Teaching with Research" with the goal of improving the teching-learning process. The basic principles of this methodology were utilized as basis for the teaching planning for the Information Systems course in an informatics technician class. Two research projects were elaborated as guiding elements of the teaching-learning process (according to the adopted methodology). As reflections regarding the methodology application experience, it could be noticed increased participation of the students during the class activities. The professor has a modified role towards the students: acting like an advisor, guiding the developed research activities inside the classroom. With respect to the learning evaluation methods, it happened an increased valorization of the research path coursed by the student, mainly due to the methodologic organization of the proposed activities (based on the steps of scientific method).*

1. Introdução

Metodologias tradicionais de ensino, fortemente baseadas na transmissão de conhecimentos, privam os alunos de processos mentais que são imprescindíveis na construção das suas representações próprias do conhecimento. Da mesma maneira que crianças necessitam agir sobre o mundo de forma a experimentá-lo das mais diferentes formas, a sala de aula também deve proporcionar oportunidades para que os alunos

realizem suas investigações e experiências (BECKER; MARQUES, 2007). Além disso, a partir de metodologias de ensino que incorporem estes processos de construção do conhecimento, abre-se a oportunidade para que os resultados dos procedimentos de avaliação da aprendizagem se detenham mais no acompanhamento do percurso do aluno, não sendo apenas o produto da reprodução mecânica dos conceitos apreendidos (LUCKESI, 2011).

Neste sentido, pesquisadores da área da educação vêm desenvolvendo metodologias de ensino diferenciadas (chamadas de metodologias ativas), as quais consideram o aluno como um agente pesquisador e um solucionador de problemas dentro do processo de ensino-aprendizagem. Este trabalho foca, em especial, em metodologias que incorporam aspectos inerentes da pesquisa científica, utilizando-a como instrumento para as atividades de sala de aula. Esta teoria é chamada de Ensino com Pesquisa e tem como pilar fundamental o uso dos procedimentos típicos do método científico para promover a (re)construção do conhecimento (DEMO, 1991).

A área de computação tem um potencial especial quando consideramos o uso de metodologias ativas. Considerando especificamente a metodologia de Ensino com Pesquisa, há imensas oportunidades em disciplinas da computação para o uso desta estratégia, principalmente quando se considera o incentivo à problematização e o uso de raciocínio lógico estruturado. Como estudos de caso, vários trabalhos da literatura trazem relatos e análises de experiências na aplicação de metodologias ativas em disciplinas da computação, demonstrando o interesse acadêmico na aplicação de tais estratégias nesta área do conhecimento (LIMA et al, 2016; BRANDÃO; NEVES, 2014; SANTOS; COSTA, 2006).

Neste contexto, este trabalho tem como *objetivo relatar experiências e realizar uma reflexão sobre as contribuições da aplicação na prática docente da metodologia baseada no Ensino com Pesquisa, como forma de aprimoramento do processo de ensino-aprendizagem*. Esta reflexão se dará por meio de um relato de experiência da prática pedagógica em uma turma de ingressantes do Curso Técnico em Informática Integrado. A metodologia foi aplicada na disciplina de Sistemas de Informação, a qual pertence ao núcleo técnico do curso.

2. Referencial Teórico

A estratégia pedagógica de Ensino com Pesquisa se vale dos princípios educativos da pesquisa científica para o aprimoramento da prática docente (DEMO, 1991). Nesta metodologia de ensino, professores e alunos realizam constantemente as etapas definidas pelo método científico nas atividades de sala de aula. O objetivo é promover a construção, desconstrução e reconstrução do conhecimento a partir de planejamentos de pesquisa realizados previamente. Estes planejamentos são construídos a partir de temáticas que sejam resultantes do programa da disciplina e são constituídos de modo a mobilizar e despertar o interesse dos alunos por meio do processo investigativo. Tem-se, desta forma, um processo de ensino-aprendizagem conduzido na direção de um conhecimento metódico, formado a partir de situações no universo da pesquisa científica (POZO; CRESPO, 2009).

Considerando o caráter educativo da pesquisa, vale ressaltar que a construção do processo de investigação e o desenvolvimento das etapas metodológicas são tão, ou até

mais, importantes do que propriamente os resultados alcançados. Desta forma, critérios adotados para avaliação da aprendizagem, quando inseridos neste contexto, devem levar em conta um acompanhamento processual das atividades, levando em conta o entendimento e a desenvoltura dos estudantes perante às etapas de construção do conhecimento (LIMA, 2000).

Um dos grandes benefícios desta estratégia, quando trazida para o aperfeiçoamento da prática docente, é a transformação do papel do aluno dentro da sala de aula. Agora, ele se torna um agente participante do processo construtivo do conhecimento (DEMO, 1991). Desta forma, há um incentivo ao desenvolvimento de sua atitude de pesquisa, fazendo com que o estudante aprimore cada vez mais sua "capacidade de aprender a aprender" (LIMA, 2000).

O docente, quando imerso nesta metodologia de ensino, deve organizar os encontros de sala de aula como se estivesse em processo de orientação dentro de um grande grupo de pesquisa. O professor orientador tem como papel, para cada uma das atividades investigativas planejadas, agir de modo a acompanhar seus alunos nas seguintes tarefas: (1) coleta e catalogação de material para embasar as atividades, (2) incentivo aos estudantes para eles mesmos elaborarem suas próprias interpretações acerca dos estudos realizados, (3) insistência para a (re)construção do conhecimento e, por fim, (4) desafio à turma de estudantes para que eles próprios possam elaborar suas próprias impressões e conclusões sobre a atividade (DEMO, 2011).

É condição fundamental para a organização das atividades na metodologia de Ensino com Pesquisa a construção de projetos de pesquisa norteadores (LIMA, 2000). O programa da disciplina deve ser encarado de forma a se descobrir possibilidades para problematização e para a sistematização das atividades de acordo com a metodologia científica. A elaboração destes projetos deve ser realizada, sempre que possível, em cooperação com os estudantes. Inclusive, é desejável que, dependendo da carga horária de trabalho com a turma, os próprios estudantes problematizem questões referentes ao conteúdo a ser trabalhado e, eles mesmos, construam o planejamento metodológico da atividade.

De acordo com Lima (2000), alguns itens imprescindíveis para uma correta elaboração de um projeto de pesquisa, quando utilizado para princípios educativos, estão relacionados à forma de execução da atividade investigativa. Inicialmente, um planejamento de pesquisa deve ser delimitado, trazendo para os alunos qual o tema a ser abordado, o problema (ou a pergunta) de pesquisa que será trabalhada, as justificativas e motivações para o projeto e os objetivos a serem alcançados. Ademais, detalhes referentes à metodologia que será adotada para a execução da atividade investigativa devem ser trazidos, tais como: formulação de instrumentos de coleta, formas para o tratamento dos materiais e dos dados adquiridos, bem como a construção de descrições e análises capazes de fundamentar as conclusões alcançadas.

3. Metodologia

Esta seção tem como objetivo descrever a metodologia utilizada neste trabalho. Este artigo engloba o relato das experiências da prática docente na turma de primeiro ano do Curso Técnico em Informática Integrado, mais especificamente na disciplina de Sistemas de Informação. A turma era composta por 34 estudantes. O período da prática

docente nesta turma totalizou uma carga horária de 30 horas. Vale ressaltar que a disciplina tem o período anual com carga horária de 64 horas.

A disciplina de Sistemas de Informação é parte integrante do núcleo técnico do currículo do Curso Técnico em Informática Integrado. Dentre aqueles especificados para a disciplina no Plano de Ensino, o período da prática docente relatado neste trabalho aborda os seguintes objetivos específicos: (1) Aplicação da tecnologia da informação como base para a construção de sistemas de informação e (2) Estabelecimento de relações entre a evolução histórica da área de computação (e dos sistemas de informação) com os impactos na sociedade contemporânea.

Conforme definido na fundamentação teórica, a prática pedagógica utilizando a metodologia de Ensino com Pesquisa está baseada na formulação de projetos de pesquisa como elementos norteadores das atividades de sala de aula. Para cada um dos objetivos adotados para a prática docente (apresentados na Seção 3) foi elaborado um projeto de pesquisa.

Os projetos foram construídos e organizados em três partes principais. Inicialmente, (1) um planejamento da metodologia de pesquisa que seria utilizada foi definido, especificando o tema de pesquisa, a pergunta (ou problema) de pesquisa, os objetivos e a metodologia que seria adotada. Após, (2) um cronograma das aulas foi definido, apresentando em uma forma sistematizada a divisão das atividades propostas ao longo do tempo. Por fim, (3) detalhes sobre os procedimentos de avaliação da aprendizagem foram especificados, indicando os instrumentos e os critérios de avaliação que seriam utilizados ao longo do desenvolvimento do projeto de pesquisa em questão.

Sempre no início da execução de cada um dos dois projetos elaborados, um tempo da primeira aula era destinado à discussão coletiva dos mesmos. Nestes momentos, todos os detalhes eram passados aos alunos, que tinham a oportunidade de contribuir com sugestões ou alterações nas especificações. Uma vez finalizada a construção do projeto, este era disponibilizado a todos por meio do ambiente virtual de aprendizagem utilizado pela instituição. A cada aula, os principais tópicos do projeto eram relembrados, sempre com a ideia de tornar aquela aula específica parte integrante e fundamental para o alcance dos objetivos definidos no projeto.

A seguir, os dois projetos de pesquisa elaborados e trabalhados em sala de aula, como parte necessária para a aplicação da metodologia de Ensino com Pesquisa, serão apresentados.

3.1 Primeiro Projeto de Pesquisa: “Impactos da Evolução dos Computadores e dos Sistemas de Informação”

O resumo da etapa de planejamento do primeiro projeto de pesquisa está apresentado no Quadro 1. Este projeto foi construído para tratar de diversos conteúdos programáticos definidos no Plano de Ensino da disciplina, como o histórico da evolução dos computadores e dos sistemas de informação, bem como os impactos desta evolução na sociedade contemporânea.

Quadro 1 - Resumo da Etapa de Planejamento do Projeto de Pesquisa "Impactos da Evolução dos Computadores e dos Sistemas de Informação"

Tema: Impactos da Evolução dos Computadores e dos Sistemas de Informação
Pergunta de Pesquisa: Como as evoluções tecnológicas, principalmente aquelas relacionadas com a evolução dos computadores e dos sistemas de informação, tem impacto nas relações sociais e de trabalho dos dias de hoje?
Objetivos: - Compreender as principais evoluções tecnológicas da computação e dos sistemas de informação nas últimas décadas; - Refletir sobre os recentes avanços tecnológicos na área da computação, relacionando com suas consequências no dia a dia da sociedade; - Se posicionar perante os pontos positivos e negativos de tais evoluções.
Metodologia: As reflexões acerca da temática serão baseadas em três recursos: (1) filme que aborda o tema; (2) texto que propõe discussões sobre o assunto; e (3) momentos em sala de aula para a compreensão do percurso tecnológico recente dos computadores. Para cada um destes três recursos serão realizadas atividades para registro dos pontos principais.

Como metodologia de trabalho, as atividades de orientação por parte do docente considerou a oferta de diferentes subsídios para a reflexão pessoal do aluno. Como um dos instrumentos, foi trabalhado o episódio "Fifteen Million Merits", do seriado britânico Black Mirror, o qual apresenta uma realidade alternativa onde os personagens vivem mais intensamente a vida virtual do que propriamente a vida real. Além disso, críticas ao consumismo e à manipulação midiática perante à sociedade são apresentadas. Outros instrumentos pedagógicos também foram utilizados, como: a leitura e a discussão de um artigo sobre o uso desenfreado das Tecnologias de Comunicação e Informação (ANDRADE; SILVA, 2008), bem como a construção coletiva de cartazes a partir das reflexões individuais sobre os materiais estudados.

Todas as atividades desenvolvidas observam os fundamentos de orientação, definidos pela metodologia de Ensino com Pesquisa: a atuação do docente como orientador das atividades, criando um clima de investigação e pesquisa dentro da sala de aula. Apenas as aulas iniciais tiveram o perfil expositivo, com o objetivo de apresentar os conceitos básicos sobre as dinâmicas que seriam trabalhadas.

3.2 Segundo Projeto de Pesquisa: "Identificação de Demandas e Proposição de Sistemas de Informação"

O Quadro 2 apresenta o resumo da etapa de planejamento do segundo projeto de pesquisa aplicado como instrumento da prática docente no campo prático deste trabalho. Neste projeto, as habilidades de problematização e identificação de demandas por sistemas informatizados foram desenvolvidas. Além disso, após o reconhecimento das necessidades, os alunos deveriam propor sistemas de informação que pudessem ser empregados nestes campos.

Quadro 2 - Resumo da Etapa de Planejamento do Projeto de Pesquisa "Identificação de Demandas e Proposição de Sistemas de Informação"

Tema: Identificação de Demandas e Proposição de Sistemas de Informação
Pergunta de pesquisa: *Cada grupo, de acordo com a demanda identificada, deveria elaborar o seu próprio problema (pergunta) de pesquisa a ser investigado durante o desenvolvimento do projeto.*
Objetivos: - Identificar problemas/necessidades que podem ser solucionados através de um sistema de informação; - Propor uma problematização do cotidiano na busca por deficiências que possam ser aprimoradas por sistemas computacionais; - Incentivar a criatividade na proposição de soluções de sistemas de software.
Metodologia: - Formação de grupos de trabalho: 3 integrantes; - Acompanhamento aula a aula: em cada aula serão apresentados detalhes de como construir cada etapa do planejamento da pesquisa; - Etapas que deverão ser vencidas para a conclusão do projeto: Definição do tema, Problema, Justificativas, Objetivos, Metodologia, Referências - Produtos do projeto: Apresentação para a turma: entre 5 e 10 minutos. Construção de um relatório (no Moodle/Google Docs) do resultado de cada uma das etapas do projeto.

Este trabalho, além de empregar os conceitos da metodologia de Ensino com Pesquisa na construção das atividades, também proporciona aos alunos a construção de seus próprios planejamentos científicos. A partir de demandas identificadas pela própria turma, os alunos se organizaram em grupos nos quais o objetivo foi o de planejar a construção de um software que contribuísse para a resolução da problemática definida inicialmente. De forma a prover um contato ainda maior com o método científico, o planejamento do desenvolvimento do software deveria ser feito em forma de um planejamento de pesquisa. Como produtos finais, foi pedido um relatório com cada uma das etapas do planejamento, bem como a construção de uma apresentação dos resultados de cada grupo para o restante da turma.

As atividades de sala de aula tiveram como objetivo instrumentar os alunos com detalhes mais práticos das etapas de planejamento da pesquisa: definição do tema, problematização, construção de justificativas e motivações, especificação de objetivos e planejamento metodológico. A dinâmica adotada para as aulas foi a seguinte. Os momentos iniciais dos encontros eram destinados a uma discussão sobre cada uma das etapas da metodologia científica. Após, de acordo com a temática escolhida por cada grupo, os alunos trabalhavam na construção desta etapa metodológica em seus próprios projetos. A prática docente para estes momentos era dada pela orientação grupo a grupo, auxiliando-os a organizar suas ideias e pesquisa em forma de planejamento científico.

Para prover um acompanhamento processual na construção de cada uma das etapas, um cronograma de entregas parciais foi elaborado. Após cada entrega parcial, o professor retornava aos alunos uma série de sugestões para a melhoria da etapa em questão. Este retorno tinha como objetivo contribuir com a construção dos alunos, orientando-os na correta elaboração das suas ideias. Desta forma, os produtos finais deste projeto foram finalizados já com uma série de contribuições do professor,

aumentando a qualidade final dos trabalhos. A última aula foi destinada a uma rodada de discussões com cada grupo. Nestas rodadas, o professor passou os pontos positivos e negativos que observou ao longo do processo.

4. Relato das Experiências e Discussões

O objetivo desta seção é, sob a luz dos conceitos teóricos trazidos na fundamentação deste artigo (Seção 2), relatar as experiências vividas na aplicação da metodologia de Ensino com Pesquisa como forma de aprimoramento da prática docente. Como forma de sistematização das observações realizadas, foram elaboradas notas de campo baseadas na metodologia de *Bogdan e Biklen* (1998).

4.1 Indissociabilidade entre Ensino e Pesquisa

Embebidos das noções básicas de como as atividades se dariam, criou-se um clima de trabalho e de constante descoberta na sala de aula. Ao longo do primeiro projeto, por exemplo, quando vários materiais foram trabalhados sob a mesma problemática (impactos da computação na sociedade), cada estudante teve a oportunidade de se munir de evidências para que, no fechamento do projeto, ele pudesse elaborar suas próprias conclusões sobre o tema. Da mesma forma, ao longo do segundo projeto, os alunos trabalharam em forma cooperativa, criando um ambiente de construção de ideias, com o objetivo de elaborar sistemas informatizados para serem empregados nas demandas levantadas pelos próprios alunos.

Vale ressaltar também o especial potencial de melhoria no ensino em disciplinas da área de informática quando utilizada a metodologia de Ensino com Pesquisa. Para os conteúdos que foram trabalhados no campo prático deste trabalho, a adaptação e elaboração dos projetos de pesquisa se mostraram diretas. Tipicamente, alunos enfrentam problemas com o raciocínio lógico e com os conceitos iniciais da área de computação. Desta forma, quando a sua atuação como estudante é guiada por um trabalho metódico o seu percurso escolar se torna mais natural.

4.2 Trabalho de Sala de Aula Conduzido por Projetos de Pesquisa

A elaboração dos projetos de pesquisa para o trabalho com a metodologia de Ensino com Pesquisa em sala de aula se mostrou bastante satisfatória. Do ponto de vista da construção (conduzida pelo docente), criou-se um momento de reflexão sobre o percurso didático que deveria ser oferecido aos alunos ao longo do trabalho de sala de aula. Este caminho foi especificado em forma de planejamentos científicos, contendo a descrição de todas as etapas metodológicas da pesquisa. Comparando com a simples aplicação do que foi pensado para o Plano de Ensino, foi possível notar uma melhor organização dos trabalhos de sala de aula.

Outra contribuição importante está relacionada com o entendimento dos alunos sobre o que seria trabalhado ao longo das aulas. Apresentar, discutir e aprimorar estes projetos em conjunto com a turma foi de essencial importância para uma maior integração e participação dos estudantes nas atividades desenvolvidas. Como mencionado anteriormente, decidiu-se incorporar dentro dos projetos de pesquisa informações sobre o cronograma das aulas e os critérios de avaliação que seriam adotados. Esta interconexão de informações compiladas em um único documento, bem como a sua disponibilização para a turma no início da execução do projeto, foi de

extrema importância para bom andamento da aplicação da metodologia de Ensino com Pesquisa.

4.3 Atuação do Professor como Orientador

Os encontros de sala de aula foram planejados para terem um clima de trabalho como se todos estivessem em um grande grupo de pesquisa. Em alguns momentos, como planejado, metodologias expositivas foram utilizadas para definir e delimitar os aspectos fundamentais das atividades. De forma geral estes momentos foram diluídos ao longo das aulas, geralmente acontecendo nos momentos iniciais de cada encontro.

Ao longo das atividades, o principal papel do docente foi o de orientação do percurso dos estudantes pelos conteúdos, conceitos e raciocínios que deveriam ser percorridos. Pensando no espaço físico tradicional da sala de aula, a área de atuação do professor saiu da parte frontal da sala e se deu de forma mais intensa em meio aos alunos, participando e orientando as discussões internas nos grupos de trabalho. Isso proporcionou um contato mais próximo com a turma, quando as conversas tinham um clima menos formal, abrindo espaço para uma grande troca de ideias.

4.4 Aluno como Partícipe do Processo

O ambiente de investigação e de criação impulsionado pela metodologia de Ensino com Pesquisa mobilizou de forma satisfatória o envolvimento dos alunos. Foi com os apontamentos deles que se chegou ao episódio do seriado britânico que foi trabalhado no primeiro projeto. Foram através de sugestões da turma que se implementou o sistema de construção dos relatórios por etapas, com avaliações parciais de cada etapa do trabalho. Foi por meio de ideias dos estudantes que se definiram as problemáticas que foram abordadas no segundo projeto trabalhado. Desta maneira, observou-se que a metodologia aplicada tirou do professor o papel centralizador de propor e de definir, por si só, como se daria o andamento do processo de ensino-aprendizagem.

4.5 Percurso de Construção do Conhecimento e Processo Avaliativo

Um principais pontos positivos encontrados durante a aplicação da metodologia de Ensino com Pesquisa foi a possibilidade de avaliação do percurso percorrido pelos alunos. Mais importante do que o resultado final, o qual poderia ser avaliado por um exame final, foi o desempenho e a evolução dos alunos ao longo do processo.

Exemplificando, o primeiro projeto previa como metodologia de trabalho uma avaliação final individual. Ao longo do processo, diversas atividades foram desenvolvidas com o objetivo de instrumentar os alunos com materiais e com reflexões a partir de diferentes pontos de vista. Cada uma destas atividades “parciais” gerou um pequeno registro, onde o professor pôde acompanhar os “resultados parciais” obtidos pelo aluno. A avaliação final foi construída de forma a reunir os resultados cada uma destas atividades: os alunos, inclusive, puderam consultar seus registros. Desta forma, mais do que uma avaliação de aprendizagem ao final, diferentes procedimentos foram aplicados para o acompanhamento do andamento do processo investigativo realizado.

No segundo projeto de pesquisa desenvolvido nas aulas, uma estrutura de entregas parciais, juntamente com o respectivo retorno do docente, foi implementada também no sentido de se ter um acompanhamento dos alunos. Neste caso, o professor

retornava aos alunos um parecer descritivo com a sua avaliação daquela etapa específica, juntamente com correções e com sugestões de melhoria. A avaliação final do aluno foi composta grande parte por estes acompanhamentos parciais, tendo um grande peso na nota final de cada estudante.

De forma geral, foi natural a construção de uma metodologia avaliativa que levasse em conta o processo de ensino-aprendizagem como um percurso do aluno. Por trazer os aspectos educativos da pesquisa científica para a sala de aula, sendo estes fortemente baseados em etapas metodológicas definidas, foi possível observar de forma mais clara a evolução do discente ao longo do processo.

4.6 Dificuldades Encontradas

Apesar dos inúmeros benefícios discutidos anteriormente sobre a aplicação da metodologia de Ensino com Pesquisa, algumas outras questões importantes devem ser levadas em consideração. O desenvolvimento desta estratégia em uma turma com muitos alunos torna o acompanhamento por parte do docente mais complexo. Por exigir tarefas de orientação intensas por parte do professor, cada minuto de sala de aula deve ser aproveitado ao máximo. Desta forma, seguir o planejamento adotado é essencial para que se dê uma boa orientação aos alunos.

Outro ponto importante está relacionado à maturidade dos alunos para o entendimento das etapas metodológicas da pesquisa científica. Este aspecto foi constatado no campo prático deste trabalho, visto que o alunos, em sua grande maioria, eram egressos do ensino fundamental com faixa etária entre 14 e 16 anos. Tornou-se desafiador a tarefa de desmistificar a pesquisa científica perante este perfil de turma. Contudo, observou-se alguns resultados extremamente satisfatórios, pois alunos souberam se expressar organizando seus relatos de acordo etapas metodológicas da pesquisa. Entretanto, alguns outros alunos tiveram bastante dificuldade em estabelecer com clareza tais etapas, o que foi observado de forma mais latente no momento das exposições dos resultados dos projetos.

5. Considerações Finais

Este artigo apresentou um relato das experiências da aplicação da metodologia de Ensino com Pesquisa como forma de aprimoramento da prática docente. Como campo prático, a metodologia foi aplicada na turma de ingressantes do Curso Técnico em Informática Integrado. Foram criados dois projetos de pesquisa os quais embasaram todas as atividades desenvolvidas em sala de aula. Estes projetos foram criados a partir de reflexões realizadas sobre o Plano de Ensino da disciplina, criando percursos investigativos que deveriam ser percorrido pelos alunos, fomentando a (re)construção dos conhecimentos a serem trabalhados. No geral, a experiência se mostrou positiva em muitos aspectos, apresentando-se como uma boa alternativa para os docentes como metodologia de ensino. Como principais benefícios, observou-se: (1) o aumento da participação dos alunos, (2) uma melhor organização das atividades e um melhor entendimento por parte dos alunos de como os conteúdos se relacionam, (3) a atuação do professor como orientador das atividades, (4) a valorização do percurso dos alunos ao longo das etapas de pesquisa desenvolvido ao longo das aulas, resultando em um processo avaliativo que leve em conta, de maneira mais clara, o caminho percorrido como um todo. Desta forma, grande parte dos benefícios previsto na fundamentação da

teoria de Ensino com Pesquisa foram verificados ao longo da experiência de aplicação. Juntamente com os aspectos positivos, alguns pontos precisam ser levados em consideração: a aplicação da metodologia em turmas com muitos alunos e a pouca maturidade dos alunos podem prejudicar o bom andamento do processo de ensino-aprendizagem.

Como trabalhos futuros, a aplicação da metodologia de Ensino com Pesquisa no contexto de outras disciplinas da área de computação está planejada. Cada componente curricular em específico da área profissional abre portas para inúmeras potencialidades de proposições de percursos investigativos aos estudantes. Além disso, a integração destes projetos de pesquisa com temáticas de outras áreas do curso apresenta grandes possibilidades para contribuir com a formação de indivíduos de maneira cada vez mais integral e completa.

Agradecimento

Agradecimento ao Instituto Federal do Rio Grande do Sul pelo apoio financeiro para a participação e apresentação deste trabalho na conferência.

Referências

ANDRADE, L. M.; SILVA, F. Tecnologias da Informação e Comunicação: As Influências das Novas Tecnologias perante a Sociedade. **Anais do 4º Seminário Nacional O Profesor e a Leitura do Jornal**, 2008.

BECKER, F.; MARQUES, T. B. I. (Orgs.), **Ser Professor é Ser Pesquisador**, Porto Alegre: Mediação, 2007, 136p.

BRANDÃO, J.; NEVES, J. Aplicação da metodologia ativa "Peer Instruction" em um curso técnico em informática. **Anais do IX Workshop de pós-graduação e pesquisa do centro Paula e Souza**. 2014.

DEMO, P. **Pesquisa: princípios científico e educativo.** 2. ed. São Paulo: Cortez: Autores Associados, 1991.

DEMO, P. **Educar pela Pesquisa**. Campinas: Autores Associados, 2011.

LIMA, M. Ensino com Pesquisa: uma revolução silenciosa. São Paulo: M. Lima, 2000.

LIMA, S.; DA SILVA, E.; ALVES, V.; CASTANHO, C.; ESPINDOLA, P.; BACHINSKI, R. Aplicação de uma Metodologia Ativa para o Ensino de Lógica de Programação. **Anais do Encontro Anual de Tecnologia da Informação e STIN - Simpósio de tecnologia da Informação da Região Noroeste do RS**. p. 209-212. 2016.

LUCKESI, Cipriano Carlos. **Avaliação da aprendizagem escolar**: estudos e proposições. 22.ed. São Paulo: Cortez, 2011.

POZO, J.; CRESPO, M. A aprendizagem e o ensino de ciências: do conhecimento cotidiano ao conhecimento científico. 5 ed. Porto Alegre: Artmed, 2009.

SANTOS, R. P. ; COSTA, H. A. X. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. **Anais do INFOCOMP – JOURNAL OF COMPUTER SCIENCE**, Lavras/MG – Brasil, v. 5, n. 1, p. 41-50, 2006.

Uma Experiência sobre a Aplicação de Aprendizagem Baseada em Projetos com Revisão por Pares no Ensino de Gestão de Sistemas de Informação

Leo Natan Paschoal[1], **Simone do Rocio Senger de Souza**[1]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC) – Universidade de São Paulo (USP)
Caixa Postal 668 – 13.566-970 – São Carlos – SP – Brasil

paschoalln@usp.br, srocio@icmc.usp.br

Abstract. *Management of Information Systems is a discipline that requires an intensification of intellectual efforts to be taught and learned, because it combines a technical area (Computing) and a social area (Management). In this sense, educational approaches that allow the development of these skills are necessary. This paper presents an experiment report on the use of Project Based Learning and Peer Review during the teaching of this discipline. Therefore, projects that simulate authentic problems that require "real world" solutions have been developed, and an environment that facilitates the development of skills such as autonomy, collaboration and critical thinking has been provided. To evaluate the results, three questionnaires were applied to the students. These questionnaires evaluated the involvement of the students during the practical projects development and the opinion of the students about the used approaches. The results indicate that the used approaches are promising and appropriate to teaching of Management of Information Systems.*

Resumo. *Gestão de Sistemas de Informação é uma disciplina que requer uma intensificação dos esforços intelectuais a serem ensinados e aprendidos, pois combina uma área técnica (Computação) e uma área social (Gestão). Nesse sentido, abordagens educacionais que permitam o desenvolvimento dessas habilidades são necessárias. Este artigo apresenta um relato experimental sobre o uso da Aprendizagem Baseada em Projetos e Revisão por Pares durante o ensino desta disciplina. Portanto, projetos que simulam problemas autênticos que exigem soluções do "mundo real" foram desenvolvidos, e um ambiente que facilita o desenvolvimento de habilidades como autonomia, colaboração e pensamento crítico foi fornecido. Para avaliar os resultados, três questionários foram aplicados aos alunos. Esses questionários avaliaram o envolvimento dos alunos durante o desenvolvimento de projetos práticos e a opinião dos alunos sobre as abordagens utilizadas. Os resultados indicam que as abordagens utilizadas são promissoras e adequadas ao ensino de Gestão de Sistemas de Informação.*

1. Introdução

O currículo de referência da Sociedade Brasileira da Computação (SBC) menciona que egressos de um curso de Sistemas de Informação devem possuir algumas habilidades, especialmente associadas a resolução de problemas autênticos que fazem parte do "mundo

real" [Zorzo et al. 2017]. Esses problemas são provenientes de um contexto social ou organizacional e podem ter relação com a identificação de requisitos, desenvolvimento, evolução e/ou administração de Sistemas de Informação. Dentre os componentes curriculares da formação de um Bacharel em Sistemas de Informação, um tópico importante é a Gestão de Sistemas de Informação.

A relevância da Gestão de Sistemas de Informação é observada ao passo que pelo currículo de referência da SBC esse tópico pode ser considerado em um curso de Sistemas de Informação como um eixo temático de formação (Gestão de Sistemas de Informação e da Tecnologia da Informação). O objetivo principal deste eixo é capacitar o aluno em algumas competências tais como: exercer gerência sobre os Sistemas de Informação; aplicar conceitos, vistos durante o curso, adequados à gestão de governança de informação e tecnologia da informação; dentre outros. Além de ser um dos sete eixos temáticos que o currículo da SBC estabelece, o tópico pode ser tratado como um componente curricular do eixo de formação "Desenvolvimento de Software para Sistemas de Informação". Esse componente curricular visa capacitar os estudantes em "avaliar as necessidades de informatizar sistemas, articular visões individuais e organizacionais e apreciar oportunidades de melhorias e ou mudanças em processos, com o uso ou evolução do software" [Zorzo et al. 2017].

Apesar da relevância, ensinar Gestão de Sistemas de Informação é um desafio, pois envolve duas áreas distintas: a Computação, uma área técnica; e a Gestão, uma área social [Devece et al. 2011]. Por envolver essas duas áreas, abordagens de ensino que oportunizam o desenvolvimento de habilidades técnicas e sociais, necessárias para o desempenho das funções que os futuros profissionais irão desempenhar, precisam ser consideradas. [Devece et al. 2011] mencionam sobre a importância de considerar abordagens com foco em promover condições para os estudantes poderem participar ativamente do processo de aprendizagem, como estudos de caso, aprendizagem baseada em problemas e aprendizagem baseada em projetos. Estudo de caso é uma abordagem bastante utilizada na área de Gestão [Cameron et al. 2012]. Em contrapartida, aprendizagem baseada em problemas e aprendizagem baseada em projetos são bastante populares na Computação, especialmente no contexto de desenvolvimento de software [Souza et al. 2016, Fioravanti et al. 2018].

Os currículos atuais de Computação, Nacional e Internacional, representados pela SBC e ACM (*Association for Computing Machinery*), respectivamente, mencionam a importância dos egressos de cursos de Computação desenvolverem não somente habilidades técnicas, mas também habilidades sociais, as quais são necessárias em atividades de gestão de projetos e de sistemas. Nesse sentido, abordagens ativas de aprendizagem podem ser alternativas viáveis ao ensino de disciplinas como Gestão de Sistemas de Informação. Uma abordagem que visa exercitar o desenvolvimento de habilidades não técnicas como autonomia, colaboração, cooperação e pensamento crítico é a Revisão por Pares (*Peer Review*) [Kern et al. 2006].

Revisão por Pares é uma abordagem amplamente utilizada no âmbito acadêmico para avaliação de artigos científicos. Ela serve como um meio para o controle da qualidade de artigos científicos que são submetidos a *journals* e conferências. Seu funcionamento pode ser compreendido da seguinte maneira: quando um artigo é submetido por um pesquisador ou por um grupo de pesquisadores, esse artigo é revisado (avaliado) por

um grupo de pesquisadores que são especialistas na área. Nesse sentido, a qualidade do artigo é controlada a partir da revisão do mesmo pelos pares de seus pares. No âmbito educacional, essa abordagem vem sendo adaptada, porque pode oferecer diferentes benefícios para os sujeitos que estão envolvidos nos processos de ensino e aprendizagem [Jaime et al. 2016]. Em estudos em que Revisão por Pares é usada como abordagem de apoio ao ensino, os autores adaptam a abordagem para satisfazer os objetivos de aprendizagem que são propostos para uma disciplina e/ou curso [Kern et al. 2003].

Uma característica que se faz presente no uso de Revisão por Pares é a formação de grupos para desenvolvimento de trabalhos. Desse modo, no contexto de uma disciplina, os estudantes são divididos em alguns grupos. Esses grupos são formados com o objetivo de realizar uma atividade de maneira colaborativa, podendo ser uma atividade que resulte em um artigo científico, um relatório ou um projeto. Após a realização da atividade, os trabalhos produzidos são avaliados por estudantes que constituem um outro grupo. Desse modo, cada grupo revisor analisa o trabalho de um outro grupo, seguindo os critérios de revisão que são estipulados pelo professor. Com o uso dessa abordagem, o professor consegue promover a interação e estimular o desenvolvimento do senso crítico dos alunos, fazendo com que os estudantes contribuam para o aprimoramento da qualidade do trabalho desenvolvido por seus pares [Jaime et al. 2016].

Outra abordagem que pode apoiar o aprendizado de estudantes de Gestão de Sistemas de Informação é a aprendizagem baseada em projetos (PBL - *Project Based Learning*), visto que, essa abordagem oferece subsídios a realização, por parte dos alunos, de atividades que normalmente envolvem projetos que são oriundos do "mundo real". Essa abordagem caracteriza-se pelos seguintes aspectos: os trabalhos são planejados de modo que sejam focados em problemas que levam os alunos a aplicar conceitos e princípios teóricos visto em aula; por meio dos projetos, os alunos são incentivados a desenvolver e/ou exercitar algumas habilidades técnicas e não técnicas; os alunos são envolvidos em atividades autênticas e reais, que normalmente incluem a criação de um produto ou solução, envolvendo tarefas interdisciplinares; a condução do trabalho pode levar semanas e até mesmo meses [Acosta 2016].

Estudos sobre o uso de PBL revelam que a mesma oferece diversos benefícios aos estudantes, promovendo: motivação, condições para os estudantes serem responsáveis pela própria aprendizagem, engajamento na atividade, situações para o aluno experimentar o desenvolvimento de projetos em equipes constituídas por pessoas com diferentes perfis [Jaime et al. 2016, Fioravanti et al. 2018]. Além disso, os currículos de referência de Computação estão recomendando que projetos do "mundo real" sejam explorados no contexto acadêmico, pois é uma maneira de prover condições para os estudantes se prepararem para se inserirem no mercado de trabalho.

Em razão dos benefícios que são oferecidos aos alunos quando as abordagens PBL e Revisão por Pares são utilizadas, [Boubouka and Papanikolaou 2013] mencionam sobre a possibilidade de incorporar Revisão por Pares em PBL. Esse pensamento é posto em prática na disciplina de Gerenciamento de Projetos, conforme relatado no estudo de [Jaime et al. 2016]. Com isso, estudantes formam grupos, desenvolvem os projetos, e após os mesmos serem desenvolvidos, os projetos são revisados por seus pares. Essa mistura de abordagem é capaz de oferecer condições para os estudantes alcançarem um alto nível de sucesso na disciplina [Jaime et al. 2016]. Uma experiência semelhante

foi realizada no contexto de ensino de Algoritmos e Estruturas de dados, relatada por [Machanick 2005], em que os alunos analisam os projetos uns dos outros.

Diante do que foi exposto, este artigo relata uma experiência conduzida no contexto de uma disciplina de Gestão de Sistemas de Informação, oferecida aos alunos do curso de Bacharelado em Sistemas de Informação, que teve como objetivo principal oferecer condições para que os alunos desenvolvessem diferentes habilidades, não apenas técnicas, durante o aprendizado da disciplina. Essa experiência constitui-se pelo uso de duas diferentes abordagem de apoio ao ensino: PBL e Revisão por Pares. A primeira é usada para instigar os alunos a trabalharem em equipe com projetos do "mundo real", desafiando-os a trabalharem com projetos interdisciplinares que necessitam de apoio colaborativo. Ela também oferece oportunidades para os alunos não somente aprender a teoria da Gestão de Sistemas de Informação, mas aplicar os conceitos vistos durante as aulas em projetos próximos aos que irão desenvolver futuramente no mercado de trabalho. A segunda por sua vez, é utilizada principalmente para oferecer subsídios ao desenvolvimento do senso crítico entre os alunos, e aumentar a qualidade dos projetos desenvolvidos.

Este artigo está estruturado da seguinte maneira. Na seção 2 é descrita a estratégia educacional que foi adotada na disciplina de Gestão de Sistemas de Informação. Na seção 3 são discutidos os resultados obtidos. Por fim, na seção 4 são apresentadas as conclusões do artigo e perspectivas de trabalhos futuros.

2. Descrição da Experiência

A experiência da aplicação das abordagens ocorreu na disciplina de Gestão de Sistemas de Informação, oferecida pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo. Essa disciplina é um componente curricular obrigatório para alunos do curso de Bacharelado em Sistemas de Informação (BSI) e optativo para os cursos de Bacharelado em Ciências de Computação (BCC) e Engenharia de Computação (EngComp). A disciplina é oferecida no sexto semestre para alunos do BSI e pode ser cursada no quinto ano pelos alunos do BCC e EngComp. Para o contexto dessa experiência, a disciplina foi oferecida no 2º semestre de 2017, possuindo quatro créditos-aula (noventa horas-aula no semestre, divididas em duas aulas semanais), e foi constituída por 31 alunos do BSI, 9 do BCC e 3 do EngComp. O objetivo da disciplina é oferecer uma visão geral sobre as principais aplicações de Sistemas de Informação nas organizações atuais, assim como, os princípios de Governança e Gestão de Serviços de Tecnologia de Informação, focando principalmente na prática que envolve a análise e o dimensionamento de soluções de Sistemas de Informação adequadas ao contexto organizacional.

A experiência foi planejada visando consolidar os conhecimentos que são transpassados no decorrer da disciplina. Para tanto, a disciplina foi planejada de modo que o conteúdo teórico fosse ministrado antes do início do desenvolvimento dos projetos. Assim, ela foi dividida em duas parte. A primeira foi constituída por aulas expositivas sobre o conteúdo, exercícios (intra e extra-classe), estudos de caso, e discussões sobre os exercícios e estudos de casos, visando a participação dos alunos na construção dos conhecimentos teóricos. A segunda parte consistiu no uso da abordagem PBL em conjunto com Revisão por Pares. O foco deste artigo é relatar a experiência realizada na segunda etapa da disciplina, em que os alunos já haviam estudado o conteúdo teórico da disciplina necessário para o desenvolvimento dos projetos.

Para apoiar a condução da atividade, a disciplina contou com a colaboração de um aluno de mestrado, que desempenhou atividades didáticas em conjunto com o professor da disciplina, na função de estágio docente. Nesse sentido, a estratégia pedagógica que engloba as duas abordagens mencionadas foi planejada da seguinte forma. Inicialmente, o professor da disciplina preparou, em conjunto com o estagiário, três problemas fictícios, que simulam problemas reais. Esses problemas necessitam de soluções que obrigatoriamente envolvem o desenvolvimento de uma solução em formato de projeto. Os problemas foram derivados de outras disciplinas do curso de Sistemas de Informação: em uma disciplina os alunos propuseram esses problemas e em outra foram usados de maneira similar ao que foi feito nesta experiência. Após a definição dos projetos, o professor e o estagiário da disciplina estabeleceram algumas regras para a formação de grupos: (i) os 43 alunos matriculados na disciplina foram convidados a se dividir em 12 grupos; (ii) cada grupo poderia ser constituído por pelo menos 3 e no máximo 4 alunos; (iii) os alunos poderiam definir os grupos. Após a divisão dos grupos, os projetos foram distribuídos randomicamente entre os mesmos, de modo que cada projeto fosse desenvolvido por quatro equipes.

A descrição dos projetos utilizados na disciplina (3 projetos no total) podem ser encontrados no pacote de laboratório que foi preparado para este artigo, disponível em: <https://goo.gl/24CS9t>. Em linhas gerais, os grupos teriam que analisar os projetos e propor soluções de Sistemas de Informação adequadas ao contexto da organização em questão. No pacote também foi disponibilizado uma especificação que contém informações sobre o que os alunos deveriam entregar durante o desenvolvimento do projeto. Salienta-se que foi definido que os alunos teriam que fazer duas entregas referentes aos seus projetos, pois a atividade seria desenvolvida em duas fases. Na primeira fase, cada grupo deveria preparar propostas de soluções para automatização do processo das organizações, levando em consideração as especificações. Nessa fase, os alunos deveriam utilizar os conhecimentos adquiridos na disciplina ou em outras relacionadas ao assunto, visando enriquecer as propostas dos projetos. No final desta fase, os alunos deveriam entregar (via ambiente Moodle) e apresentar as soluções elaboradas pelos grupos para todos os colegas da disciplina, ao professor e ao estagiário.

Considerando que 12 grupos foram formados, e cada 4 grupos trabalhou com um projeto diferente, foi determinado que três aulas seriam destinadas a apresentações das soluções propostas pelos grupos, um tema por aula. Salienta-se que cada grupo recebeu a descrição do tema do projeto uma semana antecedente ao prazo para entregar a proposta de solução do projeto. Nesse sentido, cada grupo teve uma semana para desenvolver os projetos. Portanto, no momento em que os projetos foram distribuídos randomicamente entre os grupos, os grupos não tiveram acesso a especificação.

Assim, em uma aula 4 grupos apresentaram suas soluções de projeto para o mesmo tema. Os demais grupos (8) assumiram o papel de revisores dos projetos. Esses grupos receberam as propostas dos grupos para avaliar antes da aula de apresentação. Durante as apresentações, a dinâmica dos seis chapéus foi utilizada para melhorar a participação dos avaliadores no processo.

A dinâmica dos seis chapéus desenvolvida originalmente por Edward De Bono é uma estratégia que visa contribuir com a produção do pensamento crítico, colaboração, comunicação e criatividade [Kwasnik 2014]. Ela fornece diferentes visões para avaliar uma dada situação sobre o ponto de vista único de cada pessoa

[Scott and Sewchurran 2008]. Utiliza o princípio por trás do pensamento paralelo que prevê que todos os participantes de uma reunião se concentram e pensam no mesmo assunto ao mesmo tempo [Scott and Sewchurran 2008]. De acordo com [Kwasnik 2014] cada um dos seis chapéus representa um espectro do pensamento humano, por exemplo, chapéu azul identifica os benefícios; chapéu vermelho identifica as limitações. Detalhes sobre a dinâmica estão disponíveis no pacote de laboratório.

Para o contexto desta experiência, a dinâmica dos seis chapéus serviu como uma mecanismo de apoio à Revisão por Pares, dado que ao invés de oferecer um formulário para os estudantes preencherem, os mesmos teriam que expressar suas opiniões durante a aula e divulgar a opinião em um fórum de discussão disponibilizado no ambiente virtual de aprendizagem Moodle. Essas opiniões serviram de base para as melhorias das propostas para a segunda fase do projeto. Cada grupo foi sorteado com um chapéu de uma cor e precisou apontar suas percepções sobre o trabalho desenvolvido pelos colegas, considerando a cor do seu chapéu. Ao final de cada apresentação, um avaliador por grupo (de acordo com o chapéu) realizava seus questionamentos e ponderações. A cada apresentação, o chapéu era trocado entre os membros do grupo, de modo que todos que estavam assistindo pudessem colaborar. Durante essa intervenção, o professor e o estagiário ocuparam o papel de mediadores.

Após a avaliação dos alunos em relação às apresentações dos projetos, os alunos avaliadores elegeram a melhor proposta apresentada. Para tanto, deveriam considerar a qualidade de informações presentes no documento entregue e a qualidade da apresentação. Não foi elaborado especificações maiores a respeito de pontos que os alunos deveriam considerar na avaliação. Apenas foi especificado que o documento deveria conter pelo menos as seguintes informações: descrição sobre a empresa, solução proposta (com aspectos técnicos, escopo, melhoria de processo), parâmetro da solução (custo de aquisição, tempo de implantação, treinamento dos profissionais que atuam na organização, infraestrutura necessária, custo operacional), impacto organizacional, impacto social e político. Foi disponibilizado uma enquete no Moodle para que os alunos pudessem eleger a melhor proposta.

Além da avaliação dos seus pares, os grupos receberam um *feedback* com sugestões de melhorias, emitido pelo professor da disciplina em conjunto com o estagiário. Com as revisões e sugestões de melhorias, a segunda fase foi estabelecida para a melhoria das propostas e reapresentação das mesmas. Semelhantemente a fase anterior, cada 4 grupos responsável por um projeto teria uma aula para fazer a apresentação. Adicionalmente, foi solicitado aos grupos que adicionassem ao documento elaborado os diagramas utilizando BPMN (Business Process Model and Notation) para os processos de negócios, em especial a modelagem do estado original da organização (*as is*) e a modelagem do estado futuro (*to be*). Após a entrega do documento e apresentação, os grupos foram novamente avaliados pelos seus colegas da disciplina.

Ao final da experiência os alunos tiveram que responder três questionários. O primeiro questionário foi criado para cada aluno fazer uma autoavaliação sobre a sua participação na atividade. O segundo questionário foi desenvolvido para os alunos avaliarem os seus colegas de equipe, assim, cada aluno avaliou o nível de participação e colaboração dos membros de seus grupos (de maneira anônima). Esses questionários foram construídos para apoiar a avaliação dos trabalhos que foram conduzidos e verificar

o envolvimento dos alunos no desenvolvimento dos projetos. Por fim, o terceiro questionário teve como objetivo coletar informações sobre a percepção dos alunos quanto às abordagens utilizadas na experiência. Os questionários estão disponíveis no pacote de laboratório. Os resultados serão apresentados na próxima seção.

3. Resultados Obtidos

Esta seção apresenta os resultados referentes a essa experiência de ensino, incluindo o envolvimento dos alunos durante as duas fases que englobam o desenvolvimento do projeto, sobre as melhores soluções propostas para os projetos, e sobre a percepção dos mesmos quanto a experiência vivenciada nessa etapa da disciplina.

Para averiguar o envolvimento dos alunos foram utilizados os dois questionário específicos para o envolvimento. Ambos os questionários possuíam as mesmas questões. A única diferença é que um possibilita a autoavaliação e o outro possibilita a avaliação dos pares. Cada questionário possuía 1 questão destinada aos alunos atribuírem uma pontuação para eles mesmos e para seus pares. Sobre a percepção dos alunos na experiência, um questionário foi enviado por *e-mail* para todos os alunos matriculados na disciplina. Apesar da disciplina ter 43 alunos, apenas 40 alunos responderam ao questionário.

Os resultados sobre o envolvimento dos alunos estão resumidos e apresentados na Tabela 1. Nela é possível observar as médias das pontuações por aluno e os desvios-padrões, que foram calculados por meio das pontuações que cada aluno recebeu de seus pares em conjunto com a pontuação que cada aluno se atribuiu. Essas pontuações auxiliaram na identificação de alunos que tiveram um maior comprometimento com a equipe e consequentemente um envolvimento maior no desenvolvimento dos projetos. Por exemplo, o aluno número 12, integrante do Grupo 3, ficou com uma média de 3,75 pontos, um valor baixo se considerado o valor máximo que poderia ser obtido.

Tabela 1. Envolvimento dos alunos

	Grupo 1				Grupo 2				Grupo 3				Grupo 4		
Aluno	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Média das pontuações	9,00	9,25	9,25	4,25	7,00	8,00	7,00	8,67	7,00	4,67	9,00	3,75	5,00	5,50	8,00
Desvio padrão	0,82	0,96	0,50	2,22	0,00	1,00	0,00	0,58	3,00	4,04	1,00	2,63	4,36	2,12	2,83

	Grupo 5			Grupo 6				Grupo 7				Grupo 8		
Aluno	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
Média das pontuações	10,00	9,67	6,67	9,25	8,75	9,00	9,00	9,00	9,50	9,25	7,50	8,00	9,67	8,67
Desvio padrão	0,00	0,58	0,58	0,50	0,96	0,82	0,82	0,00	0,58	0,50	0,58	1,00	0,58	0,58

	Grupo 9			Grupo 10				Grupo 11			Grupo 12			
Aluno	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**
Média das pontuações	9,33	9,67	9,33	8,25	8,75	9,25	7,75	7,33	7,33	10,00	8,75	9,25	9,75	9,00
Desvio padrão	1,15	0,58	0,58	0,96	0,50	0,50	1,89	0,58	1,53	0,00	0,96	0,96	0,50	1,41

Na mesma tabela também é possível observar os desvios-padrões, que auxiliam na identificação de variações entre as pontuações obtidas pelos alunos. Quanto mais próximo de zero, menor será a diferença entre as pontuações recebidas pelo aluno de seus pares em conjunto com a auto-avaliação. Observando o valor de desvio padrão do aluno número 39, integrante do Grupo 11, nota-se que o mesmo é zero, o que significa que ele se auto avaliou com 10 pontos e toda a sua equipe também atribuiu 10 pontos para ele. Em outras ocasiões é possível observar algumas semelhanças com o que foi mencionado, por exemplo, no Grupo 2, os alunos número 5 e número 7 obtiveram os valores dos desvios-padrões equivalentes a zero e a média de ambos os alunos foi 7, o que demonstra que

ambos os alunos se autoavaliaram com 7 pontos e os seus pares também os avaliaram com 7 pontos. Nota-se por meio da avaliação a participação efetiva dos alunos no trabalho. Os alunos que se comprometeram foram sinceros e apontaram os alunos que foram passivos na experiência, isto é, que não participaram ativamente no desenvolvimento das soluções.

Conforme mencionado no decorrer deste artigo, os alunos tiveram que escolher as melhores soluções para os projetos. A Figura 1 apresenta os dados das votações e, conforme é possível observar, o grupo que propôs solução para o Projeto 1 melhor votado foi o Grupo 1. O grupo que apresentou a melhor solução para o Projeto 2 foi o Grupo 8. Por fim, o grupo que apresentou a melhor proposta de solução para o Projeto 3 foi o Grupo 12. Os grupos que obtiveram uma melhor votação em relação aos demais foram premiados na disciplina. A ideia da premiação é vista na literatura como positiva, no sentido de trazer elementos relacionados com a motivação, dado que pode fazer com que os alunos fiquem motivados em desenvolver projetos com alta qualidade. Algo semelhante é relatado por [Souza et al. 2016] ao utilizarem a premiação para motivar alunos a desenvolver projetos de software com qualidade. Salienta-se que os alunos foram notificados no início da experiência que as melhores propostas de solução para os projetos seriam premiadas.

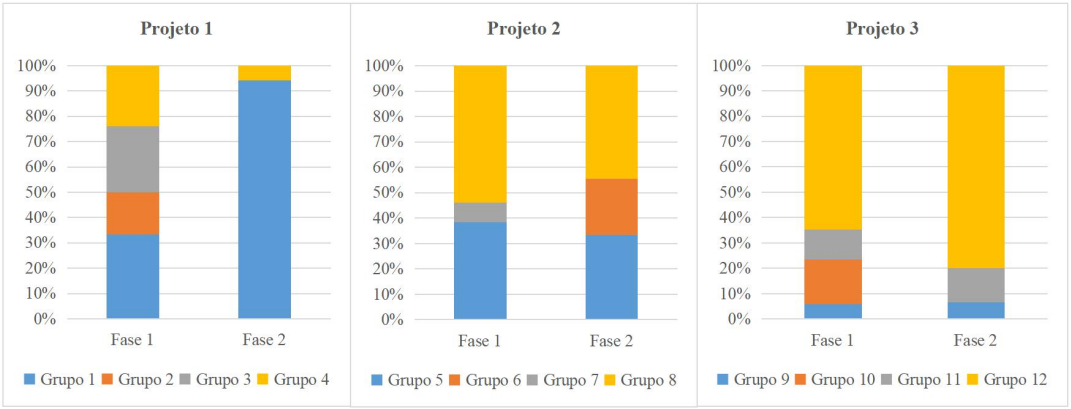


Figura 1. Escolha das melhores propostas de soluções

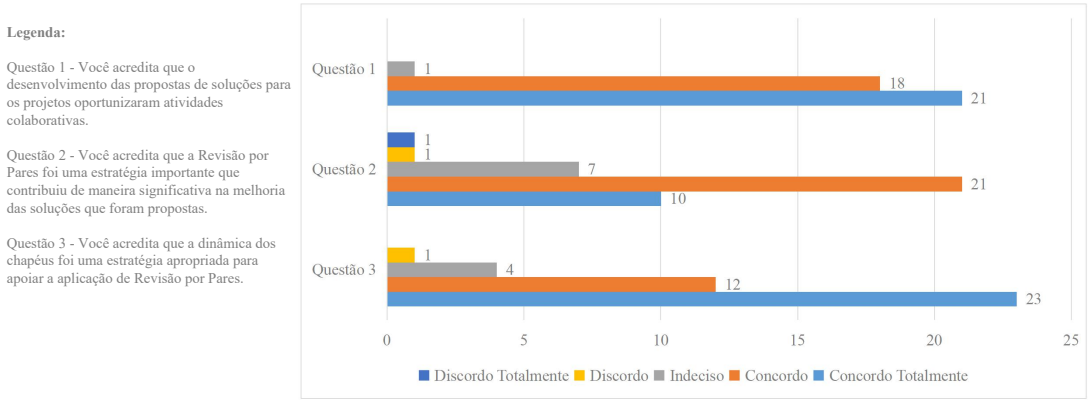


Figura 2. Percepção dos alunos sobre as abordagens

Em relação à percepção dos estudantes sobre a estratégia pedagógica adotada, um terceiro questionário contendo 3 questões fechadas foi disponibilizado. A primeira

questão teve o intuito de saber se o desenvolvimento do projeto em equipe oportunizou um cenário de atividades que deveriam ser realizadas de forma colaborativa. A maioria dos participantes concordaram com a afirmativa. Aos serem questionados sobre a relevância do uso de Revisão por Pares para melhoria da qualidade dos projetos apresentados, na segunda questão, as respostas oscilaram principalmente entre as opções concordo totalmente, concordo e indeciso. Por fim, quando questionados sobre a importância da dinâmica dos chapéus para apoiar a Revisão por Pares os alunos apontaram que aprovam o uso da dinâmica. A Figura 2 apresenta os resultados. A partir desses dados é possível reconhecer que os alunos aprovaram o uso das abordagens, pois apoiaram a visão prática dos conceitos vistos em aula.

4. Considerações Finais

No decorrer deste artigo uma experiência com alunos de Gestão de Sistemas de Informação foi apresentada. Os motivos que levaram o uso das abordagens PBL com Revisão por Pares relacionam-se a capacidade das mesmas oferecerem oportunidades para os alunos trabalharem com projetos que se assemelham ao "mundo real", que poderão ser vistos pelos alunos no mercado de trabalho, com a possibilidade de utilizar a opinião de seus pares visando a melhoria da qualidade do trabalho realizado. A visão mais prática do conteúdo é altamente recomendado pelos currículos de referência da ACM e SBC. Além disso, a experiência buscou contribuir com o desenvolvimento de habilidades não técnicas dos alunos, visto que, as abordagens oferecem subsídios ao trabalho colaborativo, a competição saudável entre os grupos, a importância da comunicação entre os integrantes dos grupos, e a relevância da interpretação dos documentos. Por fim, com o desenvolvimento dos projetos e uso das abordagens esperava-se que os alunos pudessem ter um entendimento em um nível mais prático sobre análise e o dimensionamento de soluções de Sistemas de Informação que são apropriados para as organizações. A partir dos dados apresentados, acredita-se que os objetivos foram atingidos.

Os resultados encontrados com a experiência não visam demonstrar que as abordagens possam ser consideradas as melhores a serem aplicadas para o contexto apresentado, visto que, não foi feito um estudo experimental para averiguar tal condição. Entretanto, eles podem ser utilizados como argumento de que a percepção dos alunos é bastante favorável, podendo inclusive ser vista como uma alternativa para o modelo tradicional de ensino, em que a ênfase do ensino não é a visão mais prática do conteúdo. Em contrapartida, conforme mencionado no decorrer do artigo, o emprego das abordagens criou condições para capacitar os alunos a aprender o conteúdo com um nível maior de profundidade. Assim, espera-se que essa experiência seja replicada por outros professores da área, de forma a contribuir com a formação dos alunos.

5. Agradecimentos

Os autores gostariam de agradecer ao CNPq, CAPES e a Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP – Processo 2017/10941-8) pelo apoio financeiro.

Referências

Acosta, O. C. (2016). *Recomendação de conteúdo em um ambiente colaborativo de aprendizagem baseada em projetos*. PhD thesis, Centro de Estudos Interdisciplina-

res em Novas Tecnologias da Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brasil.

Boubouka, M. and Papanikolaou, K. A. (2013). Alternative assessment methods in technology enhanced project-based learning. *International Journal of Learning Technology*, 8(3):263–296.

Cameron, A.-F., Trudel, M.-C., Titah, R., and Léger, P.-M. (2012). The live teaching case: a new is method and its application. *Journal of Information Technology Education: Research*, 11:27–42.

Devece, C., Lapiedra, R., and Gil, I. (2011). Teaching information systems in business management studies: Basic competencies to achieve and methodologies assessment. In *5th International Multi-Conference on Society, Cybernetics and Informatics*, pages 114–119.

Fioravanti, M. L., Sena, B., Paschoal, L. N., Silva, L. R., Allian, A. P., Nakagawa, E. Y., Souza, S. R., Isotani, S., and Barbosa, E. F. (2018). Integrating project based learning and project management for software engineering teaching: An experience report. In *49th ACM Technical Symposium on Computer Science Education*, pages 806–811.

Jaime, A., Blanco, J. M., Domínguez, C., Sánchez, A., Heras, J., and Usandizaga, I. (2016). Spiral and project-based learning with peer assessment in a computer science project management course. *Journal of Science Education and Technology*, 25(3):439–449.

Kern, V., dos Santos Pacheco, R., Saraiva, L., and Pernigotti, J. (2006). *Peer review in computer science: Toward a regular, large scale educational approach*, chapter 3, pages 45–65. Information Science Publishing, EUA.

Kern, V. M., Saraiva, L. M., and dos Santos Pacheco, R. C. (2003). Peer review in education: Promoting collaboration, written expression, critical thinking, and professional responsibility. *Education and Information Technologies*, 8(1):37–46.

Kwasnik, M. (2014). Nature of creativity in computer science education. designing innovative workshops for cs students. In *International Conference on Multimedia, Interaction, Design and Innovation*, pages 8:1–8:7.

Machanick, P. (2005). Peer assessment for action learning of data structures and algorithms. In *7th Australasian Conference on Computing Education*, pages 73–82.

Scott, E. and Sewchurran, K. (2008). Reflection-in-action: Using experience to reconstruct meaning in a learning environment. In *International Conference on Computer Science and Software Engineering*, pages 81–86.

Souza, S. R. S., Oliveira, B. H., Grillo, F., and Cico, C. (2016). Construção de plataformas digitais durante o ensino de engenharia de software: um relato de experiência. In *IX Fórum de Educação em Engenharia de Software (FEES 2016)*, pages 13–22.

Zorzo, A. F., Nunes, D., Matos, E. S., Steinmacher, I., Leite, J. C., Araujo, R., Correia, R. C. M., and Martins, S. (2017). Referenciais de formação para os cursos de graduação em computação 2017. Technical report, Porto Alegre, Brasil.

Laboratório remoto de robótica para o ensino de programação com suporte à análise, codificação e teste

Maísa Soares dos Santos Lopes[1], Rodrigo Silva Lima[1], João Vitor Ferraz[1], Matheus Lima Viana[1], Roque Mendes Prado Trindade[1], Alzira Ferreira da Silva[1], Antônio Cezar de Castro Lima[2]

[1]Departamento de Ciências Exatas e Tecnológicas – Universidade Estadual do Sudoeste da Bahia (UESB) – Vitória da Conquista, BA – Brasil

[2]Departamento de Engenharia Elétrica – Universidade Federal da Bahia (UFBA) – Salvador, BA – Brasil

{maisa, roquetrindade}@uesb.edu.br, {rodrig0_lima, matheuscond15}@hotmail.com, {jvitor.ferraz14, arizlas}@gmail.com, acdcl@ufba.br

Abstract. *Remote laboratories are being applied in various educational settings, including in teaching computer programming. However, most of these laboratories do not support the phase of problem analysis. This work presents an educational environment for programming teaching that allows analyzing problems, planning, coding and testing solutions. It uses a virtual learning environment, a remote robotics lab, an online compiling tool, and an online text editor. In the environment, the teacher can define practical programming activities and the student can design a plan, build algorithm, implement it in a programming language and test the solution on a remote mobile robot.*

Resumo. *Os laboratórios remotos estão sendo aplicados em diversos cenários educacionais, entre eles, no ensino de programação de computadores. Entretanto, a maioria desses laboratórios não dá suporte a fase de análise de problemas. Este trabalho apresenta um ambiente educacional voltado ao ensino de programação que permite analisar problemas, planejar, codificar e testar soluções. Ele utiliza um ambiente virtual de aprendizagem, um laboratório remoto de robótica, uma ferramenta de compilação online e um editor de texto online. No ambiente, o professor pode definir atividades práticas de programação e o aluno pode projetar um plano, construir algoritmo, implementá-lo em uma linguagem de programação e testar a solução em um robô móvel remoto.*

1. Introdução

O avanço tecnológico permitiu o surgimento de vários laboratórios remotos que estão sendo aplicados em diversos cenários educacionais [Heradio et al., 2016], [Alkhaldi et al., 2016]. Alguns deles possibilitam o ensino de programação de computadores através da integração com ferramentas de programação. Entretanto, gerar código é apenas uma etapa do processo de aprendizagem de programação. Antes disso, é necessário entender o problema para formular uma solução, produzir um algoritmo e, então, escrever o algoritmo em uma linguagem de programação.

Os estudantes novatos tendem a ignorar as fases de análise e projeto, desenvolvendo programas diretamente no computador usando tentativa e erro para encontrar uma solução e não entendem de fato o que estão fazendo [Tan et al., 2014]. Estas fases são importantes pois ajudam a fixar conceitos e desenvolver habilidades de programação uma vez que os alunos são estimulados a pensar sobre o problema e propor soluções adequadas que devem ser implementadas, testadas, validadas e, se necessário, corrigidas e aprimoradas.

Este trabalho apresenta o AAPC (Ambiente de Análise, Programação e Controle) voltado ao ensino de programação para iniciantes. O ambiente tem como objetivo permitir a aplicação das etapas de análise, codificação e teste nas atividades práticas de programação. O AAPC utiliza um ambiente web de programação e controle de um robô móvel de baixo custo integrado a um editor de texto simples. Dentro do mesmo ambiente o estudante pode desenvolver seus algoritmos, implementa-los e testá-los no laboratório remoto.

O artigo está organizado da seguinte forma: a seção 2 apresenta conceitos relacionados ao ensino de programação e laboratórios remotos; a seção 3 descreve alguns laboratórios remotos de robótica móvel voltados ao ensino de programação; a seção 4 descreve o AAPC e, finalmente, a seção 5 apresenta as considerações finais sobre o trabalho.

2. Fundamentação teórica

2.1. Ensino de programação

O objetivo do ensino de programação é capacitar os alunos a desenvolverem soluções computacionais para resolver problemas do mundo real. Portanto, programação não se restringe ao estudo de linguagem de programação, mas envolve um conjunto de princípios, técnicas e formalismos que visam o desenvolvimento de programas de qualidade [Santos & Costa, 2006]. Segundo Koorsse et al. (2015), para desenvolver programas de computador são necessárias três etapas:

1. Entender o problema para formular uma solução – é necessário estudar a declaração do problema, o conjunto de requisitos e decidir sobre a melhor estratégia de programação a ser usada.
2. Produzir um algoritmo para resolver o problema – é necessário conhecer os princípios e conceitos de programação para selecioná-los e aplicá-los corretamente.
3. Traduzir o algoritmo para uma linguagem de programação – isto exige o conhecimento da sintaxe da linguagem de programação. O código do programa é testado e alterado até que o programa atenda ao conjunto original de requisitos e, assim, resolva o problema.

Os alunos devem, então, aprender o processo de resolução de problemas, desenvolver o raciocínio lógico e aprender linguagem de programação. Este é um processo considerado difícil, especialmente para alunos iniciantes. Gomes et.al (2008) identificam um conjunto de fatores que complicam o aprendizado de programação: os métodos de ensino não são de fato centrado no aluno, as turmas grandes não permitem

um ensino personalizado com feedback e supervisão adequados às necessidades de cada estudante; os conteúdos de programação são dinâmicos e abstratos, além disso, é comum começar ensinar os detalhes sintáticos de uma linguagem de programação antes que os alunos percebam qual a finalidade e utilidade de aprender programar; e os alunos estão habituados com o modelo de estudo baseadas em leituras, memorização de fórmulas e uma certa mecanização de procedimentos, mas programação exige intenso trabalho prático extraclasse, uma verdadeira compreensão dos assuntos e reflexão.

Souza et al. (2016) enfatizam que os alunos tem dificuldade para aprender os conceitos de programação, para aplicar estes conceitos durante a construção de programas e falta motivação para realizar as atividades de programação. A maior parte do tempo das disciplinas é gasta ensinando a sintaxe da linguagem de programação e faltam sessões práticas *hand on* [Husain et al., 2013]. Como forma de amenizar tais problemas e dificuldades, vários trabalhos apontam o uso de robôs no ensino de programação.

Robótica na educação é uma tendência em expansão [Cruz-Martín et al., 2012]. A capacidade dos robôs de explorar e interagir com o mundo real através de sensores e atuadores possibilita sua aplicação em uma série de atividades educativas que ajudam e promovem a aprendizagem [Benitti, 2012].

Apesar das vantagens educacionais, a maioria dos laboratórios de robótica só está disponível para os alunos em períodos restritos. O uso de laboratório remoto possibilita compartilhamento de recurso e garante flexibilidade de tempo e espaço para os usuários.

2.2 Laboratório remoto

A expressão laboratório remoto é utilizada para definir experimento que é conduzido e controlado remotamente através da Internet. Os laboratórios remotos utilizam componentes ou instrumentação reais em um local diferente de onde eles estão sendo manipulados. O usuário acessa e controla o computador do laboratório e pode acionar equipamentos, fazer observações, testar condições e coletar dados do experimento. A instalação de câmeras de vídeo no ambiente do laboratório permite que o usuário acompanhe em tempo real a execução do experimento [Chen et al., 2010], [Ma & Nickerson, 2006].

Os laboratórios remotos modernos são ferramentas de software e hardware que possuem um conjunto típico de componentes [Gomes & Bogosyan, 2009]: o objeto controlado (experimento, instrumento ou maquete experimental); os equipamentos e dispositivos de instrumentação que permitem a aquisição e controle de dados; o computador servidor do laboratório que assegura o controle e monitoramento do experimento; o servidor que faz a ligação, através da Internet, entre os usuários remotos e o servidor do laboratório; o servidor de câmera web; e a aplicação cliente que permite ao usuário acessar o experimento através de um navegador web.

Estudos mostram que este tipo de laboratório é uma ferramenta eficaz na compreensão conceitual de conteúdo [Gomes & Bogosyan, 2009], [Machotka et.al, 2009], [Lang, 2012], [Brinson, 2015].

3. Trabalhos relacionados

Apesar de existirem vários laboratórios remotos de robótica atualmente, não foi encontrado na literatura nenhum com suporte as atividades de análise e implementação de programas. Alguns destes laboratórios permitem que os usuários façam *upload* de código que são executados pelos robôs remotos. Alguns outros fornecem ferramenta de programação onde o estudante pode criar e executar o seu próprio código no ambiente do laboratório.

Aroca et al. (2012) propõem um sistema web onde o usuário pode desenvolver seu próprio programa para controlar o robô. O sistema utiliza um smartphone como computador principal para controlar o robô e como servidor web. Entretanto, não há um controle que permita acompanhar as atividades e desempenho de cada aluno.

Iturrate et al. (2013) mostram a integração de um laboratório remoto robótico com *Serious Games*. Os estudantes podem controlar o robô usando o teclado ou através da linguagem de programação gráfica *Google Blockly*. O código é executado no computador que controla e envia comandos para o robô. O programa não é executado no Arduino do robô, mas no computador.

Park & Lenskiy (2014) apresentam um laboratório remoto de robótica para ensino de programação C. O projeto de baixo custo simula sensores através de uma única câmera complementada com o algoritmo de visão computacional. Entretanto, o artigo não mostra o ambiente onde os alunos podem desenvolver os programas.

Chen & Zhou (2015) propõem um ambiente de programação robótico baseado na Web denominado eRobotic. O ambiente permite a programação no estilo arrastar e soltar. Os objetivos do sistema são motivar os alunos e melhor a compreensão dos conceitos básicos de computação e programação.

4. Ambiente de Análise Programação e Controle (AAPC)

O AAPC é parte do projeto LARA – Laboratório Remoto em AVA (Ambiente Virtual de Aprendizagem), desenvolvido na UESB, que tem como objetivo projetar e implementar uma arquitetura pedagógica que integra recursos tecnológicos e metodologia de ensino para melhorar o processo de aprendizagem de disciplinas do curso de Ciência da Computação.

A estratégia do LARA para atrair e motivar os estudantes de computação para realizar as atividades práticas de programação foi a utilização de laboratório remoto de robótica móvel através do ambiente de programação e controle – APC (Figura 1) [Lopes et al., 2016]. Ele permite que o usuário crie seus próprios códigos para manipular e controlar o robô em um ambiente web sem a necessidade de baixar ou instalar qualquer software adicional.

A versão inicial do projeto permitia a criação e teste de códigos. Em um ambiente educacional, devem ser estimuladas as boas práticas de programação incluindo o entendimento do problema, análise e projeto da solução. O AAPC foi projetado para fornecer suporte a todas estas etapas, além de possibilitar a aprendizagem em contexto, através do uso de laboratório remoto de robótica. Os componentes fundamentais do ambiente são:

- Módulo de estudo – responsável por hospedar material didático que ajuda os alunos na aquisição de conceitos teóricos e na realização de atividades práticas, além de permitir o gerenciamento de mídias educacionais, o gerenciamento dos usuários e a comunicação entre eles.

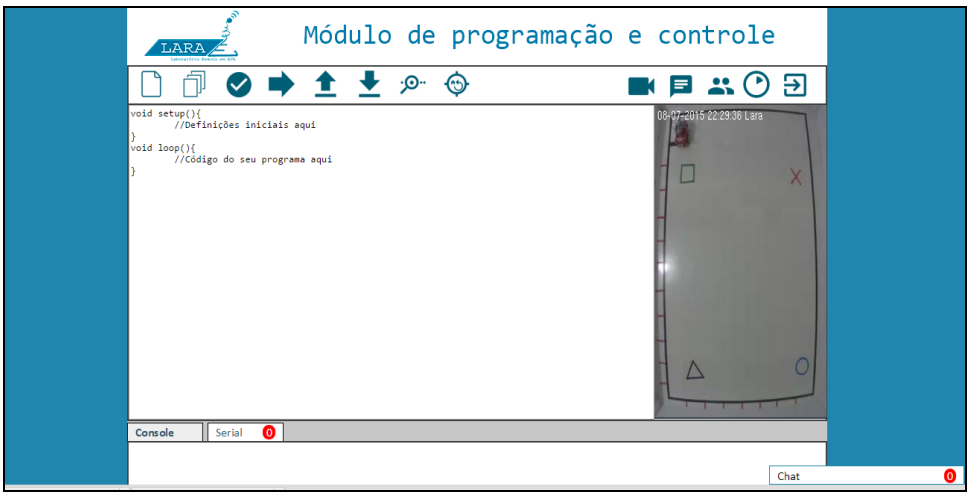


Figura 1 – Ambiente de Programação e Controle do LARA

- Módulo de análise – tem como objetivo estimular as boas práticas de programação incluindo o entendimento do problema e análise da solução. A proposta é que o professor defina um modelo de plano que deve ser seguido e preenchido pelo aluno antes de iniciar a codificação.
- Módulo de codificação – permite a implementação de algoritmos na linguagem de programação Arduino C/C++.
- Módulo de experimentação (ou modulo de laboratório remoto) – permite a manipulação e controle do robô através de códigos desenvolvidos pelo próprio aluno.

A Figura 2 apresenta a visão geral do AAPC. O ambiente é acessado através do AVA *Moodle* que compõe o módulo de estudo. O *Moodle* possui funções de gerenciamento de usuários, de cursos, de objetos de aprendizagem, sistema instrucional de ensino-aprendizagem, entre outras. Para realizar as atividades práticas, o usuário deve realizar a análise do problema, codificação e teste de soluções acessando a área de realização de tarefas que é formado pelos módulos de análise, de codificação e de experimentação. O módulo de análise é composto pelo *Etherpad Lite* (http://etherpad.org/) um editor de texto online de código aberto que permite criar e editar planos para resolução de problemas. Os módulos de codificação e experimentação foram agrupados na área de desenvolvimento. Nesta área, é possível implementar algoritmos através do IDE online do APC e testar e validar a solução através do robô remoto.

O sistema hospeda material didático que ajuda os alunos no processo de aprendizagem de programação. Ele permite ao aluno acessar os materiais, recursos e atividades disponibilizados pelo professor, além de realizar atividades utilizando os módulos de análise, codificação e experimentação. Ao definir uma atividade prática, o professor pode disponibilizar a estrutura (modelo ou *template*) do plano para a solução

de um problema. Por exemplo, o professor pode solicitar que o aluno informe o que ele entendeu do problema, quais os requisitos, as entradas e as saídas, quais as variáveis e seus tipos, um algoritmo para solucionar o problema, etc. O aluno entra na área de análise, abre o plano, pensa sobre o problema, edita o plano com as informações solicitadas e salva.

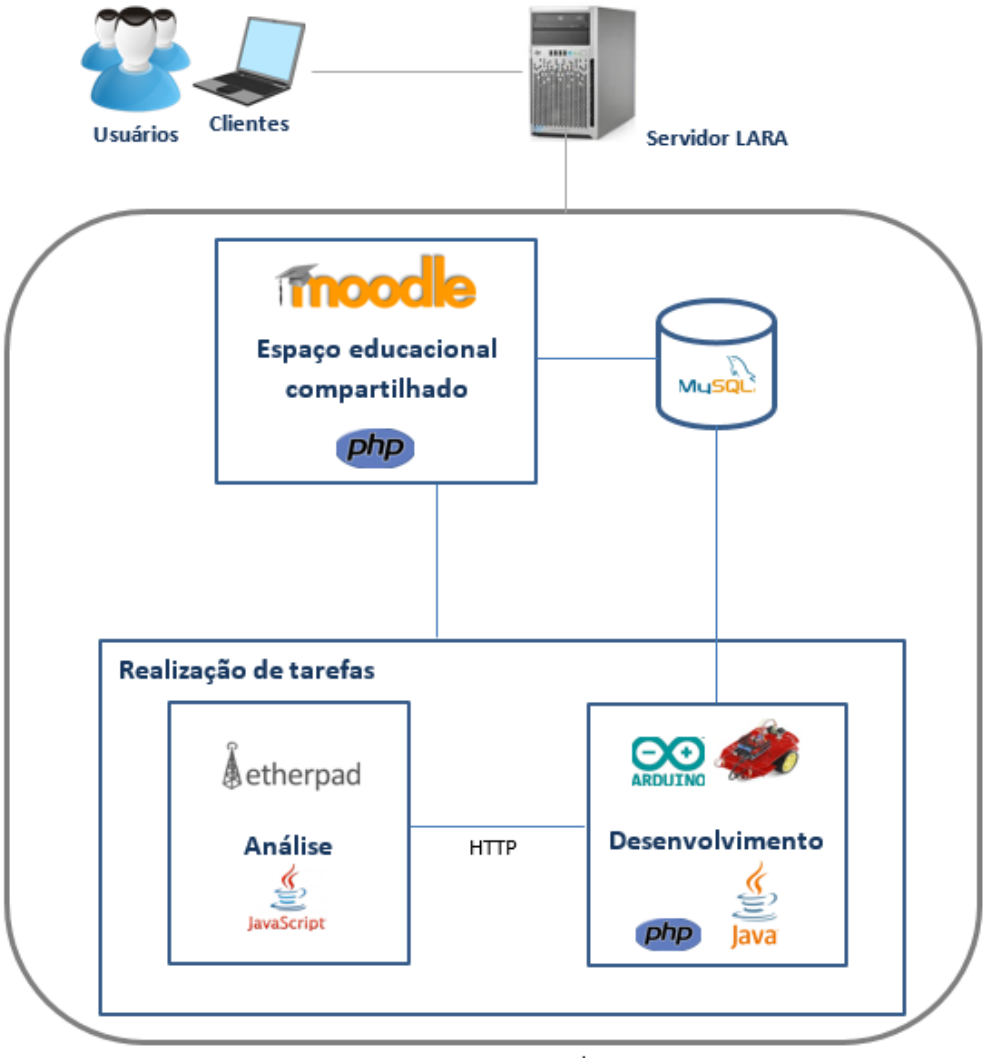


Figura 2 – Visão Geral do AAPC

Na área de desenvolvimento, o aluno executa o plano traçado. Ele cria, salva, abre, edita e compila programas escritos na linguagem de programação. Estes programas devem manipular e controlar o robô. Isto ajuda o aluno no teste e depuração de soluções de problemas. A interação com o robô tem como objetivo permitir a contextualização da programação, pois os professores podem definir contextos de aplicações para as atividades e os alunos podem verificar onde e como aplicar os conceitos abstratos estudados.

A ideia do AAPC é que o alunos leia a atividade a ser realizada, entre no ambiente de resolução de tarefa, na área de análise escreva o problema, pense e registre os requisitos e as restrições referentes ao problema, escreva diversas ideias para resolve-

lo e analise qual a melhor e, então, escreva uma algoritmos para a solução escolhida. Na área de desenvolvimento ele traduz o algoritmo para a linguagem de programação, executa e testa. Caso encontre algum erro semântico, ele deve voltar a área de análise, registrar o erro, fazer as correções necessárias na solução, depois implementa-la e testá-la novamente até resolver o problema corretamente.

4.1 Interface do ambiente

A interface do AAPC utiliza os mesmos princípios aplicados na GUI do APC para tonar o ambiente agradável e contínuo, sem ruptura para o usuário. O ambiente é acessado através do site lara.uesb.br, nele os professores podem criar cursos e definir atividades práticas que serão realizadas no laboratório remoto. Para executar as atividades, o aluno escolhe, no menu Laboratório, a opção Acessar Laboratório (Figura 3).

Figura 3 – Interface do LARA

Ao escolher "Acessar Laboratório", o usuário é direcionado para o ambiente de realização de tarefa mostrado na Figura 4. Nele é possível acessar a área de análise e a área de desenvolvimento. Ambas possuem: uma aba de identificação (1), menu da aba selecionada (2) e área de trabalho (3).

Na aba Análise é possível criar, salvar e abrir um documento, além de formatá-lo. Na área de desenvolvimento (Figura 5), é possível criar novo código, compilar código, enviar código para robô, salvar código, abrir código, abrir porta serial, retornar o robô a posição inicial, abrir/fechar câmera, cronometrar o tempo e sair do ambiente.

5. Considerações finais e trabalhos futuros

Neste trabalho, foi apresentado o ambiente de análise, programação e controle de um robô remoto para o ensino de programação para iniciantes. Para a caracterização do ambiente educacional, foi utilizado o *Moodle* que permite gerenciamento de materiais didáticos e aplicação de estratégias e objetivos de aprendizagem. Para dar suporte as fases de análise e projeto no ensino de programação, foi incorporado um editor de texto onde os estudantes podem definir planos para solucionar as atividades propostas. O

laboratório remoto possibilita a realização de atividades práticas contextualizadas de forma ubíqua, superando as limitações de tempo e espaço dos laboratórios presenciais. A IDE online garante a geração de código em uma linguagem de programação.

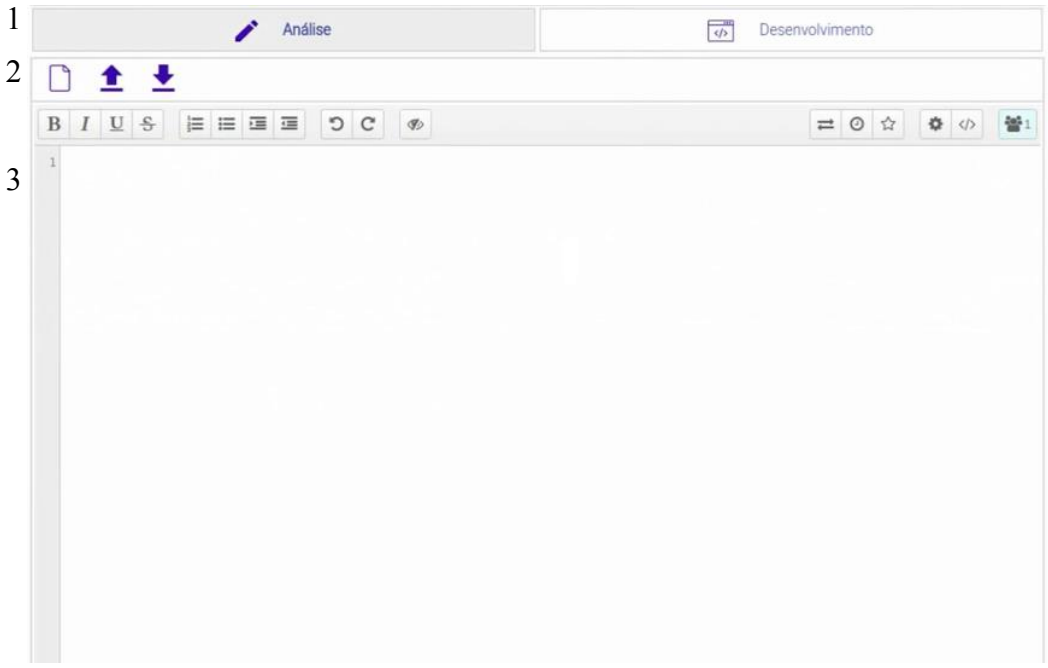


Figura 4 – Ambiente de realização de tarefa

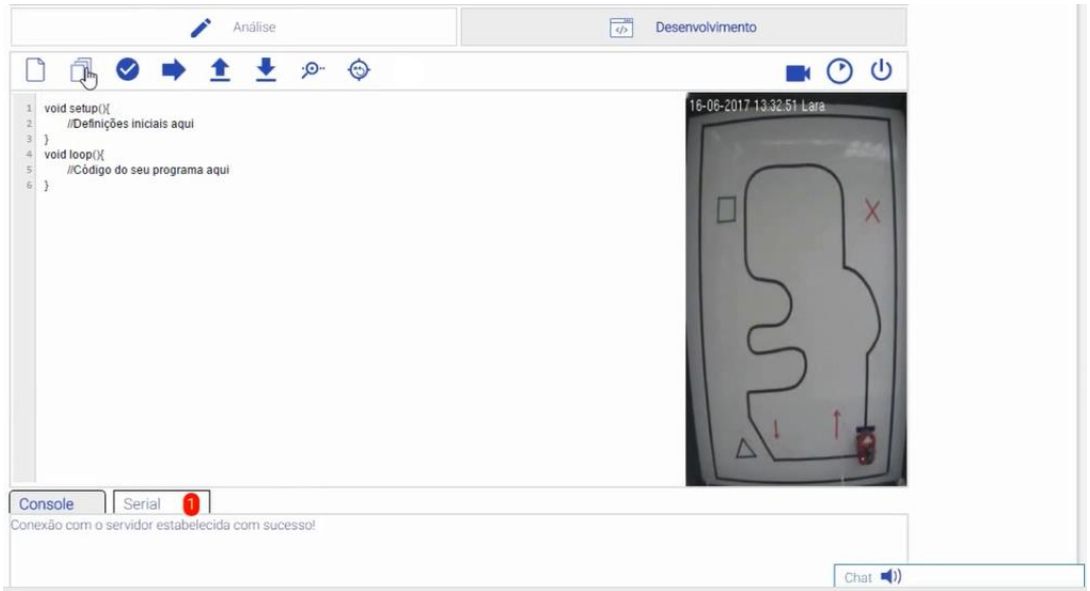


Figura 5 – Área de desenvolvimento

O ambiente aqui apresentado é, portanto, uma alternativa para unir vantagens, baixar custos e proporcionar flexibilidade para criação de contextos educacionais de programação agregando componentes para resolução de problemas (analisar o problema a ser resolvido, formular soluções, produzir algoritmo), codificação de algoritmos,

depuração e teste de soluções. Entretanto, o sistema ainda não foi testado em um ambiente real e, portanto, não foi avaliado a satisfação dos alunos e suas interações com o AAPC.

Como trabalho futuro, pretende-se utilizar o ambiente em um curso de programação com alunos de graduação e também aprimorar os relatórios do AAPC para que os professores possam acompanhar o progresso de cada estudante, mostrando, por exemplo, o plano elaborado, número de tentativas para codificar o algoritmo, se a solução foi efetiva, tempo investido na realização de cada atividade, etc.

Referências

Alkhaldi, T., Pranata, I., & Athauda, R. I. (2016). A review of contemporary virtual and remote laboratory implementations: observations and finding. *Journal of Computers in Education*, *3*(3), 329–351.

Aroca, R. V., Gardiman, R. Q., & Goncalves, L. M. G. (2012). Web-Based Robot Programming Environment and Control Architecture. *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, 27–32.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, *58*(3), 978–988.

Brinson, J. R. (2015). Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research. *Computers & Education*, 87, 218–237.

Chen, X., Song, G., & Zhang, Y. (2010). Virtual and Remote Laboratory Development: A Review. *Earth and Space 2010: Engineering, Science, Construction, and Operations in Challenging Environments*, 368–368.

Chen, Y., & Zhou, Z. (2015). Robot as a Service in Computing Curriculum. *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, 156–161.

Cruz-Martín, A., Fernández-Madrigal, J. a., Galindo, C., González-Jiménez, J., Stockmans-Daou, C., & Blanco-Claraco, J. L. (2012). A LEGO Mindstorms NXT approach for teaching at Data Acquisition, Control Systems Engineering and Real-Time Systems undergraduate courses. *Computers & Education*, *59*(3), 974–988.

Gomes, L., & Bogosyan, S. (2009). Current Trends in Remote Laboratories. *IEEE Transactions on Industrial Electronics*, 56(12), 4744–4756.

Gomes, A., Henriques, J., & Mendes, A. J. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educaçao, Formaçao & Tecnologias*, *1*, 93–103.

Heradio, R., de la Torre, L., Galan, D., Cabrerizo, F. J., Herrera-Viedma, E., & Dormido, S. (2016). Virtual and Remote Labs in Education: a Bibliometric Analysis. *Computers & Education*, 98, 14–38.

Husain, M., Tarannum, N., & Patil, N. (2013). Teaching programming course elective: A new teaching and learning experience. *2013 IEEE International Conference in MOOC, Innovation and Technology in Education* (MITE), 275–279.

Iturrate, I., Martín, G., García-zubia, J., Angulo, I., Dziabenko, O., & Orduña, P. (2013). A Mobile Robot Platform for Open Learning based on Serious Games and Remote Laboratories. In *International Conference of the Portuguese Society for Engineering Education* (CISPEE) (pp. 1–7). Porto.

Koorsse, M., Cilliers, C., & Calitz, A. (2015). Programming assistance tools to support the learning of IT programming in South African secondary schools. *Computers & Education*, *82*, 162–178.

Lang, J. (2012). Comparative Study of Hands-on and Remote Physics Labs for First Year University Level Physics Students. *Transformative Dialogues: Teaching & Learning Journal*, 6(1), 1–25.

Lopes, M., Gomes, I., Trindade, R., Silva, A., & Lima, A. C. (2016). Web environment for programming and control of mobile robot in a remote laboratory. *IEEE Transactions on Learning Technologies*, 1–1.

Ma, J., & Nickerson, J. V. (2006). Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review. *ACM Computing Surveys*, 38(3).

Machotka, J., Nedic, Z., Nafalski, A., & Gol, O. (2009). A remote laboratory for collaborative experiments. *American Society for Engineering Education*.

Park, J. S., & Lenskiy, A. (2014). Mobile Robot Platform for Improving Experience of Learning Programming Languages. *Journal of Automation and Control Engineering*, *2*(3), 265–269.

Santos, R. P. dos, & Costa, H. A. X. (2006). Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. *Infocoomp, Journal of Computer Sciense*, *5*(1).

Souza, D. M. De, Batista, M. H. S., & Barbosa, E. F. (2016). Problemas e dificuldades no ensino de programação: Um mapeamento sistemático. *Revista Brasileira de Informática Na Educação*, *24*, 39–52.

Tan, J., Guo, X., Zheng, W., & Zhong, M. (2014). Case-based teaching using the Laboratory Animal System for learning C/C++ programming. *Computers & Education*, *77*, 39–49.

Uma Metodologia para Implementação da Disciplina Informática Básica em Cursos Técnicos Integrados ao Ensino Médio

Lafayette Batista Melo[1], Paulo Roberto Santos Costa[1], Marcílio de Paiva Onofre Filho[1], Fernando Augusto Ferreira Lordão[1], Leandro Cavalcanti de Almeida[1]

[1]Unidade Acadêmica de Informática – Instituto Federal da Paraíba (IFPB)
CEP 58.015-435 – João Pessoa – PB – Brazil

{lafagoo, prbcosta, marcilionofre}@gmail.com, {fernando.lordao, leandro.almeida}@ifpb.edu.br

Abstract. *This article shows how the discipline of "Basic Computing" was reconstructed for technical courses integrated to high school, employing a methodology that involves: 1) Internet search with advanced techniques; 2) consultation of the vision of managers, teachers and students through a questionnaire; 3) confrontation of 1 with 2 for analysis and categorization of what is up and down in the discipline; 4) research about educational methodologies and technologies and 5) reconstruction of the teaching plan and training of multiplier teachers. The first group of students accepted the new version of the discipline well and the multiplier teachers are employing most of the strategies learned.*

Resumo. *Este artigo mostra como foi reconstruída a disciplina de "Informática Básica" para cursos técnicos integrados ao ensino médio, empregando uma metodologia que envolve: 1) busca na Internet com técnicas avançadas; 2) consulta da visão de gestores, professores e discentes através de um questionário; 3) confronto de 1 com 2 para análise e categorização do que está em alta e em baixa na disciplina; 4) investigação de metodologias e tecnologias educacionais e 5) reconstrução do plano de ensino e capacitação de professores multiplicadores. A primeira turma de alunos aceitou bem a nova versão da disciplina e os professores multiplicadores da capacitação estão empregando a maioria das estratégias aprendidas.*

1. Atualidade da Disciplina Informática Básica e sua Necessidade de Mudança

A disciplina "Informática Básica", oferecida aos cursos técnicos integrados, dá subsídios para uso do computador e de seus programas de modo que o aluno seja mais produtivo academicamente e profissionalmente. Este componente curricular encontra-se em uma situação peculiar devido às mudanças tecnológicas e à própria evolução da Informática, o que faz com que se questione quais conteúdos devem ser dados e de qual maneira, sua atualização, e os subsídios concretos para a formação do aluno. Além disso, há dúvidas sobre se o que está sendo ensinado é útil ou mesmo se não é dispensável, pelo fato de o alunado fazer parte de uma geração impregnada de tecnologia e supostamente já ser detentora do conhecimento de Informática, às vezes até maior do que o dos professores.

Um parâmetro para vislumbrar a disciplina no contexto histórico atual é o estudo do comportamento das gerações, conforme Cordoni (2016). A autora caracteriza as classes genealógicas da seguinte forma: a geração X (nascida de 1960 a 1980) é competitiva profissionalmente, individualista e, apesar de ser a primeira que teve contato com o computador, é a mais resistente às novas tecnologias; a geração Y (1980 a 2000) é otimista, comprometida, multitarefa e totalmente conectada às novas tecnologias – cresceram com as interfaces gráficas (que usam mouses e janelas de aplicação) e são dependentes da Internet para realizar suas tarefas; a geração Z (2000 a 2009) é a composta pelos "nativos digitais", é imediatista e tem preocupações ambientais; e a geração Alpha (2010 em diante) é curiosa, independente e é formada por indivíduos que são conectados bem antes de serem alfabetizados.

Atualmente, uma grande parte de professores de Informática Básica é formada pela geração X e a maior parte dos alunos é constituída pela geração Y. Isso é perpassado pelo olhar de professores X, que ainda se comportam como usuários passivos, ao contrário dos alunos Y. Mesmo que professores Y surjam, há ainda as gerações Z e Alpha que aparecem com novos comportamentos a serem enfrentados. Enfim, a revisão de conteúdos de "Informática Básica", o estudo de novos procedimentos e estratégias de ensino são fatores chave para que a disciplina tenha seu status renovado e com importância ainda maior para as novas gerações.

Adiante, na Seção 2, são analisados alguns trabalhos com diferentes aplicações de Informática no ensino médio e técnico. Isso não pode ser desvinculado das condições e das possibilidades dos cursos técnicos integrados dos Institutos Federais. É por isso que este relato de experiência mostra a realidade atual que é refletida nos planos de curso, conforme colocado na Seção 3. A Seção 4 descreve investigação sobre as visões que gestores, professores e discentes têm da disciplina, pois tais atores não só são imprescindíveis nesta pesquisa, mas têm representantes de diferentes gerações. A Seção 5 destaca sugestões de conteúdo, estratégias de ensino e o desenvolvimento da disciplina. Na Seção 6, temos algumas constatações e recomendações gerais.

2. Trabalhos relacionados

Trabalhos com as especificidades para construção de uma disciplina de Informática Básica, dos cursos técnicos integrados ao ensino médio de Institutos Federais, não foram encontrados, mas algumas abordagens, em um ou outro aspecto relacionado, ajudam a delinear uma metodologia de revisão da disciplina.

Machado (2010) trata da necessidade de inserir o jovem no mercado e de como a Computação é uma oportunidade muitas vezes perdida. Os estereótipos e preconceitos que circulam na escola dificultam a escolha da área como uma alternativa profissional. O autor parte das referências da Olimpíada Brasileira de Informática e da Computação desplugada para criar um projeto de descoberta de novos talentos em Computação. São chamados alunos para turmas específicas que aprendem programação em linguagem C, mas com dinâmicas que envolvem também a ferramenta Scratch. Acreditamos que, no ensino médio e na nossa realidade, é exatamente na disciplina de Informática Básica que uma atividade desse tipo deve ser feita. É nesse momento que muitos dos estereótipos podem ser mais rapidamente desconstruídos, sem contar a divulgação da Computação.

Marques (2017) lida com um projeto de curso técnico em Informática integrado ao ensino médio, mas especificamente direcionado para melhorar o pensamento computacional dos estudantes de modo a facilitar o aprendizado nas disciplinas de programação. O autor utilizou jogos de tabuleiro e problemas de lógica em encontros que davam apoio a alunos que estavam fazendo a disciplina Programação I. O projeto obteve sucesso, mas era algo extra à disciplina e unicamente direcionado à melhoria de programação sem abordar o que uma disciplina de Informática Básica poderia tratar.

Brum (2017) tem uma preocupação especial com as novas gerações, incluindo os nativos digitais, o que faz com que direcione uma disciplina optativa de robótica para a educação básica, utilizando conceitos de gamificação. O trabalho envolve distribuição de tarefas com o cuidado de usar a linguagem da geração de nativos digitais (por exemplo, "time" em vez de "grupo" ou "equipe") e divisão bem demarcada das ações de cada elemento do time de modo que todos participem colaborativamente.

Godinho (2017) trata de um projeto para incentivar crianças e jovens em atividades de programação através de jogos em várias escolas. Para tanto, adota oficinas e minicursos com muitas plataformas de jogos para só depois entrarem no uso de ferramentas como Scratch e App Inventor. As atividades obtiveram alta aceitação da população e satisfação dos alunos, mas o planejamento envolvia decidir quais tarefas realizar com cada grupo infantil ou juvenil e a estrutura da escola.

Aono (2017) realiza uma pesquisa aplicada ao sexto ano do ensino fundamental com uso de Scratch, mas também discute o uso da ferramenta no ensino médio em uma ampla revisão bibliográfica, constatando que a distribuição de aulas práticas e teóricas, lançamentos de desafios e desenvolvimento de jogos devem ser igualmente pensados, mas com planejamento totalmente direcionado para as dificuldades de cada turma.

Pode-se notar que há pesquisas quantitativas e qualitativas, experiências, aplicação de oficinas e minicursos, apoiando ou não disciplinas de Informática (em geral, de programação), com direcionamento ao ensino médio e técnico e até o médio integrado ao técnico, mas não em uma disciplina de Informática Básica. Considerando os cuidados e orientações dos demais trabalhos, acreditamos que propostas de apoio, incentivo, divulgação, desmistificação e melhoria de aprendizagem em Computação podem ser fundidas em uma disciplina deste tipo, mas não sem antes investigarmos as possibilidades e necessidades de cursos técnicos integrados ao médio, em um planejamento que contemple a realidade institucional e as vontades das várias gerações envolvidas. É por isso que adiante vamos relacionar a realidade atual da disciplina em seus planos de ensino com o que estão pensando gestores, professores e alunos.

3. A Realidade Atual dos Planos de Ensino da Disciplina

Para investigar como a disciplina "Informática Básica" está sendo desenvolvida nos cursos técnicos integrados, através de seus planos pedagógicos de curso, foi realizada uma pesquisa com os seguintes passos:

1. **Verificação, através do Google, de planos de curso da disciplina, utilizando palavras-chave e comandos avançados do Google Search**. Exemplos: "plano de curso" "Informática Básica" "curso técnico integrado" – nesse caso, para obter planos da disciplina diretamente nos resultados de busca, clicando neles e baixando

o documento quando for adequado; "Instituto Federal" "plano de curso" "Informática Básica" – do mesmo modo, para uma pesquisa mais ampla do que a anterior, que poderia não incluir “curso técnico em”, “plano do curso técnico em Informática” filetype:pdf – para obter os documentos mais comuns no formato PDF; etc.

2. **Leitura dos planos, especialmente as ementas**.
3. **Coleta dos links, da descrição das ementas dos cursos e endereços na Web onde estão os documentos** – uso da ferramenta Linkclump (https://chrome.google.com/webstore/detail/linkclump/lfpjkncokllnfokkgpkobnkbkmelfefj);
4. **Registro dos 100 primeiros resultados de pesquisa coletados em uma planilha do Google Docs** – no endereço https://docs.google.com/spreadsheets/d/1j39XdhzOQ0Hk976RYsvEV7aXGQOECF_hFOmzaQJTjGM/edit#gid=0;
5. **Geração de uma nuvem de palavras com os termos usados nas ementas dos cursos** – para verificar possíveis conceitos e assuntos que são mais tratados em todos os planos das disciplinas – uso da ferramenta Wordclouds.com: https://www.wordclouds.com/

Resultados das pesquisas são resumidos adiante, com os dados mais relevantes.

Em relação ao conteúdo das disciplinas – As ementas se baseiam em tipo de software, componentes de hardware, uso de editor de texto, planilha e softwares de apresentação. Ao final, algumas acrescentam tópicos referentes ao curso técnico (se for um curso médio integrado ao técnico). Algumas falam de ambientes virtuais de aprendizagem (quando são feitas a distância). Mesmo os cursos de ensino médio integrados ao técnico na área de Informática têm “Introdução à Informática” com uso de aplicativos, história da Internet, uso de utilitários, aplicativos e sistemas operacionais.

Em relação às referências bibliográficas adotadas – Usam muita referência bibliográfica antiga (de 2002, 2003, 2004) mesmo em cursos de 2014 até hoje.

Nomenclatura – A maioria é “Informática Básica”, mas tem “Introdução à Informática”, apenas “Informática” e “Ambientes virtuais e Informática”. Algo como “Informática Aplicada” não foi observado.

Metodologias de ensino – Nenhum plano fala de métodos novos ou aplicação de software para o ensino; são tradicionais com “aulas expositivas” e com acesso a computador e à Internet.

Objetivos dos planos – Os principais objetivos ainda são operar softwares básicos e aplicativos.

O que indicam as pesquisas – Em uma busca no Google com “Instituto Federal” plano “Informática Básica” -técnico -integrado filetype:pdf”, que quer dizer use esses 3 primeiros termos e exclua “técnico” e “integrado” da pesquisa para encontrarmos só arquivos PDF, o resultado ainda é muito grande com cursos técnicos integrados ao médio. Verificar ementa de cursos exclusivamente do ensino médio, sem técnico foi muito difícil. Tentou-se várias combinações e todas elas tinham o mesmo

problema. Esses resultados sugerem que os ensinos médios integrados aos técnicos têm uma tendência grande de serem os que mais requisitam a disciplina.

Foi feita uma nuvem de palavras das ementas com o Wordclouds.com, para vislumbrar melhor, de uma forma panorâmica, comprovar resultados anteriormente obtidos e identificar ainda outras possíveis peculiaridades, conforme a Figura 1.

Figura 1 – nuvem de palavras com termos das ementas

Comprovou-se não só que o conteúdo dado é ainda tradicional, de acordo com as ementas, mas que provavelmente é ainda fixo e pouco muda. Por exemplo, discussão sobre histórico do computador ainda é importante embora pareça ocupar menor espaço que antigamente. Uso de aplicativos (editor de texto, planilha e software de apresentações) continuam unanimidade, bem como menção aos componentes de hardware. Sistemas numéricos têm uma tendência a aparecer cada vez menos. Administração, segurança e produtividade parecem ser assuntos pontuais que, embora não ocupem grande parte da carga horária do curso, surgem mais recentemente.

4. A Visão de Gestores, Docentes e Discentes

Foi constituída uma comissão para revisão da disciplina de “Informática Básica” tanto para a investigação da realidade atual em vários planos do Brasil, como mostrado anteriormente, quanto para consulta com os vários atores institucionais em reuniões departamentais de todos os cursos técnicos integrados ao médio. Aplicaram-se questionários com gestores (que devem conhecer melhor a realidade administrativa da instituição e dos cursos), com docentes (que podem mostrar suas limitações e aplicações de novos conhecimentos) e com discentes (cujas necessidades e motivações acreditamos que devem ser sondadas, especialmente se os confrontamos com atores de outras gerações). Utilizou-se o Google Docs para aplicar o questionário on-line em

formulários, mas com horários específicos para cada turma ou reunião em laboratório, de modo que tivéssemos exatamente as reais pessoas envolvidas de cada grupo.

A pesquisa feita com os gestores envolveu uma quantidade menor de atores e as perguntas envolviam saber sua função, área, curso, expectativas sobre a disciplina, visão dos alunos, conteúdo e ferramentas a serem usados. Todos os gestores tinham a função de coordenadores de curso. Ressalta-se a pergunta sobre o que deveria ser melhorado na disciplina, com uma quantidade de respostas maior relacionada à infraestrutura, algo que não pertence à esfera de poder da comissão com a nova proposta, mas que pode ser cobrado institucionalmente. Ao menos orientou-se para refletirmos no uso ou não de ferramentas gratuitas. Como responderam apenas coordenadores de cursos técnicos integrados ao médio, as respostas sobre o conteúdo a ser aplicado variaram muito, com referência a marketing virtual, mídias sócias, Arduíno, iniciação a programação e planilhas; algo certamente relacionado à realidade de cada curso

Em relação aos docentes, as perguntas relacionadas envolviam conteúdos que deveriam continuar na nova versão da disciplina com os assuntos: conceitos básicos (definição e classificação de hardware e software foram citados por todos os professores); sistemas operacionais (gerenciamento de arquivos e pastas e configuração básica foram lembrados por 83%); Internet (uso de e-mails citados por 100%); edição de textos (os mais mencionados com cerca de 80% foram edição básica, formatação e uso de tabelas e figuras); planilhas (com ocorrência de edição de tabelas e gráficos); e noções gerais – nesse caso “aplicativos da área de cada curso” foram mencionados por todos os professores em detrimento de redes de computadores (50%) e redes sociais (25%). Sobre os conteúdos a incluir, os escolhidos com maior frequência foram noções de programação com Scratch ou App Inventor (100%), cidadania e segurança na Internet (66%) e armazenamento em nuvem (66%). A maior dificuldade descrita pelos professores em questão aberta foi a heterogeneidade de conhecimento dos alunos.

O questionário com os alunos foi mais amplo, juntando o 2º, 3º e 4º anos de alunos que já haviam feito a disciplina, em um total de 158 alunos de cursos técnicos integrados ao médio. As perguntas foram semelhantes às feitas aos professores e os resultados foram também muito parecidos. Em relação a quais conteúdos deveriam continuar, a ampla maioria citou edição de texto, planilha e programas de apresentação. Sobre quais conteúdos deveriam ser retirados, a maior incidência de respostas foi “nenhum conteúdo deve ser retirado”, com mais de 60%, mas algumas citações sobre histórico do computador ficaram em menos de 10%, enquanto que outros conteúdos foram citados de forma mais dispersa com pouco percentual. Sobre quais conteúdos deveriam ser inseridos, programação teve cerca de 40% de incidência, mas com muitas variações (Python, C, Arduíno, C++, Scratch e etc.). Também apareceram respostas como “ABNT”. É importante salientar que, diferentemente da pesquisa junto aos docentes, o questionário com os alunos foi feito com respostas abertas e certamente por conta disso houve tanta variação. Mesmo assim, há coincidência com os professores, principalmente em relação a quais assuntos deveriam continuar. As maiores reclamações dos alunos estão relacionadas à superficialidade dos assuntos e dinâmica das aulas, embora não queiram mudar os assuntos, e a necessidade de uma ementa mais direcionada ao curso, com variações conforme a área e a familiaridade dos alunos.

5. Conteúdo, Estratégias de Ensino e Desenvolvimento da Disciplina

Confrontando o levantamento dos planos de curso com a consulta feita a professores, gestores e alunos, bem como a experiência dos autores desta pesquisa, que também constituíram a comissão de revisão, o plano foi reformulado, mas houve nova reunião com as coordenações para que se mostrasse a necessidade de haver professores que tivessem algum conhecimento específico ao final do período. Isso aconteceu em decorrência de, apesar de os planos não fazerem este tipo de referência, haver uma preocupação de todos os atores da instituição. O gerenciamento da disciplina é feito pela área de Informática com os cuidados descritos a seguir nas próximas Sub-Seções.

Antes, porém, é importante que mostremos uma fase intermediária da implementação que consiste em listagem e categorização do que está em alta e em baixa para o desenvolvimento do curso, considerando todos os dados obtidos e o que poderia ser implementado no momento atual, conforme a Figura 2.

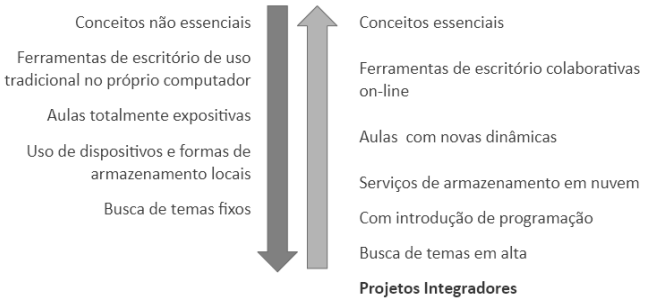


Figura 2 – o que está em alta e em baixa na disciplina Informática Básica

Em relação ao que está em baixa, consideramos conceitos não essenciais que poderiam ser retirados no momento, por não terem seguimento na própria disciplina ou por já serem abordados em outros componentes curriculares ou já serem de conhecimento do aluno como conversão numérica e classificações exaustivas de tipos de software e hardware. Dentro de conceitos em alta, consideramos que menções históricas ainda devem ser tratadas, mas de forma mais reduzida e com outra forma de apresentação. Uso de aplicativos, por exemplo, ainda é algo referido por todos como importante, mas exatamente o que deve ser trabalhado demanda investigar as necessidades atuais da turma. Em baixa, está o uso de aplicativos específicos de escritório ainda apenas instalado no próprio computador. Alunos já usam muita edição de textos nas nuvens e pacotes deste tipo são oferecidos gratuitamente para o ensino. Isso não quer dizer que nada de edição de texto e planilhas em desktop deva ser mencionado, pois em empresas e em várias submissões de documento ainda são requeridos formatos DOC ou PDF, por exemplo. A solução aqui é priorizar pacotes nas nuvens, que inclusive são usados em convênios com instituições de ensino, como o Google For Education, mas sempre orientando os professores para mencionarem as diferenças em relação ao uso único no computador. As aulas apenas expositivas e com exercícios estão em baixa, mas há possibilidades de se incluir outras estratégias e até ferramentas para dinamizar as aulas, as quais devem ser pesquisadas a cada edição da disciplina devido às mudanças e disponibilidade. Uso apenas de dispositivos de armazenamento de massa, como eram antigamente tratados, como CD e disquete não fazem mais sentido atualmente. O pendrive também está cedendo para armazenamento nas nuvens, cujos alcances devem ser esclarecidos para os alunos, seja para edição de

documento em conjunto, controle de versões, formas de conversão entre outas possibilidades, incluindo o gerenciamento móvel. Imaginar que a busca por temas consagrados e fixos devam ser um intuito em disciplinas como essa parece ser algo a fugir da realidade. Uma inspeção com professores e coordenações de curso, bem como com os alunos, para atualização constante da ementa é algo que se faz extremamente necessário. Neste período, procuramos inserir, como sugerido pelos participantes, discussões temáticas e dicas sobre cidadania digital, segurança e privacidade em redes sociais. A inserção da programação se faz necessária exatamente nesta disciplina, pois é um instante do curso que abre o aluno para várias possibilidades do uso da Informática, e que é premente hoje em dia para inserção do pensamento computacional. Isso não quer dizer que haverá um aprofundamento em programação, mas a inserção de noções de forma lúdica e sempre voltada para um passo posterior no curso técnico. A ideia de projetos integradores nos documentos de referência dos cursos técnicos integrados ao médio parece ocorrer em um momento oportuno para conversar com professores de todos os cursos e direcionar, por exemplo, até mesmo os aprendizados iniciais de programação. Vejamos como fizemos a primeira implementação da nova proposta.

5.1. Conteúdo

No primeiro ano do ensino médio, com carga horária de 66h e duas aulas por semana, com a seguinte distribuição:

1º Bimestre – Introdução à Informática – utilizamos um convênio da nossa instituição com uma empresa de renome que faz treinamentos a distância em ambientes virtuais especializados e colocamos isso como proposta para os alunos. O conteúdo trata ainda de histórico e tipos de hardware e software, mas com testes virtuais e acompanhamento que possibilita um certificado para conclusão da etapa. As aulas são feitas em grande maioria no formato de sala de aula invertida, para discutir o que os alunos estudaram no ambiente, e nesta etapa mais de 90% de uma turma já obteve o certificado. O uso desse ambiente e do convênio será sempre reavaliado.

2º Bimestre – Ferramentas colaborativas, utilizando como exemplo o Google For Education que também foi cadastrado para a instituição. Além das diferenças de usar programas apenas para o próprio computador, são feitos desafios para as equipes em cada aula para construção, por exemplo, de artigos ou formulários. Mostra-se o que há ainda de limites nas nuvens, que só com as ferramentas desktop pode ser realizado, mas também como extensões ao browser e complementos nas diferentes aplicações já inserem possibilidades novas como sumário, cores, gerenciamento de figuras, geração de nuvens de palavras, contagem de palavras, formas de compartilhamento e rastreamento de dados etc.

3º Bimestre – Fundamentos de programação – utilizamos o Scratch para iniciar programação com os cuidados descritos na Seção de trabalhos relacionados, mas voltando-se para resolver um problema específico do curso técnico integrado ao médio, cuja discussão é feita antes com as coordenações, podendo se iniciar projetos de robótica até contabilidade empresarial, conforme o que vão requerer antes os participantes.

4º Bimestre – Programação para Mobile – utilizamos o App Inventor, do MIT, que tem conceitos básicos de programação, continuando o que foi aprendido com o

Scratch, mas desbravando as possibilidades dos dispositivos móveis que vão desde a construção de aplicativos simples para acionar música com o movimento do celular até construção de jogos que associem figuras a textos, sons e animações, cujo conteúdo cada aluno poderá direcionar conforme seu interesse e o do curso.

5.2. Tecnologias e Metodologias de Ensino

Além de trabalharmos com a ideia de aula invertida para alguns conteúdos, fazermos propostas e desafios, utilizamos tecnologias que possibilitam discutir ou testar conceitos rapidamente (por exemplo, após o estudo de conceitos do 1º. Bimestre ou após explanação em laboratório de conceitos de programação no 3º. e 4º. Bimestres). Essas tecnologias de testes rápidos em sala ou laboratório possibilitam uso de celular nas aulas e o professor pode produzir vários tipos de teste configurável. Um exemplo é a criação de testes com sete perguntas de marcar, cada uma com uma cronometragem, mostrando a pontuação dos alunos (medida pela resposta certa e tempo de resposta), com a posição final de cada um após o desafio. Isso possibilita rever conteúdo, tirar dúvidas e criar novas dinâmicas de interação professor-aluno e aluno-aluno. As ferramentas que usamos para produção dos testes e compartilhamento, reuso e modificação entre os próprios professores são o Kahoot[1]e o Quizziz[2].

5.3. Plano de Ensino

A disciplina continuou com o nome de “Informática Básica”, pois o que se quer aplicar mais na disciplina ao seu final é conteúdo dos cursos para desenvolvimento do próprio conteúdo de Informática Básica, além de a grande maioria de cada um dos grupos de atores institucionais estarem todos em concordância sobre o desenvolvimento de conteúdos básicos, embora com nova extensão. Procurou-se distribuir planos com professores, tratando também das metodologias usadas, algo que não foi verificado nos planos anteriores, mas que parece enriquecedor.

5.4. Capacitação de Professores

Em conjunto com a aplicação do plano de forma sistematizada neste período – em períodos anteriores ocorreram experiências esparsas – foi realizada uma capacitação com sete professores de cada um dos cursos técnicos integrados ao médio. A capacitação trata do modo como conteúdo e estratégias são desenvolvidas, mas também é um espaço de consulta com os professores multiplicadores de detalhes do programa e sugestões adicionais para cada um dos cursos técnicos. Também obtivemos relatos de como estão aplicando o aprendido ao desenvolverem as disciplinas no período atual.

6. Algumas Constatações e Recomendações Gerais

Ao implementarmos a disciplina por meio de uma nova metodologia, pretendemos responder a questionamentos que incluem abordar ou não certos conteúdos, mas não só isso. Também norteamos sobre procedimentos pedagógicos, comportamentos, atitudes,

[1] https://kahoot.com/

[2] https://quizizz.com/

metodologias e tecnologias de ensino que são tão caras para as gerações que se relacionam hoje em dia e que precisam de novos enquadramentos e formas de interação para um melhor aproveitamento acadêmico e profissional.

Outra contribuição deste trabalho está na própria metodologia de investigação para revisão, modificação e aplicação da nova versão do componente curricular, a qual podemos descrever nos seguintes passos: 1) busca na Internet com ferramentas de pesquisa e estratégias avançadas de busca de planos pedagógicos semelhantes; 2) consulta da visão de gestores, professores e discentes através de um questionário sobre o atual desenvolvimento da disciplina; 3) confronto de 1 com 2 para análise e categorização do que está em alta e em baixa na disciplina; 4) investigação de metodologias e tecnologias educacionais atuais a serem adequadas especificamente para o novo conteúdo; 5) reconstrução do plano de ensino e capacitação de professores multiplicadores das disciplinas para atuarem de forma direcionada nos cursos técnicos integrados ao médio.

Tanto no estudo de aplicação sistemática da disciplina no ano atual quanto na capacitação dos professores, alunos e discentes têm respondido muito bem à mudança, com questionamentos apenas pontuais e eventuais sobre o nível de detalhamento dos conteúdos. Pretende-se em trabalhos futuros fazer uma pesquisa ampla sobre os vários anos de aplicação da nova disciplina de "Informática Básica" bem como replicar o método de investigação para revisão e mudança de outros componentes curriculares em Computação, verificando possíveis adequações.

Referências

Aono et al. (2017) "A Utilização do Scratch como Ferramenta no Ensino de Pensamento Computacional para Crianças", In: Anais do XXXVII Congresso da Sociedade Brasileira de Computação, SBC, São Paulo, http://csbc2017.mackenzie.br/public/files/25-wei/9.pdf, Julho.

Brum et al. (2017) "Gamificação para o Ensino de Computação na Educação Básica", In: Anais do XXXVII Congresso da Sociedade Brasileira de Computação, SBC, São Paulo, http://csbc2017.mackenzie.br/public/files/25-wei/2.pdf, Julho.

Cordoni, T. (2016), Conhecendo as diferenças. Guia geração da Internet, São Paulo, p. 6-13, 2016.

Godinho et al. (2017) "Projeto Aprenda a Programar Jogando: Divulgando a Programação de Computadores para Crianças e Jovens", In: Anais do XXXVII Congresso da Sociedade Brasileira de Computação, SBC, São Paulo, http://csbc2017.mackenzie.br/public/files/25-wei/6.pdf, Julho.

Machado et al. (2010) "Uma Experiência em Escolas de Ensino Médio e Fundamental para a Descoberta de Jovens Talentos em Computação", In: Anais do XXX Congresso da Sociedade Brasileira de Computação, SBC, Belo Horizonte, http://www.inf.pucminas.br/sbc2010/anais/pdf/wei/st01_04.pdf, Julho.

Marques et al. (2017) "Pensar para Programar: Projeto de Ensino no Curso Técnico em Informática", In: Anais do XXXVII Congresso da Sociedade Brasileira de Computação, SBC, São Paulo, http://csbc2017.mackenzie.br/public/files/25-wei/3.pdf, Julho.

Relato de Experiência Vivenciada no PIBID sobre a Utilização da Computação Desplugada, a Hora do Código e do Scratch no Ensino Médio

Anna Raquel da S. Marinho[1], Pauleany S. de Morais[2], Givanaldo R. de Souza[3], Alba S. L. do Nascimento[1]

[1]*Campus* Zona Norte – Instituto Federal do Rio Grande do Norte (IFRN)
Natal – Rio Grande do Norte – Brasil

[2]*Campus* de Educação a Distância – Instituto Federal do Rio Grande do Norte (IFRN)
Natal – Rio Grande do Norte – Brasil

[3]*Campus* Parnamirim – Instituto Federal do Rio Grande do Norte (IFRN)
Parnamirim – Rio Grande do Norte – Brasil

raquelmarinho.linfor@gmail.com,{pauleany.morais, givanaldo.rocha, alba.lopes}@ifrn.edu.br

Abstract. *This article aims to present an application of Science Unplugged, the Hour of Code and Scratch as tools for introducing Computational Thinking in High School. The methodology used in the development of this work is based on action-research (LEWIN, 1946). The experiences lived in Institutional Scholarship Program Introduction to Teaching (PIBID), Informatics subproject. After analyzing the results achieved, it is considered that it is possible to introduce the concept of Computational Thinking in High School, in order to allow students to be actors in the teaching and learning process.*

Resumo. *Este artigo objetiva apresentar a utilização da Computação Desplugada, a Hora do Código e o Scratch como ferramentas de introdução ao Pensamento Computacional no Ensino Médio. A metodologia utilizada no desenvolvimento deste trabalho é baseada na pesquisa-ação (LEWIN, 1946). A experiência relatada foi vivenciada nas ações do Programa Institucional de Bolsas de Iniciação à Docência (PIBID), subprojeto Informática. Ao analisar os resultados alcançados, considera-se que é possível introduzir o conceito de Pensamento Computacional no Ensino Médio, tendo em vista permitir que os alunos sejam atores no processo de ensino e aprendizagem.*

1. Introdução

Sabe-se que "é importante que juntamente com a inserção do computador na vida dos alunos, o método de ensinar e o conteúdo ensinado sofram alterações que permitam o uso efetivo e qualitativo dessa ferramenta tecnológica" (SILVA; ROMANI; BARANAUSKAS, 2008, p.31). Sendo assim, este trabalho apresenta em suas seções algumas considerações sobre perspectivas para o ensino de Informática, por meio da utilização da Computação Desplugada, a Hora do Código e do *Scratch.*

A experiência aqui relatada é baseada no método da pesquisa-ação (LEWIN, 1946), cujo processo orienta o pesquisador a seguir as etapas de identificar as estratégias de ação a serem implementadas, sistematizar a observação, reflexão e mudança

(GRUNDY; KEMMIS, 1982). Nessas etapas, de acordo com Tripp (2005, p. 446) "planeja-se, implementa-se, descreve-se e avalia-se uma mudança para a melhora de sua prática, aprendendo mais, no correr do processo, tanto a respeito da prática quanto da própria investigação".

Trata-se de uma abordagem experimental, visto que as estratégias utilizadas perpassam a metodologia de ensino tradicional ao construtivismo com o objetivo de atender à aprendizagem cognitivista, na qual o sujeito deve utilizar habilidades de raciocínio lógico para resolver os problemas propostos. Portanto, sugere-se que as técnicas utilizadas nas ações de pesquisa e investigação devam ser tomadas com fins de melhorar prática (TRIPP, 2005).

A primeira parte desse estudo ocorreu no ano de 2015, cuja experiência é relatada em Marinho et. al. (2017). A partir da análise das metodologias utilizadas, das respostas dos questionários aplicados e dos resultados alcançados na 1ª experiência, foi realizado um segundo estudo que corroborou com a concepção das estratégias utilizadas na 2ª experiência (no ano de 2016), que será relatada neste trabalho. Na Seção 2 serão apresentadas as ferramentas e a técnica utilizadas. Na Seção 3, as Experiências de Introdução ao Pensamento Computacional no Ensino Médio. Por fim, na Seção 4, as considerações finais.

2. O Pensamento Computacional e as ferramentas utilizadas

Desde o primeiro artigo da Jeannette Wing, publicado em 2006, dissertando sobre o Pensamento Computacional, os pesquisadores de Informática na Educação têm utilizado estratégias que corroboram com a operacionalização de práticas pedagógicas que mobilizam competências do pensar computacionalmente.

> *Pensamento Computacional é uma forma para seres humanos resolverem problemas; não é tentar fazer com que os seres humanos pensem como computadores [...]; humanos são espertos e imaginativos. Nós humanos tornamos a computação empolgante. Equipados com aparelhos computacionais, usamos nossa inteligência para resolver problemas que não ousaríamos sequer tentar antes da era da computação e construir sistemas com funcionalidades limitadas apenas pela nossa imaginação (WING, 2006, p. 4).*

Ou seja, o pensar computacionalmente não é restrito apenas a quem está inserido na área da computação, mas apresenta-se como uma competência que, de acordo com Cavalcante, Costa e Araújo (2016, p. 1119), "consiste em um conceito mais abrangente no qual há um conjunto de habilidades e atitudes vinculadas na realização de uma ação", relevante para todas as pessoas, considerando que um dos principais objetivos é a resolução de problemas, independente da área do conhecimento. Diante destas perspectivas, serão descritas as duas ferramentas e a técnica utilizadas durante a oficina relatada neste trabalho: a Hora do Código, o *Scratch* e a Computação Desplugada, respectivamente.

2.1. A Hora do Código (The Hour of Code)

A Hora do Código é um movimento da Code Studio (2017), criado com o intuito de demonstrar que independente da idade, gênero, etnia e do conhecimento que as pessoas possuem é possível aprender conceitos básicos de programação por meio de desafios lúdicos online. O movimento busca oportunizar experiências de utilização de

alguns fundamentos da ciência da computação. Essas experiências contribuem para o desenvolvimento de habilidades como raciocínio lógico, resolução de problemas e criatividade.

Os desafios da Hora do Código contêm tutoriais em forma de vídeos e balões explicativos com instruções para que o usuário consiga entender sem dificuldades o funcionamento do programa e atingir o objetivo de cada fase. Não é necessário cadastrar-se para participar dos desafios. Ao completar todas as fases o sistema gera um certificado de participação que pode ser compartilhado no Facebook. O site code.org oferece outros cursos, para crianças a partir dos 4 anos de idade. Além dos desafios e cursos online, são disponibilizadas instruções de atividades desplugadas (sem o uso do computador).

2.2. A ferramenta Scratch

Desenvolvido para que o usuário possa programar suas próprias histórias interativas, jogos e animações e compartilhar suas criações com outros membros da comunidade online. Foi projetado especialmente para estudantes entre 8 e 16 anos, mas é usado por pessoas de todas as idades. O *Scratch* é usado em mais de 150 países e está disponível em mais de 40 idiomas (SCRATCH, 2015). O acesso à ferramenta pode ser feito online, porém é possível fazer o download da versão instalável. Esta versão é compatível com os sistemas operacionais Mac, Windows e algumas versões do Linux, no entanto para que o programa funcione é preciso instalar também o Adobe Air - disponibilizado na própria página de download do *Scratch:* https://scratch.mit.edu/.

2.3. A Computação Desplugada

A Computação Desplugada (do inglês, *Science Unplugged*), conforme Bell *et al*. (2011), pode ser caracterizada como um método de ensino de fundamentos da computação sem o uso do computador, a qual possibilita o ensino de conceitos computacionais de maneira lúdica, podendo ser utilizada do ensino fundamental ao ensino superior. A relevância da utilização desta técnica é evidenciada na fala de Vieira, Passos e Barreto (2013, p.672) quando destacam "a sua independência de recursos de hardware ou software" por

> *(i) não requerer computadores; (ii) ensino da ciência da computação real; (iii) aprender fazendo; (iv) ser divertido; (v) sem nenhum equipamento especializado; (vi) variações da aplicação da técnica são encorajadas; (vii) para qualquer pessoa; (viii) durante as atividades, enfatizar a cooperação, comunicação e solução de problemas; (ix) atividades são auto-suficientes, ou seja, podem ser usadas independentemente umas das outras e; (x) devem ser flexíveis com relação a erros, isto é, pequenos erros não devem impedir que os participantes entendam os fundamentos (Vieira, Passos e Barreto, 2013, p.672).*

2.4. Produção de Jogos Educacionais

Uma alternativa bastante utilizada nos últimos anos para a introdução à programação de acordo com Gomes, Tedesco e Melo (2016, p. 62) "é o desenvolvimento de jogos digitais por meio de linguagens visuais de programação". Essa alternativa possibilita o desenvolvimento de competências do Pensamento Computacional. A produção de jogos permite trabalhar com os alunos a resolução de problemas e o raciocínio lógico, dentre outras habilidades, de maneira lúdica.

Nessa perspectiva, Gomes, Tedesco e Melo (2016) dissertam que nas práticas introdutórias à programação recomenda-se a utilização de técnicas que possibilitem a aprendizagem por meio da prática. No entanto, ressaltam como um dos desafios a escolha das ferramentas que atendam grupos heterogêneos.

3. Introdução ao Pensamento Computacional no Ensino Médio

A abstração dos conhecimentos depende muito da forma em que eles são apresentados. Por consequência, não é viável simplesmente descaracterizar a importância da aprendizagem escolar em detrimento dos recursos tecnológicos informacionais. Mesmo que exista a informação e que haja um processo cognitivo aliado, se faz necessário a orientação que está presente no contexto escolar (LIBÂNEO, 2000). Cabe aos educadores orientar os seus alunos sobre como utilizar a tecnologia de forma proveitosa, para a produção do conhecimento.

As ações realizadas na Escola Estadual Professora Ana Júlia de Carvalho Mousinho, aconteceram tendo como proposta o ensino de Iniciação à Programação com o Software Educacional Scratch, cujo intuito foi desmistificar o conceito popular de que o professor de Informática apenas dá aulas de Informática Básica, bem como a introdução ao Pensamento Computacional. Portanto, foram planejadas aulas teóricas e práticas para a oficina de Informática, trabalhando os conceitos dos elementos que compõem o Scratch, visto que neste software "a programação dispensa a digitação de código e se baseia em arrastar e soltar blocos de comandos" (BINI; KOSCIANSKI, 2009, p. 6). Nesta seção, serão descritas as etapas realizadas para a concepção dos projetos criados pelos alunos participantes da oficina.

3.1. Iniciação à programação com o uso da Computação Desplugada

Nesta subseção será descrito o relato das aulas - metodologias utilizadas e os projetos desenvolvidos pelos alunos - da 2ª edição da Oficina de Iniciação à Programação Utilizando o *Scratch* e a Computação Desplugada*,* no ano de 2016, da qual participaram 20 alunos do Ensino Médio (com idade entre quinze e vinte e um anos) do turno matutino, destes, 5 cursando o 1º ano, 6 o 2º ano, e 9 o 3º ano, ministradas por duas bolsistas do PIBID subprojeto Informática, tendo na fase de concepção dos projetos o auxílio de mais um bolsista e um voluntário do 4º período da Licenciatura em Informática do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, *campus* Natal/Zona Norte.

Iniciou-se o módulo de Computação Desplugada, apresentando o conceito dos termos Computação Desplugada e Pensamento Computacional. Para a realização das atividades desse módulo propôs-se que a turma se dividisse em três grupos. Cada grupo deveria escolher um representante responsável por executar as instruções. Para explicar na prática como funciona a sequência de instruções que nos leva a resolução de um problema, foi proposta a atividade "Qual o objeto?", uma adaptação da atividade *Descrevendo um objeto diário* encontrada na página do Google for Education. Cada grupo recebeu um envelope contendo o nome de cinco objetos.

Os representantes de cada grupo descreveram os objetos, cada um para a sua equipe e os demais componentes tiveram um minuto para decifrar cada objeto descrito, escrevendo as instruções e desenhando o objeto. Foram utilizados o computador interativo, envelopes, cartões de papel ofício e canetas. Apenas um, dos três grupos, não

conseguiu descobrir todos os objetos descritos pelo representante. Todos conseguiram compreender que para resolver um problema é necessário percorrer uma sequência de ações.

Na 2ª aula realizou-se a atividade *O Labirinto* criada com o objetivo de representar o funcionamento do computador no relacionamento solicitação do usuário x execução do sistema operacional. Para tanto, montou-se um labirinto dentro do Laboratório de Informática, delimitado com fita crepe, contendo obstáculos (cesto de lixo vazio, cadeiras, fita crepe, fios grossos de náilon) - segunda atividade de Computação Desplugada. Essa atividade foi realizada em duas etapas. Os alunos estavam fora da sala quando o labirinto foi montado. Um representante de cada grupo foi vendado para percorrer o labirinto, seguindo as instruções dos colegas. As regras foram: Não bater nos obstáculos ou na faixa de limite do labirinto, senão deveriam voltar ao início. Os participantes do grupo deveriam anotar as instruções dadas ao representante. Cada etapa foi cronometrada. Só foi permitida a presença na sala de um grupo por vez, os demais aguardaram fora do Laboratório. Utilizou-se fita crepe, cadeiras, cesto de lixo vazio e fios grossos de náilon. Os grupos executaram o percurso com tempos bem próximos, mas percebeu-se que utilizaram estratégias diferentes para conduzir o seu representante dentro do labirinto. Todos os alunos gostaram da atividade, e conseguiram compreender o funcionamento do computador.

Na 3ª aula, foi realizada a atividade *Memorização*. Para representar o armazenamento e processamento de informações pelo computador, no momento de busca de arquivos solicitada pelo usuário, a atividade III consistiu em: Os representantes dos grupos observaram a sala, saíram e aguardaram enquanto a sala era reorganizada. Em seguida, um representante do grupo entrava no laboratório e deveria encontrar quais objetos haviam sido modificados. Tudo o que estava no laboratório podia ser modificado, até mesmo os alunos que lá permaneceram. O grupo não podia dar dicas sobre o objeto modificado. Foram realizadas 4 etapas (1 objeto, 2 objetos, 3 objeto, 4 objetos). Cada representante teve 1 minuto para executar cada etapa. Nesta atividade utilizou-se cadernos, bolsas, cadeiras, casacos, mochilas, entre outros objetos que estavam na sala. Os alunos conseguiram abstrair os conceitos de busca, informação, memória e processamento presentes na atividade.

Na 4ª aula, a atividade consistiu em um mapa sequencial, composto por 8 pistas (questões) contendo problemas matemáticos e charadas a serem resolvidos, cada pista estava replicada em três cópias, tendo em vista que a turma foi dividida em 3 grupos (1 cópia para cada). As pistas foram escondidas em locais próximos ao Laboratório, na área interna da escola (quadra e pátio). Cada grupo precisou resolver o problema proposto em cada pista para passar à etapa seguinte. Foram utilizadas fichas impressas contendo os problemas a serem solucionados e fita crepe. Todos os grupos conseguiram resolver os problemas propostos, mas apenas um obteve um desempenho melhor considerando o tempo de conclusão da atividade.

Na 5ª aula, apresentou-se o conteúdo a ser estudado durante o módulo *Scratch* e iniciou-se a primeira atividade. Devido os computadores estarem sem acesso à internet, foi necessário reformular a aula que teria como objetivo a resolução do desafio A Hora do Código do site code.org/learn que objetiva demonstrar que a programação pode ser ensinada de maneira lúdica, por meio da utilização de uma linguagem de blocos. Pensou-se então em executar as etapas do desafio Angry Birds da Hora do Código

utilizando a Computação Desplugada.

Dessa vez, a turma foi separada em dois grupos, todos os componentes do grupo deveriam escrever o código para que o seu ator o executasse. Para simular as 20 etapas propostas no desafio, o caminho a ser percorrido foi demarcado utilizando fita crepe (Figura 1 (a) e (b)), enquanto para apresentar os blocos de comando, explicar as suas funções e mostrar as sugestões para a resolução das questões, utilizou-se apresentação de slides. Nesta aula apresentou-se os blocos de evento (quando executar), movimento (avance) e controle (repita (x) vezes). Foram utilizados o computador interativo, fita crepe, caneta e papel ofício. Os alunos conseguiram compreender a lógica utilizada para conseguir resolver as 8 primeiras questões do desafio.

Na 6ª aula, a atividade realizada foi uma continuação da aula anterior. Sugeriu-se que os alunos utilizassem um novo bloco de repetição apresentado, o Repita Até, cujo objetivo é que o ator percorra todo o trajeto seguindo poucas instruções. Assim como na 5ª aula, utilizou-se o computador interativo, fita crepe, caneta e papel ofício. Apesar de perceber-se algumas dificuldades em interpretar as situações propostas, os alunos tiveram um bom desempenho nas questões 9 a 12, após as seguidas explicações sobre o bloco de repetição Repita até, conseguiram compreender o objetivo da sua utilização. Na Figura 1, pode ser vista a dinâmica da aula.

(a) **(b)**

Figura 1. Alunos durante a atividade (a). Caminho demarcado com fita crepe (b).

Na 7ª aula, deu-se continuidade às atividades de movimentação, agora com novo cenário, a turma foi dividida em duplas. Nesta aula, para alcançar os objetivos foram utilizados blocos de comandos mais complexos que o da movimentação básica. Os alunos escreveram os códigos no caderno, em seguida executaram no software *Scratch* o algoritmo, uma dupla por vez. Nesta atividade, foram utilizados 2 notebook's e o computador interativo. O uso de novos blocos de comando, semelhantes ao usado no módulo de Computação Desplugada, foi concluída sem maiores dificuldades.

Na 8ª aula, foi apresentado um novo cenário às duplas, explorando também o uso dos blocos dos Sensores e de Controle, com a condição de que se o ator tocar na borda da pista deve voltar ao início. Os alunos escreveram os códigos com a resolução do problema no caderno e depois executaram no *Scratch* (no computador) o algoritmo escrito, uma dupla por vez. Nesta atividade, foram utilizados 2 notebook's e o computador interativo. Foi compreendido o uso dos blocos necessários para concluir a atividade.

As aulas de Iniciação à Programação com a metodologia Computação Desplugada, proporcionaram aos alunos momentos de descontração, colaboração, trabalho em equipe e principalmente de aprendizado. Em diálogo com a turma, o *feedback* sobre as atividades foi positivo, com pedidos de que houvesse continuidade.

Alguns alunos destacaram dificuldades que tiveram durante a realização de algumas atividades, mas informaram que no decorrer das aulas foram superadas. Essa primeira etapa da Oficina foi realizada durante 8 aulas (1h por aula).

Com o objetivo de desenvolver habilidades de autonomia, criatividade e trabalho em grupo, a etapa seguinte da Oficina foi a formação de grupos para a produção de um projeto. Cada grupo precisava escolher um tema com fins educacionais. No entanto, os grupos eram responsáveis por escolher como os projetos seriam desenvolvidos no *Scratch*, por meio da criação de jogos, histórias ou mesmo animações.

Foi solicitado que cada grupo justificasse a escolha do tema do seu projeto. Foram formados cinco grupos (três grupos com quatro integrantes, um grupo com três integrantes e um com cinco integrantes). As etapas de desenvolvimento foram as seguintes: 1ª Pesquisa na internet do material para a elaboração do projeto; 2ª Escolha de cenários e personagens; 3ª Estruturar os códigos do projeto. 4ª Acertos finais do projeto, revisão de código e funcionamento; 5ª Apresentação. O desenvolvimento do projeto teve duração de dois meses (outubro e novembro de 2016, respectivamente), foi realizado no laboratório da escola, no horário da oficina.

3.2. Descrição dos projetos desenvolvidos pelos alunos

O grupo 1, com o projeto Pré-conceito (Figura 2), buscou abordar em forma de um jogo de perguntas e respostas situações de preconceito racial, de estereótipo, deficiência e xenofobia. Em cada questão, o jogador deve julgar a atitude correta a ser tomada pelo personagem diante da situação evidenciada. Ao final, é dado o *feedback* sobre o desempenho do jogador, e a pontuação final obtida.

Figura 2. Projeto Pré-conceito.

O grupo 2, criou um jogo com cenários e personagens intergaláticos com o objetivo de revisar conceitos do Sistema Solar. O projeto *Space Gari* (Figura 3), conta a história do rei Ota que foi aprisionado por inimigos, para que o jogador ajude o personagem Kalígari a libertá-lo, deve responder corretamente às questões sobre os planetas do Sistema Solar. O Space Gari contém 3 fases, cada fase 5 questões, são disponibilizadas 3 tentativas em cada questão. Ao final da última fase existe 1 desafio, ao vencer o desafio, rei Ota é libertado.

Figura 3. Projeto Space Gari.

O grupo 3, com o projeto Quiz Biology (Figura 4), optou por criar um *Quiz* que contemplasse conteúdos estudados do 1º ao 3º ano do ensino médio, justificando que o projeto proporciona a partir da resolução das questões uma breve revisão de forma atrativa. Para cada série foram selecionados 3 temas, e para cada tema 5 questões. Os conteúdos são: 1º ano - Sistema muscular, Órgãos dos sentidos e Introdução à biologia; 2º ano - Desenvolvimento embrionário, Fotossíntese, Mitose e Meiose; e 3º Teorias da evolução, Base sobre hereditariedade e *Homosapiens*. A cada resposta correta o jogador ganha 20 pontos, caso contrário, não ganha pontos, entretanto recebe a informação sobre a resposta correta.

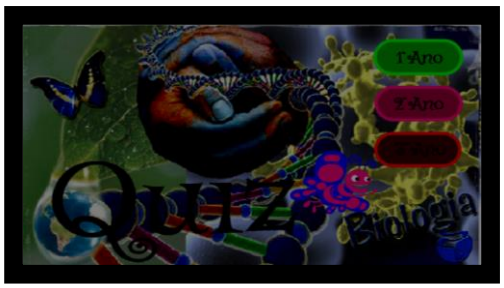


Figura 4. Projeto Quiz Biology.

O grupo 4, criou o projeto intitulado Quiz Testosterona (Figura 5), cujo nome foi escolhido por ser um grupo formado por rapazes, um Quiz contendo 4 questões para cada tema, cujos temas são Biologia, Matemática, Geografia e Conhecimentos Gerais. Para que o jogador selecione a quais questões quer responder, basta clicar no personagem correspondente ao tema. A cada questão respondida corretamente o jogador ganha 10 pontos. Ao final do Quiz é apresentado um Ranking contendo o nome dos jogadores, pontuação e a classificação.

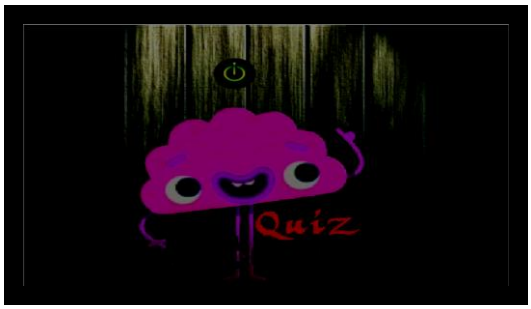


Figura 5. Projeto Quiz Testosterona.

O grupo 5, criou um jogo da Tabuada (Figura 6) com o objetivo de estimular a aprendizagem de cálculos básicos de maneira lúdica e desafiadora. Ao clicar na bandeira verde do *Scratch,* inicia-se o tempo de 60 segundos, durante esse tempo são informados 2 valores gerados aleatoriamente e a operação matemática. O jogador precisa responder o resultado do cálculo. A cada resposta correta o placar de pontos soma 1 e a cada erro subtrai 1.

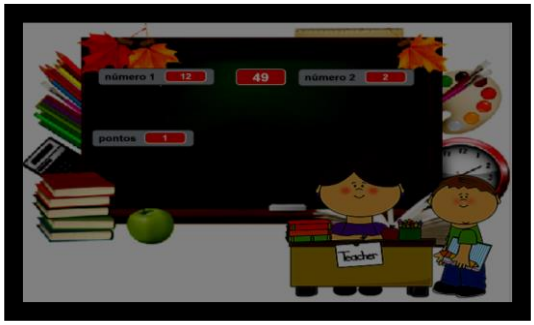


Figura 6. Projeto Tabuada.

Na primeira semana de dezembro os grupos apresentaram os projetos mostrando o funcionamento, tendo a participação dos demais no momento da execução, falaram sobre as dificuldades no desenvolvimento e mostraram os blocos de comando utilizados. Na segunda semana de dezembro de 2016 a apresentação dos projetos foi realizada para a escola e comunidade. Todos os componentes dos grupos se envolveram na apresentação, explicando o funcionamento do projeto, objetivos, e convidando as pessoas para conhecerem o que eles haviam produzido. Foram elogiados, principalmente pela escolha dos temas. Pode-se ver a empolgação e participação dos alunos, o reconhecimento por parte da gestão da escola às atividades de Informática realizadas, pois percebe-se a relevância que o ensino da Informática trouxe aos envolvidos.

3.2.1. Lições aprendidas

As oficinas realizadas, em 2015 utilizando o *Scratch*, e em 2016 utilizando a Computação Desplugada, foram de suma importância tanto para mostrar que é possível desenvolver ações desse tipo na escola pública quanto para enfatizar as perspectivas de ensino da Informática por meio dos recursos computacionais. Os processos metodológicos utilizados na edição 2016 foram consequência da avaliação feita pelos alunos da edição 2015 (MARINHO *et. al*., 2017), e certamente corroborou com a saída da zona de conforto, para traçar novas estratégias.

Tendo em vista as metodologias distintas que foram utilizadas, torna-se necessário fazer uma breve consideração sobre as estratégias que demonstraram-se mais eficazes. No caso da turma de 2015, os alunos mostraram-se interessados, pois estavam conhecendo a ferramenta, dessa forma, tratavam cada bloco estudado com empolgação. Porém as dificuldades de compreensão da lógica para a implementação dos projetos, eram superiores comparadas à turma de 2016. Utilizando a Computação Desplugada, os alunos, sentiam-se desafiados a resolver os problemas propostos, e demonstravam melhor compreensão da lógica de programação na execução das atividades.

4. Considerações finais

A avaliação feita por parte dos alunos, por meio do questionário aplicado ao final da Oficina, mostrou que os objetivos da proposta foram alcançados, por mais que tenham havido algumas dificuldades na resolução dos desafios que foram propostos durante o percurso. A análise das respostas foi fundamental para o planejamento da segunda edição da Oficina. Em 2016, no final de mais uma Oficina, os objetivos foram concretizados novamente. Os cinco grupos apresentaram os seus projetos na Feira de Arte e Cultura (FESTAC) - O evento acontece anualmente e tem por objetivo envolver a escola em situações do cotidiano através de projetos que foram desenvolvidos durante o ano letivo. Dessa vez os projetos abordaram temáticas relacionadas às disciplinas de Biologia, Geografia, Matemática e História. Sendo assim, diante do relato apresentado neste trabalho pode-se inferir que existe a possibilidade de implementação de atividades de Informática/computação nas escolas públicas, mesmo diante dos desafios constantes.

A experiência relatada corrobora com a ideia de que a computação dispõe de diversos recursos, os quais podem trazer contribuições significativas ao processo de ensino e aprendizagem. Este trabalho visa mostrar que mesmo com a utilização tímida de um software de iniciação a programação, é possível a operacionalização de um trabalho didático-pedagógico significativo. Considerando a importância dos estudantes

desenvolverem habilidades de resolução de problemas, e tendo em vista que é possível mobilizar essas habilidades por meio de atividades que envolvam situações do dia a dia, ou de conteúdos relacionados às disciplinas curriculares. Entende-se que é relevante tornar a aprendizagem mais significativa durante o processo de ensino.

A partir da análise dos resultados alcançados nas duas oficinas, pretende-se ampliar as ações de Iniciação à Programação na Educação Básica. Visto que, assim como as experiências realizadas no Ensino Médio tiveram sucesso, busca-se implementar tais ações também no Ensino Fundamental. De tal forma, possibilitar que mais estudantes tenham acesso aos conhecimentos operacionalizados em atividades que envolvam recursos computacionais.

Referências

Bini, E. M., Koscianski, A. (2009) "O ensino de programação de computadores em um ambiente criativo e motivador". In: Encontro Nacional de Pesquisa em Educação em Ciências. Florianópolis/SC.

Cavalcante, A. F. et al. (2016) "Um Estudo de Caso Sobre Competências do Pensamento Computacional Estimuladas na Programação em Blocos no Code.Org". In: Anais dos Workshops do V Congresso Brasileiro de Informática na Educação. Uberlândia/MG.

Code Studio. (2017) "About Code Studio", http://code-studio.com/about-us/, Novembro.

Gomes, T. C. S; Tedesco, P. C. DE A. R; Melo J. C. B. (2016) "Jogos no Design de Experiências de Aprendizagem de Programação Engajadoras". In: V Congresso Brasileiro de Informática na Educação (CBIE 2016). V Jornada de Atualização em Informática na Educação (JAIE 2016). Uberlândia/MG.

Libâneo, J. C. (2000) "As novas tecnologias da comunicação e informação, a escola e os professores". In: Adeus professor, adeus professora? Novas exigências educacionais e profissão docente. Cortês: São Paulo.

Marinho, A. R. da S. *et. al*. (2017) "O uso do Scratch na Educação Básica: Um relato de experiência vivenciada no PIBID. In: XXIII Workshop de Informática na Escola, Recife/PE.

Scratch. (2015) "About Scratch", http://scratch.mit.edu/about, Janeiro.

Silva, F. B.; Romani, R., Baranauskas, M. C. C. (2008) "Soo Brasileiro: Aprendizagem E Diversão". No Xo. Revista Brasileira de Informática na Educação, p.29-41.

Tripp. David. (2005) "Pesquisa-ação: uma introdução metodológica". In: Educação e Pesquisa, São Paulo, v. 31, n. 3, p. 443-466, set./dez.

Vieira, A.; Passos, O.; Barreto, R. (2013) "Um relato de experiência do uso da técnica computação desplugada". In: SBC 2013, pages 670–679.

Vygotsky, L. S. (1987) "Thinking and speech" (N. Minick, Trans.). In R. W. Rieber & A. S. Carton (Eds.), The collected works of L. S. Vygotsky: Vol. 1. Problems of general psychology (pp. 39-285).

Wing, J. M. (2006) "Computional Thinking". In: Communications of the ACM. March 2006/Vol. 49, No.3.

MetricRA: Learning Software Metrics through Augmented Reality

Rebeca Motta[1], Mario Bonicenha[1], Claudia Susie Rodrigues[1], Cláudia Werner[1]

[1]Computer Systems Engineering Programm at Federal University of Rio de Janeiro
Rio de Janeiro, Brazil

{rmotta, mario.bonicenha, susie, werner}@cos.ufrj.br

Abstract. *Augmented reality creates a bridge between virtual and real world, providing stimulating resources for different purposes. This technology enables new teaching possibilities since it can bring more abstract concepts into reality and put the knowledge related to several areas, such as Software Engineering, into practice. MetricRA is a tool developed to help Software Engineering students to understand Cohesion and Coupling metrics. The solution was implemented with Augmented Reality technology, where the user can control a class diagram to observe the metrics transformation. This article describes MetricRA tool and presents a study conducted to evaluate its ability to contribute to the understanding of the concepts proposed.*

1. Introduction

With the technological evolution, we can see the increasing availability of products and applications using Augmented Reality (AR). This technology can be applied for different purposes, in education, for instance, enables a different interaction that can help improve the learning experience [Wu *et al.* 2013]. This new option has already been used in Software Engineering, mainly assisting the understanding of software systems [Rodrigues, Werner, & Landau 2016]. This work aims to contribute to teaching Software Metrics. As software becomes more pervasive, metrics concerns are critical, being an important step towards developing quality software [Fenton & Pfleeger 1997].

In this scenario, the MetricRA tool, proposed in this work, focuses on the student experience and aims to help his/her learning regarding the concepts of Cohesion and Coupling Metrics. We define Cohesion as the relative functional robustness of a module, and Coupling as the relative interdependence between the modules [Pressman 2014]. The internalization of these concepts by the students is essential, since their conscious use, throughout the development, results in systems that have a better modular organization. The choice of these metrics came from two perspectives: education and technology. From teaching, the choice was due to the importance of these metrics in software design and architecture, as they affect the maintenance and organization of the system. From the technology, the visualization of the metrics is presented in an interactive way where AR potential can be exploited. The MetricRA uses AR to apply the metrics in a practical way, allowing the student to modify the structures of a diagram, analyzing Cohesion and Coupling metrics, and visualizing the results of the changes in a direct way.

2. Theoretical Background and Related Work

In this session, a theoretical background is presented regarding AR and Software Metrics. These concepts are the foundation of this work and are interrelated as the basis for the solution proposed by the tool. We also present Related Work regarding software visualization solutions, showing how the MetricRA tool fits in this research area.

2.1. Augmented Reality (AR)

AR relies on user interaction to supplement the real world with virtual objects that seem to co-exist in the same space. This technology is known to increase the interest of the user, thus bringing more attention to what is being manipulated in the application, which provides a great educational value. By using new strategies for passing knowledge and stimulating other senses, it is possible to collaborate in the cognitive process of the student, bringing also practical experimentation of content [Jin & Yano 1997].

With this technology, the content is not limited to the textual format, and the connection to the real world can be through graphics, images and videos, broadening the student's perception. AR´s goal is to enrich the physical environment with real-time virtual information. Therefore, interaction is a fundamental part of this technology. In the development AR applications, an interface between the user and the virtual environment is exploited to provide that experience to the user. This interface can be done in several ways. A rather simple and widely used option is through markers, where the user points a camera (e.g. mobile phone) to the marker and the application displays its corresponding 3D information overlapped with the marker. From it, virtual elements are combined with real things enabling a different kind of the interaction. Another example of AR can be recognized through the successful game, Pokémon Go.

2.2. Software Metrics

Metrics represents an important concept in Software Engineering, defining a quantitative measure of the degree to which a system, component or process has for a given attribute [IEEE 1990]. These measures can be applied to the product, artifacts or documents produced during the software life cycle, which can be evaluated in different ways, as to their internal or external quality.

In this paper, we explore the Cohesion and Coupling metrics, which are measures related to the software design that contribute to its internal quality. The concepts of Cohesion and Coupling, although having a great impact on the internal quality of systems, end up being overlooked by beginners and also by users who are often interested mainly in the interface and utility of the system [Fenton & Pfleeger 1997]. Therefore, teaching these concepts, in a direct and practical way, can contribute to software quality and to the understanding of their importance. The Cohesion metric measures how focused the class responsibilities are in meeting a specific goal. A highly cohesive class has clear and well-defined responsibilities, while a class with low Cohesion has many different and unrelated responsibilities. The Coupling metric refers to how much a functional unit depends on another, that is, the relationship between the parts of a system. The development of software with high Cohesion and low Coupling contributes to, among other things, maintenance and reuse [Chidamber & Kemerer 1992].

2.3. Related Work

The purpose of this work is to introduce a tool for visualizing software, especially regarding software metrics. As an example of related work we have a survey of visualization techniques, both in 2D and 3D, representing the static aspects of software [Caserta & Zendra 2011], and a review which identifies different solutions to visualize software metrics, including an analysis of the visual attributes and interaction mechanisms [Titang & Dur 2015]. Through these works, it is possible to observe the advances and current challenges in the area, and how our work fits in that context. A similar work to MetricRA is presented by Churcher *et al*. [Churcher, Irwin, & Kriz 2003], where the concept of Cohesion of a class is presented by a 3D image, enriched by size, shape and color. As a result, the visualization of the class takes an abstract format, like chemical molecules, representing the relationship of its elements. The differential of our work regarding the others is given by the technology used and the purpose. With AR, we achieve visual expressiveness and the capability to adapt to the user's interactions in real time, unlike a 2D image. This proposal offers more interactivity than the current related work and has a concern with education alongside with visualization.

3. MetricRA Tool

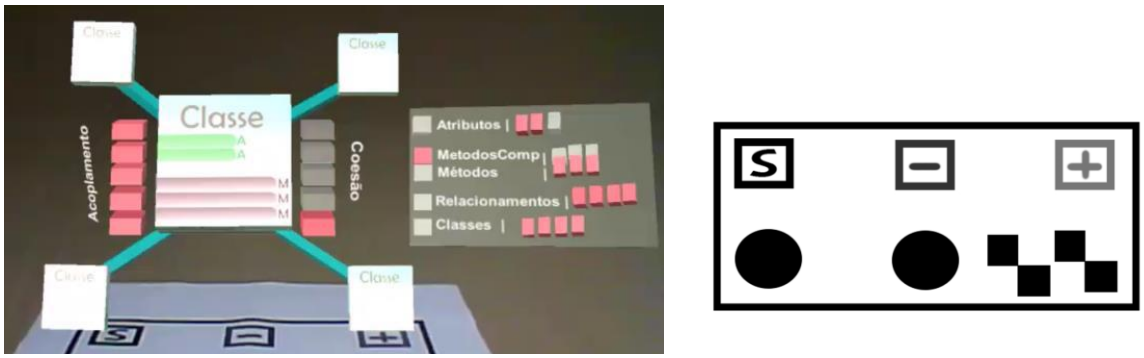
MetricRA consists of an application that uses AR to help students understanding the concept of Cohesion and Coupling metrics. The main objectives of the tool are to (i) help students internalize the concepts; (ii) reduce the gap between theory and practice through the use of contextual examples; (iii) provide immediate feedback for a rapid assimilation of knowledge; (iv) provide an exploratory and intuitive environment through students' interaction; (v) be simple and easy to use. To understand a concept, the best way is to see it in practice and AR can contribute to this. AR is widely applied to simulation and makes the learning process more dynamic. Among its main advantages are immediate feedback and continuous evaluations [Wu *et al*. 2013]. For the development of the tool, Unity5 3D[1] engine was used, as the development platform; and Vuforia[2], an extension to develop AR applications. The resources needed to run the tool is a computer equipped with a webcam. The tool was implemented in C# language.

Figure 1A shows an image of the MetricRA tool execution. In the center of the image, there is the main class that relates to other classes. On the right, there is a menu of options and diagram structures that the student can interact with. With the help of AR, it displays a class diagram overlapped on a marker. Through the menu, one can change the structure of the diagram, for example: change the number of attributes and methods. According to the chosen option, the tool changes the diagram, recalculates and displays Cohesion and Coupling, indicating their satisfaction level that varies from satisfactory to unsatisfactory, in green and red colors respectively.

In Figure 1A, the main class has low Cohesion (a red cube on the right) meaning that the class is not converging only to one main goal (unsatisfactory). The coupling, on the other hand, is high level (five red cubes on the left), meaning that it is not satisfactory since it is depending directly on many other classes. In the example, the class has two attributes and three non-shared methods.

[1] https://unity3d.com
[2] https://developer.vuforia.com/

In MetricRA, a marker was used (Figure 1B). The marker is used to change the structures and to select the options in the menu. It is possible to increase or decrease the number of structure items. The marker has three buttons: "+" which adds the selected item in the right menu, "-" to decrease and "S" to select one of the menu options.



A B

Figure 1. A - Image of the MetricRA tool execution. B - Tool Marker.

For didactic reasons, the metrics used simple formulas calculated according to the diagram structure: methods, shared methods, relationships and number of classes. The Cohesion metric was calculated according to LCOM1 and LCOM2 (Lacks of Cohesion Metrics) [Chidamber & Kemerer 1992], which considers the information shared between the methods. The formula used for Cohesion was: 1-(SM)/(TM*SA), where SM stands for Shared Methods, TM means the number of Total Methods and SA, the number of Shared Attributes between the methods. For Coupling, a C/R ratio was used, where C is the number of Classes and R is the number of Relationships of that class. Since it is a classical approach for metrics, we chose the LCOM CK suite. Different works redefine them but remain based on the original proposal. In the tool is possible to add other metrics and redefine the current ones, by adjusting the formulas.

4. Evaluation

The effectiveness of new tools to support teaching and learning should be evaluated in order to observe their effects on users [Von Wangenheim, Savi, & Borgatto 2012]. In addition, since AR is a recent technology that allows unconventional interaction, it is essential to analyze its quality, mainly regarding the usability of its applications [Martins, Paulo, & Correa 2013], before being applied in the classroom. Thus, a feasibility study was initially conducted with the objective of analyzing the tool support to understanding the Cohesion and Coupling metrics, through the real-time interactivity offered by AR, regarding to motivation effectiveness, user experience and learning aid.

4.1 Planning

In the study, we aim to observe the ability of the MetricRA to support the understanding of Cohesion and Coupling metrics in a different experience through the use of AR. Based on GQM (Goal-Question-Metric) [Basili, Caldeira, & Rombach 1994], the objective of the evaluation is: to analyse *the MetricRA tool* with the purpose of *characterizing it,* in relation to *its motivation effectiveness, user experience and learning aid,* from the point of view of *undergraduate students,* in the context of *teaching metrics in Software Engineering.*

As a basis for the evaluation material, the methodology proposed by Savi *et al*. [Savi, Wangenheim, & Borgatto 2011] was used. This methodology is focused on the evaluation of the quality of educational games, however, the material includes contents such as the evaluation of training [Kirkpatrick & Kirkpatrick 2009], motivational strategies [Keller 1987] and, also consider user experience, of great importance in the AR context. The adaptation of the methodology for the evaluation of the tool considered three main aspects: **Motivation**, **User Experience** and **Learning**, which seek to reflect the evaluation goals. From the original proposal [Savi *et al*. 2011], some aspects outside the scope of the tool, such as Challenge and Social Interaction, were not considered.

The evaluation form presented 15 statements that considered the three aspects to be evaluated. To indicate their experience, they evaluated these statements on a 5-point Likert scale: (-2) strongly disagree, (-1) disagree, (0) neither agree nor disagree, (+1) agree and (+2) strongly agree, with the freedom to include comments on each item, if necessary. As for the learning aspect, the participant evaluated his/her perception of the concepts covered by the tool before and after the experience. The last part of the evaluation had space for the participant to express the positive and negative points of the tool so that it was also possible to identify improvement opportunities.

4.2 Execution

All participants signed a Consent Term and a Characterization Form (completed before interacting with the tool). The participants received a numeric code for anonymization and received the same instructions in video format. After that, participants had the opportunity to interact with the tool and then carry out the proposed tasks, individually and without any contact with each other.

Pilot Study. Performed by a member of the Virtual and Augmented Reality laboratory, to observe the suitability of the study. From the results, we made some improvement, such as a better explanation, as well as clarity in the presented virtual elements.

Proof of Concept. The participants were chosen for convenience and were students of the Computer Science course. Seven participants performed the assessment and they reported that the video was fast and with poor quality audio. From these comments, some details were added to the tool and the video updated.

Observational Study. The study was carried out in April 2017 with the 10 undergraduate students in computing, who participated voluntarily in response to invitations published in students mail list. The analysis of results, presented in the next subsection, refers to the data collected in this study. The explanatory video used is available[3], as well as the executable file and evaluation package[4].

4.3 Analysis and Results

Characterization. All participants had graduation in progress, chosen by convenience, aiming at a group that would have greater gain with the tool. Among the participants, 60% reported having experience with research and 60% reported from 1 to 3 years of experience in the market, with an overlap of 2 participants with experience in both. Most participants reported a low level of experience with software metrics (70%), with

[3] https://goo.gl/UaUEyk
[4] https://goo.gl/nYWDKj

30% having no experience at all (Figure 2). When asked specifically about Cohesion and Coupling metrics, 50% of participants reported having little knowledge about it. As for Class Diagrams, 40% reported that the experience was obtained through classroom projects, and 30% in personal projects.

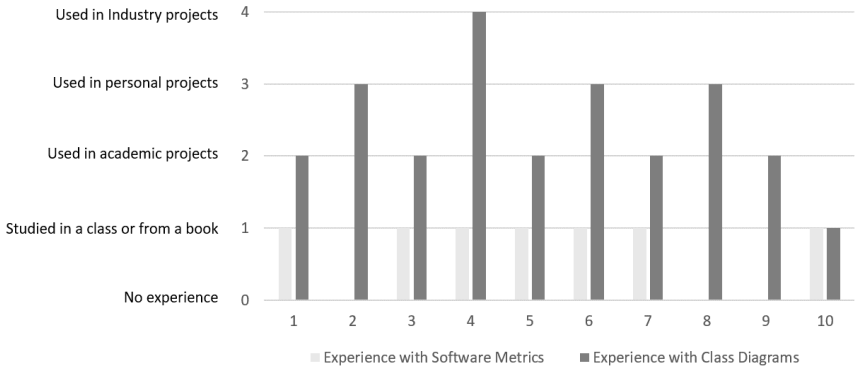


Figure 2. Participants' experience in Metrics and Class Diagrams.

Tasks Execution. To encourage interaction, the participants performed 5 tasks with the support of the tool. Tasks seek to contribute to the sedimentation of related concepts learned before. The participants were asked to interact and answer the following questions while using the tool: a) cite what information is shown in the main class and which structures are, in fact, significant for the Cohesion and Coupling metrics; b) list two ways of achieving high cohesion and two ways of achieving low coupling; c) describe how the metrics can affect the quality of a system. Participants then answered each question and assessed their difficulty level on a scale ranging from 1 (very easy) to 4 (very difficult), as shown in Figure 3. Each participant is represented by their code (from 1 to 10), showing the level of difficulty considered for each task (from very easy to very difficult) and the index of success (the incorrect tasks are marked with "wrong"). The average time for the execution was 28 minutes. The hit rate was very high, with 82% of the tasks performed correctly. This step is necessary so participants can experience the tool with the same orientation, enabling a uniform observation.

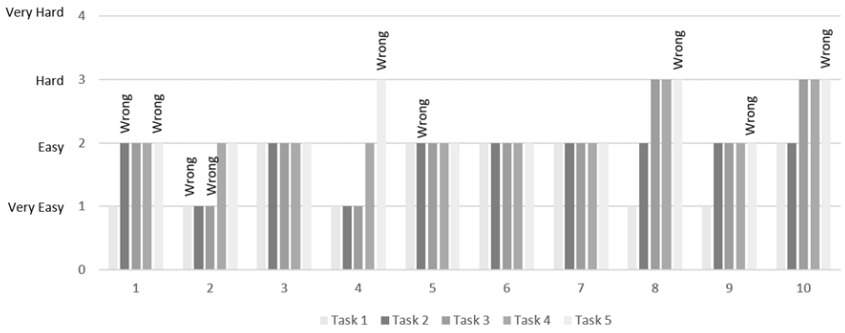


Figure 3. Tasks Results and Difficulty Level.

MetricRA Evaluation. The evaluation of the tool was performed by the 10 participants through a questionnaire adapted from [Savi *et al*. 2011]. The evaluated aspects were divided into three categories: **Motivation**, **User Experience** and **Learning**, according to the research objectives. We understand that these concepts are abstract and may have more than one interpretation, however in this article each category, and their sub-characteristics has been defined based on the protocol used.

The **Motivation** to learn is essential for teaching and the use of motivational strategies seeks to bring an engagement. The Motivation aspect, based on the model used, counts on the sub-characteristics of **Satisfaction**, **Relevance** and **Attention**. **Satisfaction** refers to a positive feeling about the effort in the learning experience. The **Relevance** tries to observe the consistency between the content taught with its utility. **Attention** is related to the cognitive responses to what is being taught. These aspects were translated into seven statements assessed as presented in Figure 4.

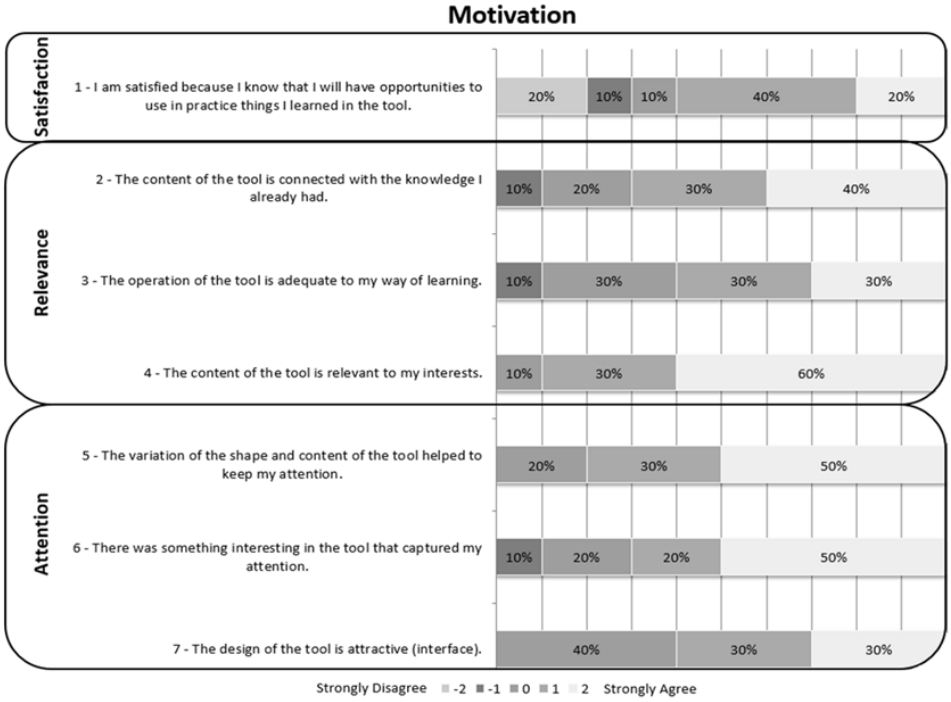


Figure 4. Evaluation of the Motivation category.

Satisfaction was met by 60% reflecting the opportunities to use the concepts treated in the tool in practice. For abstract concepts, such as metrics, aligning practice with theory is a good strategy since it brings the student closer to the concepts. The **Relevance** aspect obtained a high evaluation in all items (with 70%, 60% and 90% in items 2, 3, and 4, respectively). The last item shows the content relevance as to its suitability to the student's academic interests. For **Attention**, the evaluation was also high in all items (with 80%, 70% and 60% in items 5, 6, and 7, respectively). By analyzing the reported comments, participants thought the interface was interactive and attractive (participants 3, 4 and 8), but that there was some instability in the interaction, "deconcentrating" (participant 7), identified as an improvement opportunity.

Figure 5 presents the results of students' responses related to the **User Experience** category. The concept of experience is individual and somewhat abstract, which may hinder its description and evaluation. However, it was of interest to contemplate this concept through the sub-characteristics of **Competence**, **Amusement** and **Immersion**. To provide a good experience, the tool seeks to support the understanding of the concepts by recognizing the participants' previous knowledge, affirming their **Competence**. The **Amusement** aspect seeks to reflect feelings such as distraction and satisfaction. **Immersion** is an important sub-characteristic for an AR tool because it proposes a deviation of focus from real to virtual world.

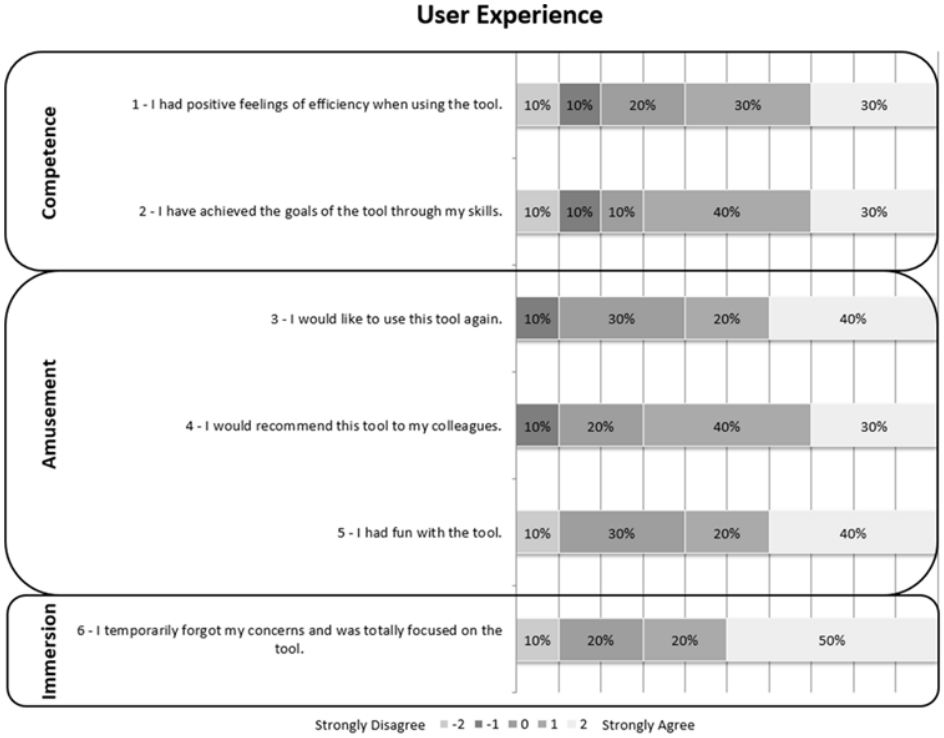


Figure 5. Evaluation of the User Experience category.

In the **Competence** aspect, the evaluation had in average 60% and 70% in the evaluated items. Participants with discordance ratings (10% in each item) left no comment. In **Amusement**, the rating was high on all items (with 60%, 70% and 60% in items 3, 4, and 5, respectively) with positive feedbacks in the comments, such as the tool was "fun and very well illustrated" (participant 8). As for **Immersion**, 70% of the participants agreed that they were totally concentrated on the tool, with positive comments such as: "AR contributed to my interest and concentration" (participant 4).

Finally, items related to **learning** were evaluated, where the items received 70% and 90% of agreement, respectively (Figure 6). One can observe the positive results of the evaluation regarding the efficiency in learning the concepts of metrics.

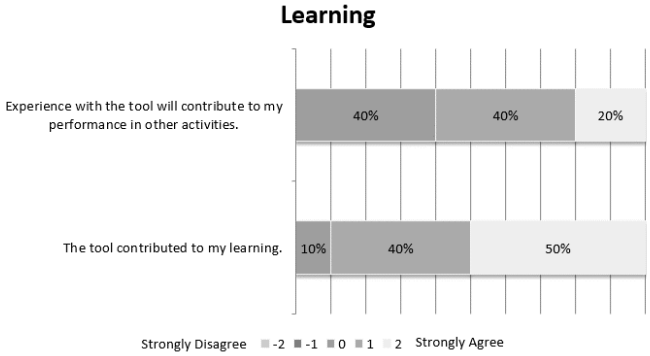


Figure 6. Evaluation of the Learning category.

Self-evaluation. A self-evaluation to assess the learning, comparing how it was before and after the tool (Figure 7). The evaluation consists of assigning a grade within a scale ranging from 1 (little) to 5 (very) the level of knowledge of the concepts presented: Coupling, Cohesion, Main Class, Attributes and Methods.

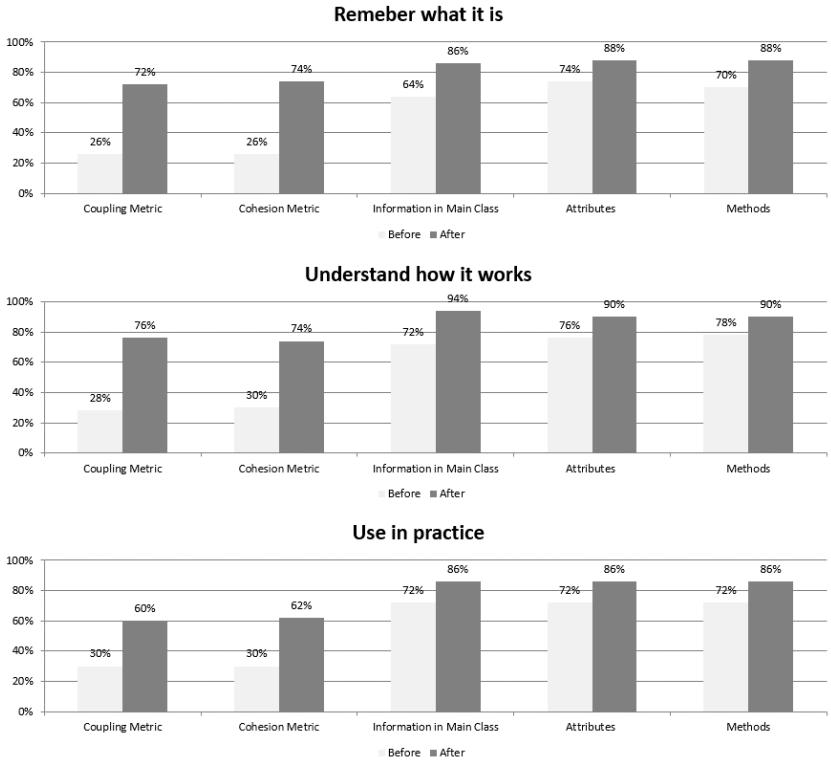


Figure 7. Self-evaluation measures on learning objectives.

It concerns three perspectives related to the concepts: (a) remember what it is; (b) understand how it works; (c) use in practice. Some comments such as "it is clear how to calculate the metrics" (participant 3) show the contribution of the tool. They had a positive result in the tasks (73% accuracy) and, in the self-evaluation, all reported some improvement. Figure 7 shows an increase in the concepts presented (gains of 60%).

4.4 Threats to Validity

It is important to highlight some threats to validity [Wohlin *et al.* 2012]. For conclusion validity, the sample size, chosen for convenience, and the limited experience in relation to the concepts are identified. As mitigation for the later, the instructions were in video, avoiding training differences. From the qualitative perspective, the participants' comments are worthy, revealing improvement opportunities. Despite that, they are opinions, a limitation on external validity, being not possible to generalize the results.

5. Conclusion

The MetricRA has aims to help in the understanding of Cohesion and Coupling metrics. To this end, the tool uses AR technology, allowing changing diagram structure, in real time, so the student can control and internalize the concepts. An observational study was performed with 10 participants. The results were positive with 82% of the tasks performed correctly and an increase of the level of knowledge, indicated by a self-evaluation performed by the students themselves. From this study and the presented results, we can observe that some objectives of the tool were achieved (help students internalize the concepts and provide an exploratory learning environment). This is an initial effort to evaluate the tool in relation to its motivation, user experience and learning aid, from the point of view of undergraduate students. As future activities we

intend to complement it and improve the data significance by comparing gain aspects using a baseline scenario in comparison with the tool. For this paper, the focus was on the tool and the metrics proposed, not being addressed the perspective of structural complexity, an option for future work.

References

Basili, V. R., Caldeira, G., & Rombach, H. D. (1994). Goal Question Metric Paradigm.

Caserta, P., & Zendra, O. (2011). Visualization of the static aspects of software: A survey. *IEEE Transactions on Visualization and Computer Graphics*, *17*, 913–933.

Chidamber, S. R., & Kemerer, C. F. (1992). a Metrics Suite for Object Oriented Designa Metrics Suite for Object Oriented Design. *PhD Proposal*, *1*(6), 476–493.

Churcher, N., Irwin, W., & Kriz, R. (2003). Visualising Class Cohesion with Virtual Worlds. In *INVIS.AU* (Vol. 24, pp. 89–97).

Fenton, N. E., & Pfleeger, S. L. (1997). *Software Metrics: A Rigorous and Practical Approach*. *It Professional* (Vol. 2). International Thomson Publishing.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990). *IEEE Computer Society*.

Jin, Q., & Yano, Y. (1997). Design issues and experiences from having lessons in text-based social virtual reality environments. *Ieee SMC*, *2*, 1418–1423.

Keller, J. M. (1987). Development and use of the ARCS model of motivational design. *Journal of Instructional Development*, *10*(1932), 2–10.

Kirkpatrick, D. L., & Kirkpatrick, J. D. (2009). Evaluating: part of a ten-step process. *Evaluating Training Programs*, 3–20.

Martins, V. F., Paulo, S., & Correa, A. G. (2013). Usability Test for Augmented Reality Applications.

Pressman, R. S. . B. R. M. (2014). *Software Engineering A Practitioner's approach Eighth Edition*. Palgrave Macmillan.

Rodrigues, C. S. C., Werner, C. M. L., & Landau, L. (2016). VisAr3D. In *38th ICSE '16* (pp. 451–460). New York, New York, USA: ACM Press.

Savi, R., Wangenheim, C. G. von, & Borgatto, A. F. (2011). A Model for the Evaluation of Educational Games for Teaching Software Engineering. In *SBES* (pp. 194–203).

Titang, S., & Dur, A. (2015). The Visualization of Software Quality Metrics Bachelor of Science Thesis in Software Engineering and Management Dur Abuzaid.

Von Wangenheim, C. G., Savi, R., & Borgatto, A. F. (2012). DELIVER! - An educational game for teaching Earned Value Management in computing courses. *IST*, *54*(3), 286–298.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Wu, H. K., Lee, S. W. Y., Chang, H. Y., & Liang, J. C. (2013). Current status, opportunities and challenges of augmented reality in education. *C&E*, *62*, 41–49.

Um Relato de Experiência: Ensinando Robótica por Meio de Microcontroladores em uma Escola Profissional de Ensino Médio

Mateus Lima Alves, José Robertty de Freitas Costa, Paulo Miranda e Silva Sousa Paulo Ricardo da Silva Lopes, Carla Ilane Moreira Bezerra

[1] Campus de Quixadá – Universidade Federal do Ceará (UFC)
63.902-580 – Quixadá – CE – Brasil

{mateuslima.ti, robertty, paulomirandamss, prslopes1}@alu.ufc.br

carlailane@ufc.br

Abstract. *The enthusiasm for robotics and automation encourages many young people to participate in projects involving the construction of hardware and software. Microcontrollers are a "backbone" of robotics and automation projects. Such devices control and manage the other system components. For this, it is necessary to know in programming and programming, to become fundamental to the young developer. In this context, a proposed proposal was proposed as an initiative for the study of robotics. An action visa is part of the high school in the area of robotics, through a course of microcontrollers. This article aims to make an account of the extension action, with its planning, execution, results achieved and future work.*

Resumo. *O entusiasmo pela robótica e automação incentiva vários jovens a participarem de projetos que envolvem a construção de hardware e software. Os microcontroladores são a "espinha dorsal" de projetos de robótica e automação. Tais dispositivos controlam e gerenciam os demais componentes do sistema. Para isso, conhecimentos em eletrônica e programação, tornam-se fundamentais para o jovem desenvolvedor. Nesse contexto, uma ação de extensão foi proposta, como uma iniciativa ao estudo de robótica. A ação visa inserir alunos do ensino médio à área da robótica, por meio de um curso de microcontroladores. O presente artigo tem como objetivo fazer um relato da ação de extensão, apresentando seu planejamento, execução, resultados alcançados e trabalhos futuros.*

1. Introdução

Atualmente, os estudantes do ensino básico estão imersos em um ambiente em que a tecnologia é facilmente percebida: carros, celulares e computadores são exemplos que todos conhecem e muitos utilizam, no entanto, poucos entendem dessas tecnologias [Benitti et al. 2009]. Esse tipo de tecnologia é composto por componentes físicos, denominados *hardware*, que são coordenados por softwares que gerenciam esses recursos físicos. A união desses dois eixos gera um sistema automatizado e a compressão desse sistema compõe a robótica [Barrientos and Barrientos 2007].

A robótica é uma área relativamente recente que tem ganhado espaço no mercado. Essa área se envolve de maneira intrínseca, com outras grandes áreas como a eletrônica,

computação e mecânica. Esse domínio trata do estudo e desenvolvimento de sistemas automáticos e controlados por chips de circuitos integrados [Miyagi and Villani 2008].

No processo de ensino aprendizado, o aluno passa para a posição ativa quando abstrai um problema do cotidiano e compreende os mecanismos que o rege. A utilização da tecnologia nesse processo está apresentando resultados positivos em países da Europa e América do Norte. Nesses países, o ensino de programação e robótica é aplicado a problemas práticos, resulta na melhoria do nível de abstração de tecnologias e motiva jovens estudantes inseridos em vários contextos [Avila et al. 2016].

Na literatura brasileira existem vários relatos do ensino de robótica [Nesi et al. 2014, de Jesus and Cristaldo 2014, Neto et al. 2015]. Em um destes relatos, os autores defendem o ensino de robótica na grade curricular do ensino médio, como estímulo aos alunos seguirem a área de exatas [Nesi et al. 2014]. No entanto, o ensino de robótica exige equipamentos de hardware que podem ter alto custo. Dessa forma, o uso de microcontroladores, que são equipamentos de baixo custo, torna-se uma alternativa viável para o ensino de robótica em escolas de ensino médio púbicas.

Nesse contexto, o presente artigo apresenta um relato de experiência de uma atividade de extensão de ensino de robótica no ensino médio utilizando microcontroladores. O projeto foi idealizado pelo Programa de Educação Tutorial - Tecnologia de Informação Conexões de Saberes do Campus da Universidade Federal do Ceará (UFC) em Quixadá, e foi iniciado no ano de 2017 tendo como objetivo permitir a troca de experiências e conhecimentos entre estudantes de Engenharia de Computação e estudantes do Ensino Médio de uma escola profissional da região. O objetivo do projeto é inserir os estudantes do Ensino Médio na área da robótica incentivando-os a desenvolver projetos na área por meio de microcontroladores, que são dispositivos de controle, programáveis e de baixo custo. A abordagem de robótica com microcontroladores utilizada foi *down-top*, para que os alunos aprendam desde aplicações básicas como acender de um LED àquelas mais completas como a montagem do primeiro robô, um carro de controle remoto.

O texto está organizado da seguinte forma. A Seção 2 descreve a fundamentação teórica que embasou o projeto, contendo a base teórica do ensino de robótica e conceitos relacionados à utilização de microcontroladores. A metodologia aplicada no trabalho, o levantamento de demanda e a solução prática para o problema estão dispostos na Seção 3. O desenvolvimento da atividade está especificado na Seção 4. Os resultados do projeto são descritos na Seção 5. Por fim, na Seção 6, são apresentadas as conclusões e trabalhos futuros.

2. Fundamentação Teórica

Nesta Seção, são detalhados conceitos e trabalhos relacionados ao ensino de robótica e conceitos sobre microcontroladores utilizados na ação de extensão relatada.

2.1. Ensino de Robótica

Na literatura existem vários trabalhos relacionados ao ensino de robótica [Nesi et al. 2014, de Jesus and Cristaldo 2014, Neto et al. 2015].

O trabalho de [Nesi et al. 2014] propõe o ensino e a inclusão da robótica na grade curricular do ensino médio brasileiro, como um estímulo para que jovens estudantes ingressem e permaneçam em cursos de ciências exatas, engenharia e tecnologia. A proposta

de [Nesi et al. 2014] se baseia em iniciativas na Inglaterra e Austrália, que incluíram em suas grades curriculares obrigatórias disciplinas voltadas a programação e raciocínio lógico. Nesse trabalho, foi desenvolvido uma proposta de plano de aula, cujo objetivo é o desenvolvimento do raciocínio lógico, bem como das características básicas de *hardware* e *software*. O projeto de [Nesi et al. 2014] foi a base para o planejamento e desenvolvimento do presente trabalho, pois através dele percebe-se a necessidade do ensino de robótica no âmbito nacional, considerando algumas diretrizes internacionais.

Os autores [de Jesus and Cristaldo 2014] mostram a eficiência do ensino de robótica no processo de aprendizagem, através de uma aplicação de uma olimpíada de robótica envolvendo estudantes do Ensino Médio Integrado de Informática. O trabalho avaliou o desempenho escolar dos alunos antes e durante o projeto, para relacionar o ensino de robótica com o desempenho escolar dos estudantes. Os resultados do trabalho mostram que houve um aumento de 8% das notas acima de $9,0$ da turma selecionada. Além disso, foi avaliada a qualidade do projeto, o resultado foi expressivo, mais de 80% dos alunos avaliaram o projeto como excelente. Os resultados de [de Jesus and Cristaldo 2014] foram fundamentais para justificar a implementação do presente projeto, pois apresentam uma grande porcentagem de aceitação do projeto por parte dos alunos do Ensino Médio Integrado.

[Neto et al. 2015] coletaram dados de artigos relacionados ao uso da robótica na educação, publicados no Simpósio Brasileiro de Informática na Educação (SBIE), no Workshop de Informática na Educação (WEI) e no Workshop de Robótica na Educação (WRE) no período de 2004 a 2014. Para essa revisão, foram selecionados 65 artigos. Os resultados desse trabalho mostram que grande parte das pesquisas estava localizada nas regiões Sudeste e Nordeste do Brasil. Em um dos questionamentos do trabalho, notou-se que há uma escassez de experimentações educacionais na área, com isso os autores concluem com a afirmação que o ensino da robótica no país ainda lida com muitas propostas, tornando ele uma ciência, até esse tempo, teórica. A revisão sistemática de [Neto et al. 2015] mostrou dados sobre o estado da arte do uso da robótica na educação brasileira, por meio de alguns desses dados percebeu-se a necessidade da utilização da robótica educacional em caráter experimental.

Há diversas iniciativas de levar a robótica para o ambiente escolar, porém com kits completos de desenvolvimento ou plataformas já implementadas, como apresentam os trabalhos de [Benitti et al. 2009] e [Santos et al. 2010].

O trabalho de [Benitti et al. 2009] apresenta um experimento realizado com alunos do primeiro ano do ensino médio. Nesse experimento, os alunos programam robôs para realizarem rotas sobre o mapa de Santa Catarina, aplicando conceitos relacionados à matemática, geografia e programação de computadores.

[Santos et al. 2010] abordam o desenvolvimento de um kit de robótica de baixo custo, levando em consideração o alto custo de kits proprietários já existentes, e utilização do kit nos cursos técnicos e tecnológicos. Os kits desenvolvidos nesse trabalho custaram em torno de 30 reais, mostrando assim a viabilidade do kit para escolas públicas, por exemplo.

Diferente da ação apresentada no trabalho de [Benitti et al. 2009] e [Santos et al. 2010], o curso de microcontroladores possui um custo menor, pois os

microcontroladores são circuitos integrados compactos, reduzindo assim o custo e o tamanho do produto desenvolvido. Com relação à iniciativa apresentada no artigo de [Santos et al. 2010], o curso de microcontroladores se diferencia no tipo de produto desenvolvido, pois o trabalho de [Santos et al. 2010] mostra o desenvolvimento de kits de robótica educacional, enquanto o curso de microcontroladores propõe a inserção dos alunos de ensino médio por meio de experimentos com microcontroladores.

2.2. Conceitos de Microcontroladores

De acordo com [Martins 2005], os microcontroladores estão presentes na grande maioria dos dispositivos eletrônicos atuais, tornando tais dispositivos compactos, o que facilita a manutenção e gerenciamento das funcionalidades internas desses aparelhos.

Os microcontroladores são circuitos integrados, encapsulados em pastilhas, que possuem processador, pinos de entrada e saída e memória. As variações de microcontroladores disponíveis no mercado, de acordo com [Martins 2005], diferem na quantidade de memória interna, velocidade do processamento de dados, quantidade de pinos de entrada e saída, forma de alimentação, consumo de energia, quantidade de periféricos, arquitetura e conjunto de instruções.

Um sistema com diversas funcionalidades necessitaria de um grande número de componentes, porém [Martins 2005] relata em seu trabalho que o mesmo sistema pode ser simplificado ao ser construído com um microcontrolador e conclui que aprender a programar para um microcontrolador é "aprender a resumir circuitos em um único componente".

Vários conceitos de eletrônica estão envolvidos na experimentação em robótica e, dependendo da aplicação, vários circuitos devem ser construídos. Nesse contexto, os microcontroladores são essenciais na síntese desses circuitos. A abordagem da programação para microcontroladores remete o estudante a essa condensação de circuitos em uma única pastilha microcontrolada.

No trabalho de [Ordoñez et al. 2005], é apresentado o PIC16F628[1] da Microchip[2]. O autor cita o chip como bem difundido e presente em projetos de automação nos mercados brasileiro e mundial. Além disso, ele relata que a alta performasse do chip o tornara um sucesso de vendas para sistemas embarcados.

Diferentemente do trabalho de [Ordoñez et al. 2005], o presente projeto utiliza o microcontrolador PIC16F870, também da Microchip, que por sua vez é um circuito integrado (CI) com um maior número de módulos que o PIC16F628, porém com a mesma quantidade de instruções (35). A utilização desse dispositivo fundamentou-se por seu baixo custo e sua aplicabilidade em diversos projetos do mercado.

3. Procedimentos Metodológicos

Devido o caráter prático e as aplicações pedagógicas deste projeto, a metodologia utilizada neste trabalho foi baseada na Investigação-Ação (IA). De acordo com [Coutinho et al. 2009], a principal característica da metodologia IA é o fato de ser uma

[1]http://ww1.microchip.com/downloads/en/DeviceDoc/30569C.pdf
[2]www.microchip.com

pesquisa de essência prática e aplicada, e que se justifica pela necessidade de resolver problemas reais.

Primeiramente foi realizado um levantamento de demanda com a coordenação do curso de Informática da Escola Estadual de Educação Profissional Maria Cavalcante Costa. O levantamento foi realizado por meio de uma reunião, em que se encontravam o diretor da instituição e o coordenador do curso.

Após o levantamento da demanda com os responsáveis pelo curso técnico na escola, viu-se a necessidade de uma ação que incentivasse o interesse dos alunos pela programação e deu-se inicio a elaboração de um curso que aplicasse a programação no desenvolvimento de sistemas microcontrolados com enfoque na robótica.

Em seguida, foi selecionada a turma de segundo ano de informática da escola para a implementação do projeto. O critério de escolha foi os conhecimentos básicos de programação que a turma possuía, mais especificamente em lógica de programação.

Assim como no trabalho de [Nesi et al. 2014], foi desenvolvido um plano de aulas. O esboço foi elaborado baseado na disciplina de Microcontroladores ofertada no curso de Engenharia de Computação da Universidade Federal do Ceará - Campus de Quixadá. As aulas foram planejadas de modo que os alunos imergissem na experimentação da robótica [Neto et al. 2015]. A partir disso, foram estabelecidos os principais pontos a serem abordados no curso: introdução aos paradigmas de programação na linguagem C, configuração e programação de periféricos IO, configuração e programação para o display LCD, introdução a comunicação serial com o módulo USART, introdução ao tratamento de sinais analógicos com o módulo ADC, introdução ao controle de servomotores com módulo PWM e a construção do primeiro robô. Os temas foram definidos por serem conceitos de microcontroladores utilizados na robótica.

No decorrer do curso, foram programadas atividades práticas em grupos onde os alunos tiveram contato com programação e eletrônica na resolução de problemas reais, seguindo a metodologia IA. A organização dos tópicos do curso foi idealizada de modo que ao final do projeto os alunos estejam aptos a construir seu primeiro robô, um carro de controle remoto.

Para as aulas, foram utilizados microcontroladores PIC16F870 e o Ambiente de Desenvolvimento Integrado (IDE) MPLAB X[3], ambos da empresa Microchip. O microcontrolador escolhido é de baixo custo, de fácil aquisição e bastante utilizado em aplicações de baixa complexidade e a IDE é disponibilizada gratuitamente pela empresa fabricante do chip.

Foram desenvolvidos apresentações de slides e listas de exercícios para que os alunos também compreendessem o conteúdo teórico, todos os materiais foram disponibilizados em um grupo da rede social WhatsApp, em que participaram 25 dos 30 alunos participantes.

4. Execução do Curso de Robótica

A primeira turma do curso foi iniciada em maio de 2017 com a participação de 30 alunos e foi planejada para ter 32 horas de duração distribuídas em 2 horas semanais. Após

[3]http://www.microchip.com/mplab/mplab-x-ide

a conclusão da primeira turma, houveram algumas modificações nos tópicos do curso conforme a necessidade dos alunos. A segunda turma foi iniciada em março de 2018 e contou com a participação de 40 alunos.

4.1. Alterações dos Tópicos Entre a Primeira e Segunda Turma

A principal dificuldade observada na aplicação do curso para a turma de 2017 foi nos paradigmas de programação em C, mesmo após a disciplina de lógica de programação em C. Essa problemática levou os organizadores da ação a adicionarem uma aula de introdução aos paradigmas de programação em C, para que esse obstaculo fosse sanado.

A partir da turma de 2018, foi utilizado o *software* Open Broadcaster Software[4] (OBS) durante as aulas. O OBS é um *software* livre para gravação de vídeos, para que as aulas pudessem ser disponibilizadas para os alunos. As aulas gravadas foram publicadas no canal do YouTube do grupo PET Tecnologia da Informação Conexões de Saberes, na Playlist do curso[5].

4.2. Tópicos Abordados no Curso

Nos itens seguintes, são apresentados os resumos de todas as aulas ministradas na turma de 2017.

- **Apresentação do curso, conceitos gerais envolvendo microcontroladores**: Nessa aula, foi apresentada a justificativa do curso, o plano de ensino e o grupo idealizador do projeto, também foi apresentado o conceito de microcontroladores e alguns projetos reais utilizando o chip. Após a apresentação dos microcontroladores, os alunos se apresentaram e comentaram um pouco sobre suas experiencias e habilidades.
- **Introdução aos paradigmas de programação em C**: Alguns paradigmas de programação em C foram abordados no curso, como operações com bits e a manipulação dos bits de configuração. Na aula de paradigmas de programação, foi aplicado uma técnica de aprendizado denominada programação em par que é também utilizada na metodologia *Extreme Programming* [Teles 2004]. Essa técnica consiste na programação conjunta com dois alunos, sendo um deles o piloto (o que produz o código de fato) e o copiloto (o aluno que auxilia o piloto). Utilizando-se essa técnica, percebeu-se a maior participação da turma na aula, além do protagonismo dos alunos na construção do conhecimento. Para a experimentação em programação, foi utilizada a IDE Dev-C++ [6], como ilustrado na Figura 1.
- **Configuração e Programação de Periféricos I/O**: Após a exposição da visão geral do PIC16F870 e dos paradigmas de programação em C, foram explorados os pinos digitais de entrada e saída do chip. Foram desenvolvidos projetos de semáforos, com *display* de 7 seguimentos, botões e teclado matricial.
- **Configuração e programação para o *display* LCD**: O último dispositivo de saída de dados apresentado foi o *display* LCD. Na primeira aula, foi apresentado o documento do *display* LCD 16x2 e os alunos desenvolveram as funções de configuração do mesmo. Após a construção das funções do *display*, foi montado o

[4]https://obsproject.com/pt-br
[5]https://bit.ly/2rBUY1s
[6]http://orwelldevcpp.blogspot.com.br/

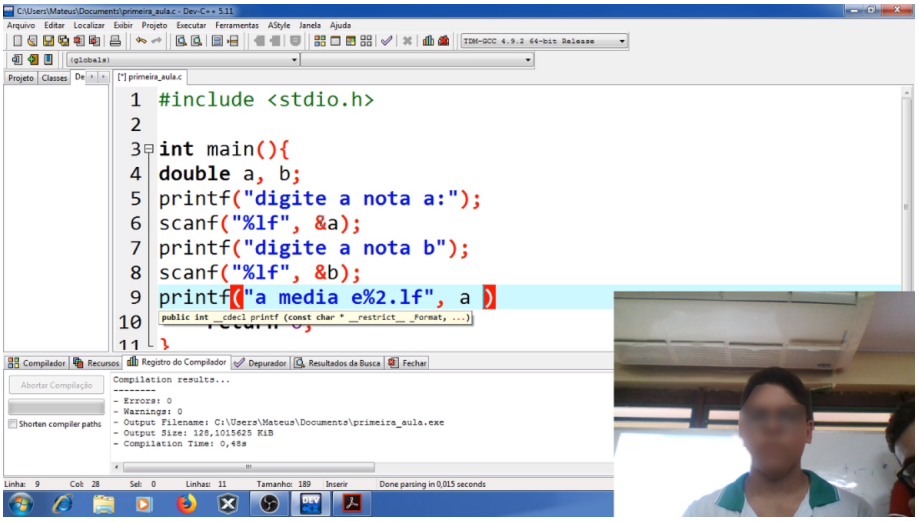


Figura 1. Aplicação da metodologia de programação em par.

circuito do *display* através de um mapeamento de pinos com o diagrama de pinos do PIC18F4550. Na segunda aula desse tópico, foi realizada uma atividade de construção de uma calculadora, unindo a biblioteca do teclado matricial com a do *display*, conforme ilustrada na Figura 2.

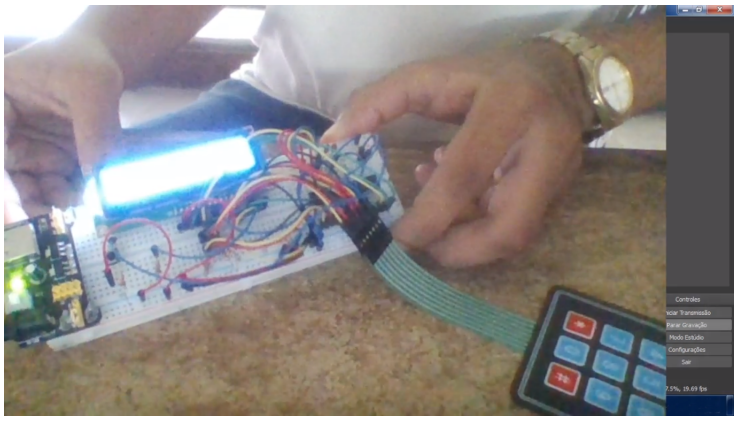


Figura 2. Projetando uma calculadora.

- **Introdução a comunicação serial com o módulo USART**: Nessa aula, foi explorado o protocolo de comunicação USART, seus modos de operação e suas configurações de operação. Para exemplificar a utilização do protocolo USART, foi utilizado o módulo de comunicação *bluetooth*, que mais à frente foi utilizado como interface entre um carro de controle remoto e o controle remoto.
- **Introdução a leitura de sinais analógicos com o conversor ADC**: Neste tópico, foram abordadas as características do módulo ADC e suas configurações básicas. Os alunos utilizaram o sensor de luminosidade LDR para coletar o nível de luminosidade ambiente e enviar esse nível para o PIC em forma de onda analógica. Foi desenvolvido uma aplicação envolvendo a leitura do sensor de luminosidade e o acionamento de uma lâmpada através de um módulo relé.
- **Introdução ao controle de servomotores com PWM**: Nessa aula, foi abordado

o módulo *Pulse Width Modulation* (PWM), bem como suas características e configuração no PIC. Foi utilizado o servomotor 3g, que é controlado por pulsos periódicos gerados pelo módulo PWM. Esse serviço foi utilizado para controlar o ângulo de direção das rodas de um carro de controle remoto.

- **Construção do primeiro robô - O carro de controle remoto**: Na parte final do curso, os alunos desenvolveram um carro de controle remoto. Esse projeto final teve como objetivo levar os alunos a uma compreensão maior dos conceitos, e a aplicação do conhecimento adquirido no curso, por meio do desenvolvimento de uma aplicação prática unindo todos conceitos.

5. Resultados

Visando identificar se a realização da atividade instigou o interesse dos alunos pela robótica e para obter um *feedback* acerca do curso, foi solicitado aos alunos que respondessem a um questionário de avaliação. No momento da aplicação do questionário, estavam presentes em sala vinte e nove (29) dos trinta (30) participantes.

No questionário, foram feitas perguntas relacionadas ao conhecimento prévio dos alunos sobre programação, e todos eles já haviam tido contato no curso técnico, como esperado. Poucos alunos (17,3%) afirmaram ter um desempenho alto nas disciplinas de programação que fizeram, a maioria (51,7%) afirmou ter desempenho mediano e o restante (31%) afirmou ter baixo desempenho. Maior parte dos alunos (65,5%) manifestou ter interesse por robótica.

No mesmo questionário, foram realizadas perguntas com o intuito de obter um retorno dos alunos sobre o curso. Quase todos os alunos (93,1%) consideraram interessante a proposta do curso. Mais da metade dos que responderam ao questionário (58,6%) afirmaram que o conteúdo foi aplicado com velocidade ideal e as outras respostas relacionadas à velocidade da aplicação do conteúdo foram distribuídas entre rápida (27,6%), muito rápida (10,3%) e lenta (3,4%).

As últimas perguntas do questionário foram relacionadas à experiência dos alunos ao entrarem em contato com a robótica. Todos os alunos (100%) passaram a considerar a robótica importante para a sociedade. A maioria dos alunos (69%) se sentiu estimulado a aprender robótica, enquanto (31%) não tem interesse.

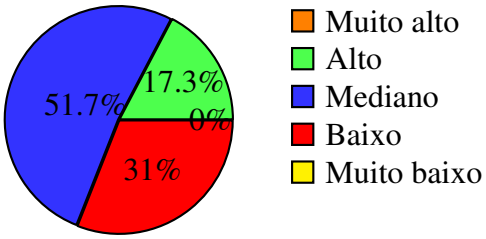
A resposta dos alunos em relação ao estimulo ao aprendizado de robótica mostrou que a abordagem com microcontroladores não os desestimulou, pelo contrário, esse estilo de ensino de robótica fez os alunos conhecerem de fato como funciona a lógica que rege os dispositivos.

Os resultados obtidos foram satisfatórios, pois os alunos se interessaram pela proposta do curso e se sentiram estimulados a aprenderem mais sobre a área. Essas respostas também mostraram que os alunos acreditam que a robótica é importante para a sociedade, o que foi refletido na abordagem prática da ação.

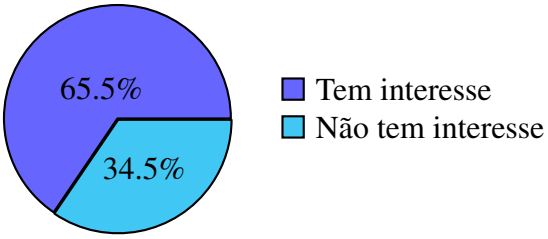
Todos esses dados podem ser visualizados nos gráficos da Figura 3.

6. Conclusão e Trabalhos Futuros

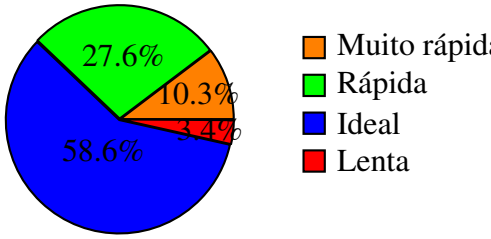
Este trabalho apresentou um relato de experiência do PET TI - UFC Campus Quixadá no ensino de robótica em uma escola profissional de ensino médio utilizando microcontro-



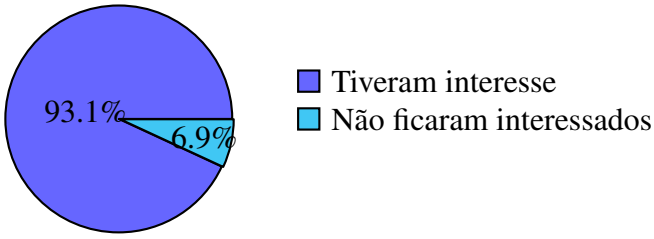
(a) Desempenho dos alunos nas disciplinas de programação vistas no curso técnico.



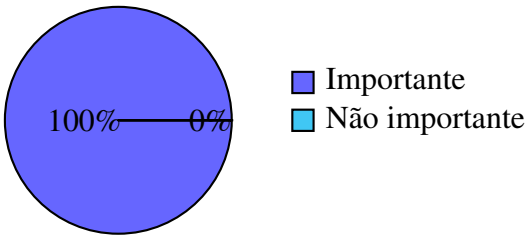
(b) Interesse dos alunos por robótica



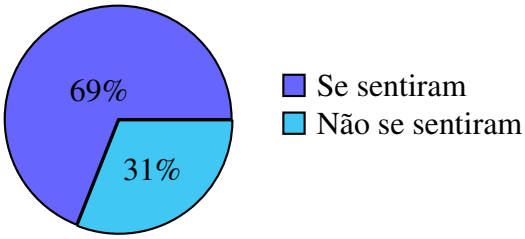
(c) Avaliação dos alunos relacionada a velocidade de aplicação do conteúdo.



(d) Interesse dos alunos pela proposta do curso.



(e) Resposta dos alunos sobre a importância da robótica para a sociedade.



(f) Resposta dos alunos sobre se sentirem estimulados a aprender robótica, ou não.

Figura 3. Gráficos de repostas

ladores. A ação visou o incentivo aos jovens alunos na área de exatas, não só ensinando programação, mas à construção de um carrinho de controle remoto com hardware de baixo custo. A partir das experiências vividas e dos dados coletados, foi possível inferir que os alunos conseguiram compreender os temas abordados no curso e que passaram a ter um interesse maior por aprender sobre robótica.

A experiência de ensino contribuiu também para o aprendizado dos facilitadores, pois estes vivenciaram a experiência ensino aprendizado do ponto de vista docente. O retorno dos alunos em relação a configurações de periféricos foi notável, pois eles compreenderam o funcionamento do dispositivo e construíram as bibliotecas, mesmo com uma restrita carga horária por tópico. A construção de slides compactos para a introdução dos conteúdos tornou a explicação teórica breve e objetiva, adquirindo com isso mais atenção dos alunos.

Como trabalhos futuros, pretende-se realizar à aplicação de novas metodologias de ensino, como a Aprendizagem Baseada em Problemas (PBL) [Ribeiro 2008] para comparações entre os resultados obtidos, de modo a obter uma melhoria do curso no decorrer

das próximas ações, além disso também é pretendido realizar um estudo para verificar se os conceitos aprendidos no projeto influenciaram do desempenho dos alunos nas demais disciplinas do curso técnico em Informática.

Referências

[Avila et al. 2016] Avila, L., Bernardini, F. C., and Moratori, P. (2016). O uso de robótica para aprendizado de programação integrando alunos de educação básica e ensino superior. In *XXIV Workshop de Educação em Computação-WEI. Porto Alegre, RS, Brasil*.

[Barrientos and Barrientos 2007] Barrientos, A. and Barrientos, A. (2007). Fundamentos de robótica. Technical report, e-libro, Corp.

[Benitti et al. 2009] Benitti, F. B. V., Vahldick, A., Urban, D. L., Krueger, M. L., and Halma, A. (2009). Experimentação com robótica educativa no ensino médio: ambiente, atividades e resultados. In *Anais do Workshop de Informática na Escola*, volume 1, pages 1811–1820.

[Coutinho et al. 2009] Coutinho, C. P., Sousa, A., Dias, A., Bessa, F., Ferreira, M. J. R. C., and Vieira, S. R. (2009). Investigação-acção: metodologia preferencial nas práticas educativas. *Revista Psicologia, Educação e Cultura*.

[de Jesus and Cristaldo 2014] de Jesus, L. and Cristaldo, M. F. (2014). Uma abordagem utilizando lego mindstorms education ev3 para verificar o desempenho acadêmico dos estudantes do instituto federal de educação, ciência e tecnologia de mato grosso do sul do câmpus aquidauana. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 25, page 1198.

[Martins 2005] Martins, N. A. (2005). Sistemas microcontrolados. *Uma abordagem com o Microcontrolador PIC 16F84. Editora Novatec Ltda, 1ª edição*.

[Miyagi and Villani 2008] Miyagi, P. E. and Villani, E. (2008). Mechatronics as a solution for system automation. *Journal of Exact Sciences*, 5(2).

[Nesi et al. 2014] Nesi, I. C., Nesi Junior, V., et al. (2014). Robótica educacional: uma proposta curricular para o ensino médio.

[Neto et al. 2015] Neto, R. P. B., Santana, A. M., Rocha, D. P., and Souza, A. (2015). Robôtica na educação: Uma revisão sistemática dos últimos 10 anos. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 26, page 386.

[Ordoñez et al. 2005] Ordoñez, E. D. M., Penteado, C. G., and da Silva, A. C. R. (2005). *Microcontroladores e FPGAs: aplicações em automação*. Novatec Editora.

[Ribeiro 2008] Ribeiro, L. R. d. C. (2008). Aprendizagem baseada em problemas (pbl): uma experiência no ensino superior.

[Santos et al. 2010] Santos, F. L., Nascimento, F. M. S., and Bezerra, R. M. (2010). Reduc: A robótica educacional como abordagem de baixo custo para o ensino de computação em cursos técnicos e tecnológicos. In *Anais do Workshop de Informática na Escola*, volume 1, pages 1304–1313.

[Teles 2004] Teles, V. M. (2004). Extreme programming. *São Paulo: Novatec*.

Ensino de bioética em cursos superiores de computação: uma análise crítica

Rodrigo Cândido Borges[1], Maria Márcia Bachion[2]

[1]Departamento de Áreas Acadêmicas – Instituto Federal de Goiás (IFG)
Inhumas – GO – Brasil

[2]Faculdade de Enfermagem – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brasil

rodrigocand@gmail.com, mbachion@gmail.com

Abstract. *Since its inception, bioethics has gone through different phases and one of the best known is clinical bioethics. However, its original scope is much broader and is being resumed, which allows it to approach more diversified areas. In the present article, we try to highlight themes and dimensions of Computer Science that lack bioethical reflections and the reader is invited to the dialogue about the possibility of insertion of Bioethics in the curriculum of the superior courses of computation.*

Resumo: *Desde o seu surgimento, a bioética passou por diferentes fases e uma das mais conhecidas é a bioética clínica. Contudo, seu escopo original é bem mais amplo e está sendo retomado, o que permite sua aproximação com áreas bastante diversificadas. No presente artigo, busca-se evidenciar temas e dimensões da Ciência da Computação que carecem de reflexões bioéticas e convida-se o leitor para o diálogo sobre a possibilidade de inserção da Bioética no currículo dos cursos superiores de computação.*

1. Introdução

Os avanços científicos e tecnológicos advindos da evolução computacional possibilitaram significativas transformações nas interações sociais, na comunicação, no acesso e na difusão da informação. O impacto dessas inovações requer discussões acerca de suas implicações, seus danos e benefícios, bem como diálogo sobre suas consequências éticas e bioéticas.

A compreensão que a ciência resulta em mais tecnologia, ganho capital e bem-estar social tem sido questionada, pois as linhas tecnológicas não são equânimes e possuem implicações e usos que também precisam ser refletidos (Roso e Auler, 2016).

Com um escopo pluralista e reflexivo (Potter, 1970), a bioética integra saberes científicos e morais, o conhecimento e os valores humanos, transcendendo mais do que o mero plano das normas e dos códigos de ética.

A preocupação com formação em Bioética por meio de sua incorporação nos currículos em nível de graduação ocorreu inicialmente na área de saúde (Neves Júnior, Araújo e Rego, 2016; Maluf e Azambuja, 2015; Couto Filho *et al*., 2013). Gradativamente, áreas como as Ciências Biológicas passaram a perceber a necessidade de incluir o tema na formação dos futuros profissionais (Dória e Moreira, 2011).

No entanto, isso pouco tem sido explorado em outras graduações que, embora não sejam da área da saúde, atuam direta ou indiretamente em contextos nos quais são inerentes questões e dilemas envolvendo o bem-estar, a felicidade e a segurança de seres humanos.

A computação é geradora de tecnologias variadas e revolucionárias, capazes de promover a qualidade de vida da humanidade, mas, também, com possibilidades de comprometer o ambiente, as relações sociais e desvalorizar determinadas culturas. Promover reflexões bioéticas, no processo de construção do conhecimento daqueles que estarão envolvidos no desenvolvimento científico, torna-se relevante em um contexto onde suas ações sempre são repletas de consequências para a vida humana.

O presente trabalho tece uma análise crítica que contempla especialmente a inclusão da bioética no currículo de formação superior dos profissionais ligados à informática, abordando realidades, iniciativas e contestações.

2. A Bioética de Potter

Van Rensselaer Potter, em 1970, divulgou seu primeiro texto utilizando a palavra Bioética (do inglês *Bioethics)*. O artigo intitulado *Bioethics, the science of survival* destacava a necessidade de uma ética abrangente, aliando os saberes científico e moral (Potter, 1970). Em 1971, publicou o livro *Bioethics: A Bridge to the Future*, no qual definiu bioética como ponte entre ciências e humanidades (Potter, 1971). Os questionamentos levantados pelo autor apoiavam-se na preocupação com um futuro dito duvidoso, sobretudo devido à falta de diálogo entre os saberes científicos e os princípios éticos.

Em 1988, foi publicada sua segunda obra, *Global Bioethics: Building on the Leopold Legacy* (Potter, 1988), em que ampliou a bioética a outras disciplinas, não se restringindo à ponte entre a biologia e a ética, mas com a dimensão de uma Ética Global. Nesta concepção, estão envolvidos todos os aspectos relativos ao viver, o conhecimento biológico, a ciência dos sistemas vivos, os valores humanos e a ética médica.

Posteriormente, em 1998, propôs a definição da chamada Bioética Profunda (Potter, 1998), uma nova ciência ética apoiada na humildade, responsabilidade, competência intercultural e interdisciplinar, potencializando o senso de humanidade. Este conceito evidencia seu esforço em manter características pluralistas para incorporação de novos conhecimentos, centrado no pensamento de que a possibilidade de assumir erros promove aberturas para aprender com novas experiências.

Na sua origem, com Potter (1970), a Bioética não se propunha a ser uma teoria que pudesse nortear o processo de análise, julgamentos e tomadas de decisão. Contudo, com o fortalecimento da Bioética Clínica, após as concepções de Beauchamp e Childress (2002) desenvolvidas mediante debates suscitados por pesquisas moralmente questionáveis na área médica, e o enfrentamento de dilemas cada vez mais frequentes na área da saúde, desenvolveram-se modelos de tomada de decisão baseados em princípios éticos (Loch, 2003). Para Otero (2009), esse movimento representou uma usurpação, uma dissidência do sentido do neologismo originalmente proposto por Potter (1970, 1971), opinião corroborada por Pessini (2013). Segundo Cunha e Lorenzo (2014), a partir de 2005 ampliaram-se os horizontes da Bioética na comunidade científica, após a

publicação da Declaração Universal sobre Bioética e Direitos Humanos pela Organização das Nações Unidas para a Educação, a Ciência e a Cultura (Unesco).

Pessini (2013) destaca trechos da fala de Potter (1998) nos quais se percebe seu chamamento para que se compreenda a "bioética como uma nova ética científica que combina humildade, responsabilidade e competência, numa perspectiva interdisciplinar e intercultural que potencializa o sentido de humanidade" e a transforme em uma disciplina (entendida como um campo de conhecimento científico).

Assim, percebe-se que a Bioética é um campo dinâmico, ao mesmo tempo controverso, mas que não se pode ignorar. Cada grupo de estudiosos pode adotar um sentido específico (Cunha e Lorenzo, 2014), mas com um ponto em comum: a reflexão filosófica sobre o modo de vida e o modo de relação dos seres humanos entre si, entre eles e os outros seres vivos, entre eles e o ambiente.

Segundo Caramico, Zaher e Rosito (2007), na visão de Potter, a bioética representa uma amplitude que contempla o conhecimento científico e filosófico. Para eles, a bioética foi criada para nortear avanços tecnológicos em todas as áreas, em prol da humanidade, sendo de interesse da maioria da sociedade.

O pensar Bioética deve se estender a aspectos do cotidiano dos profissionais e, de modo geral, dos cidadãos, o que demanda processos de formação. Na perspectiva da educação, é necessária a abertura para novos conteúdos, a fim de que os alunos pensem moralmente por si próprios e decidam com autonomia (Neves Júnior, Araújo e Rego, 2016).

3. O Ensino de Bioética em Ciências da Saúde

Questões éticas estão historicamente presentes na Medicina desde seus primórdios, o que pode ser constatado no juramento solene de Hipócrates, escrito há mais de 2 mil anos (Carneiro *et al.,* 2010). Contudo, segundo Caramico, Zaher e Rosito (2007), a ética médica alcançou um novo patamar após o surgimento da Bioética e de sua inclusão no ensino de graduação nessa profissão, assim como abriu horizontes, protocolou temas para reflexão, expandiu e modificou tratamentos médicos, trouxe o imperativo de mais respeito para com o outro e inter-relacionou especialidades médicas e não médicas. Após a análise de 76 projetos pedagógicos de cursos de Medicina do Brasil, a pesquisa realizada por Neves Júnior, Araújo e Rego (2016) identificou que 25 faculdades apresentavam disciplinas com conteúdos de bioética associados a outras matérias; enquanto 31 possuíam em sua matriz curricular disciplinas autônomas com conteúdo e nome "Bioética".

Apesar de importante, a inserção de formação em bioética na Medicina não deve se limitar à dimensão curricular. O trabalho de Camargo, Almeida e Morita (2014) revelou que a maioria dos alunos do sexto ano da graduação considera este um tema importante, mas não acompanha as discussões que o envolvem nos espaços extracurriculares.

A Bioética como componente do currículo também se faz presente em faculdades de Odontologia brasileiras. Maluf e Azambuja (2015) enfatizam que o dentista realiza atividades clínicas, cirúrgicas, administra recursos humanos e materiais, mas suas relações com os pacientes ainda são permeadas por conflitos e dilemas que exigem atenção e preparo moral, ético e bioético para contorná-los ou preveni-los.

Valladão *et al*. (2011) ressaltam que, na ciência odontológica contemporânea, os problemas éticos podem surgir tanto na graduação como na pós-graduação e na clínica odontológica, assim como se entrelaçar e se sobrepor, o que evidencia a importância da difusão da bioética para a comunidade e universidade.

De acordo com Mascarenhas e Rosa (2010), o ensino da Bioética na graduação em Enfermagem deve se adequar às demandas para a formação de um profissional generalista, humanista, crítico, reflexivo e essencialmente ético, voltado para atender às necessidades de saúde, nos âmbitos individual e coletivo. Couto Filho *et al*. (2013) investigaram a oferta da disciplina ou da temática bioética nos cursos de graduação em Enfermagem das universidades federais brasileiras. Das 16 instituições que atendiam ao critério de inclusão do estudo, em oito delas (50%) a Bioética era apresentada como temática específica ou constava na grade curricular do curso. Os autores consideram que não inserir este tema na formação impõe prejuízos aos futuros profissionais.

4. A Inserção da Bioética na Formação de Recursos Humanos

No âmbito da educação superior brasileira, os cursos pertencentes às Ciências da Saúde se destacam no pioneirismo da oferta desta disciplina. Com o passar dos anos, outras áreas do conhecimento têm incluído o ensino da Bioética na composição programática de seus cursos.

No campo das Ciências Biológicas, pesquisa realizada por Dória e Moreira (2011) explorou a inserção curricular da Bioética (ou similares) nos cursos de Biologia de 36 Instituições Federais de Ensino Superior no Brasil. Segundo o estudo, 50% das universidades analisadas tinham instituído o conteúdo e trabalhavam, sobretudo, temas relacionados à ética biomédica com uma notável abordagem globalizante, incluindo questões socioambientais e ecológicas. Siebert (2015) assinala alguns dilemas bioéticos que incidem sobre a formação do biólogo: questões voltadas ao consumismo, poluição, recursos energéticos, crescimento demográfico, alimentos transgênicos e manipulações genéticas. Para este autor, o ensino de bioética é indispensável nos cursos de formação de profissionais ligados à área de ciências biológicas, sobretudo devido à desarmonia existente entre os avanços tecnológicos e as reflexões morais sobre suas consequências.

Ao analisar a formação inicial de professores de Ciências e Biologia, Silva e Krasilchik (2013) puderam verificar lacunas na formação inicial dos docentes para lidarem com fatos que demandem mediação de discussões ou tomada de posição. Os autores frisam a importância da bioética no debate de alguns temas principais, tais como limites das atividades científicas, preservação ecológica, soberania geopolítica, consentimento e autonomia, que deveriam ser obrigatoriamente abordados durante a formação inicial de professores de Ciências e Biologia, a fim de prepará-los para essas discussões com os alunos.

A inserção de bioética é crescente também nas Ciências Agrárias. A Medicina Veterinária tem se destacado por tornar a disciplina obrigatória em algumas instituições, tais como a Universidade Federal de Goiás, Universidade Federal de Uberlândia, Universidade do Estado da Bahia, entre outras. Questões importantes como o bem-estar animal e a biossegurança estão inseridas em diversos ementários, sendo também objeto de debates em congressos específicos (CFMV, 2018). Para Zanetti (2009), conhecer os elementos que compõem as experiências dos alunos nas aulas práticas em que animais são usados como recurso didático é um desafio educacional, pois essas vivências afetam

a construção da formação profissional e podem ser consideradas expressão de violência ou crueldade contra os animais e também contra os próprios alunos.

No campo das Ciências Sociais Aplicadas, as Faculdades de Direito têm se sobressaído na inserção curricular de Bioética. Novas abordagens metodológicas vislumbram familiarizar os estudantes com uma nova compreensão da ciência jurídica, correlacionando-a com as ciências da vida e conferindo sentido ao chamado Biodireito (Pereira, 2015). Villas-Bôas (2012) salienta que o cabedal epistemológico desenvolvido pela bioética nos últimos anos traz um ganho para o Direito e para a sociedade. A aproximação dos profissionais do âmbito jurídico e de saúde tende a enriquecer esses diálogos e contribuir para o crescimento e empoderamento do ser humano.

Em relação às Engenharias, Adinolfi (2014) conclui que a Bioética e a educação em valores configuram-se como uma área ainda silenciosa, uma lacuna caracterizada pela ausência de materiais, disciplinas e conteúdos nas grades curriculares dos cursos de graduação. Segundo a autora, o desconhecimento e o desinteresse pela bioética podem decorrer do aspecto tecnicista da formação de engenheiros no Brasil, cujo foco é determinar a exequibilidade de um projeto, dando pouca ênfase à sua viabilidade e ao quanto ele é, ou não, desejável. Dentre 88 cursos de graduação em Engenharia do estado de São Paulo analisados, apenas um incluía a Bioética; outros 12 tinham conteúdos de formação ética em geral (Adinolfi, 2014).

No tocante às Ciências Exatas e da Terra, não foram encontrados indícios de inserção da Bioética na matriz curricular em cursos superiores.

A baixa inclusão ou ausência da Bioética como disciplina em áreas fora do âmbito da saúde suscita uma série de indagações sobre o perfil do egresso de grande parte dos cursos de graduação brasileiros. O modelo educacional tradicionalmente adotado volta-se, em sua maioria, à formação técnica dos discentes, pouco contemplando a formação de valores e o olhar sobre outros campos do conhecimento. A educação como doutrinação dificulta a oferta de abordagens multidisciplinares como a de Bioética, que remetem à reflexão de assuntos relacionados aos impactos científicos e tecnológicos, além dos dilemas que afetam os rumos e a sobrevivência do planeta.

5. Bioética e Computação

A computação é a ciência que se dedica ao tratamento da informação mediante o uso de computadores ou dispositivos que permitem o processamento de dados. Com a inclusão digital, os computadores, inicialmente restritos às grandes organizações, passaram a integrar o contexto doméstico. A preocupação em fazer com que as máquinas e seus conteúdos fossem acessíveis a todos foi um fator primordial para a popularização das tecnologias e motivou investimentos para linhas como as arquiteturas de computadores, engenharia de software, redes de computadores, computação móvel e ubíqua, entre outras (Carvalho, 2006).

Os ganhos promovidos pelas tecnologias informáticas contribuem para o desenvolvimento científico não só das Ciências Exatas, mas também de outras áreas dos saberes, consistindo em um ramo de pesquisa altamente aplicável que objetiva atender a diferentes anseios. No contexto da saúde, métodos avançados de inteligência artificial, como o aprendizado profundo, têm sido aplicados no desenvolvimento de programas que simulam as atividades dos neurônios da região do cérebro onde ocorrem os

pensamentos. Outras frentes caminham para a ideia de supercomputadores contribuírem no diagnóstico de enfermidades, utilizando a alta capacidade de armazenamento de dados aliados a algoritmos inteligentes (Siuly, Huang e Daneshmand, 2018). No campo da Bioética, redes neurais artificiais têm sido modeladas para que as deliberações clínicas sejam acertadas do ponto de vista tanto técnico quanto ético (Siqueira-Batista *et al*., 2014).

Quanto às soluções voltadas aos usuários de dispositivos móveis, o aplicativo "Meu digiSUS" (MS, 2018), criado pelo Departamento de Informática do SUS, é um exemplo do que se pode oferecer em relação a informações em saúde de uso pessoal e restrito aos cidadãos, como o acesso a dados do cartão nacional de saúde, listas de exames realizados e de medicamentos retirados e acompanhamento do cartão de vacinação.

Apesar das notórias cooperações da computação com outras áreas na contemporaneidade, algumas de suas derivações científicas e tecnológicas são objetos de questionamentos. O mercado de software pirata é um caso típico, alimentado por um consumismo desenfreado de parte da população, mantenedora de um cenário ilícito. A empresa Business Software Alliance (BSA, 2018) relata que cerca da metade dos programas instalados em computadores no Brasil são ilegais. Ao longo de 2016, só na América Latina, foram gastos 5,8 bilhões de dólares com programas piratas.

O uso de programas piratas traz prejuízos diretos para as empresas proprietárias das tecnologias originais. A lucratividade retirada das organizações de forma ilegal compromete os planos de gestão e negócios, lesando, por exemplo, a possibilidade de contratação de funcionários, ou mesmo interferindo nos gastos destinados a possíveis melhorias dos sistemas. Trata-se de um dano que envolve condutas antiéticas tanto por parte de usuários como de profissionais de informática, disseminadores deste tipo de produto.

A necessidade de preservar a integridade, confidencialidade, disponibilidade e autenticidade dos dados digitais faz com que a segurança da informação seja uma disciplina de destaque na computação moderna. O grande dilema reside em definir quem tem direito a acessá-los e alterá-los. Uma informação, ao ser considerada confidencial, traz uma série de implicações, por extrapolar a esfera técnica e atingir a esfera ética (Mann, 2011).

Em 2017, uma série de ataques cibernéticos ocorreu em computadores de hospitais e empresas de comunicação, em pelo menos 70 países. Na Inglaterra, médicos de hospitais públicos tentavam acessar os prontuários de pacientes e se defrontavam com avisos de bloqueio das informações e liberação somente após o pagamento de um resgate. Tratava-se de um sequestro de dados digitais, uma modalidade de crime denominado *ransomware* (ANSA, 2018).

Outro fato polêmico consiste na autonomia e privacidade em ambientes corporativos. Programas têm sido desenvolvidos e implantados para monitorar imagens de tela, movimentos do mouse e sinais do teclado dos funcionários em empresas, o que tem incrementado debates sobre o limite de um software. Defensores da causa argumentam que o computador da empresa não é ferramenta dos funcionários. Assim, ver notícias, fazer compras ou realizar download de jogos e aplicativos são ações que deveriam ser executadas apenas em casa, sob risco de demissão. Outra frente defende

que tal prática denota invasão de privacidade, por ferir princípios da liberdade e legalidade (Sitesa, 2018).

O impacto ambiental da computação também é motivo de controvérsias. Os equipamentos eletrônicos são considerados sucatas, geralmente em curto prazo, despejando anualmente milhões de toneladas deste tipo de lixo, composto por uma série de materiais, alguns altamente tóxicos, como o chumbo, o mercúrio e o cádmio. Além disso, o consumo energético de equipamentos eletrônicos, na atual era da internet e das telecomunicações, é da ordem de bilhões de quilowatts-hora (Farinaccio, 2016). Essa série de questões demanda cada vez mais dos profissionais da área de computação, desde a formação, uma postura ética e de colaboração em relação à sociedade, à vida e ao meio ambiente.

Esses profissionais encontram-se cada vez mais envolvidos em decisões de projetos, avaliando o grau de confiabilidade e tolerância a falhas de sistemas (Raymond, 2015). A quase totalidade dos programas é gerenciada por usuários que têm acesso a informações confidenciais ou críticas. Portanto, a formação discente deve envolver habilidades que transcendam a capacidade prática e alcancem a reflexão das consequências. A instituição de ensino superior que não prepara seus alunos para atender a este novo quadro não o capacita para um mercado tão inovador.

O mundo para qual o ensino superior forma os profissionais tem mudado e impõe novas demandas às instituições. O cenário econômico globalizado alterou as fronteiras do conhecimento, da cultura e do mercado (Adinolfi, 2014). As tecnologias da informação e comunicação geraram uma nova linguagem, icônica e global. A era do conhecimento individual, baseado na cultura impressa, foi substituída por um tempo em que o conhecimento é uma construção coletiva, apreendido pela cultura digital (Adinolfi, 2014).

Segundo Camargo, Almeida e Morita (2014), os alunos representam um campo fértil, pois anseiam por atendimento humanitário e questionam realidades maiores, aspecto que permite inferir interesse e capacidade de absorver a abordagem da ética e da bioética na graduação.

Essa inserção poderia ocorrer sob forma de uma disciplina ou tema transversal, ambas com seus prós e contras. Como disciplina, pode constituir uma fortaleza quando desenvolvida por docentes com perfil adequado, estudiosos do campo específico e capazes de relacionar de forma dinâmica a Bioética e a Ciência da Informação, mas há risco do ensino tornar-se um processo delimitado no tempo e espaço, sem aplicação em outros momentos da formação. Para superar este aspecto, sua inserção no Projeto Pedagógico do Curso deve privilegiar seu entendimento como tema transversal. Se incluída em outras disciplinas, como Informática e Sociedade, por exemplo, pode perder a sua centralidade, a sua essência, em meio a outros temas discutidos. Por outro lado, existir ou fortalecer-se neste espaço é possível mediante a participação de especialistas e uso de abordagens que favoreçam a aprendizagem significativa. Não há um caminho exclusivo, pois as alternativas podem se somar, alternar, até que se encontre uma proposta que permita a criação de um espaço para que o diálogo entre a Bioética e a Ciência da Computação, na graduação, propicie a formação diferenciada destes profissionais.

Para Fischer *et al.* (2017), proporcionar a reflexão bioética na sociedade, por meio do encontro entre profissionais de diferentes áreas, professores universitários, graduandos e estudantes do ensino básico, promove o crescimento moral de todos. A argumentação, a interação com a realidade, e o conhecer os valores dos outros possibilitam o alcance de soluções consensuais e justas, respondendo às necessidades do indivíduo, da humanidade, da sociedade, da natureza, desta e das próximas gerações (Fischer *et al.*, 2017).

A bioética se relaciona à ideia de que é urgente, em todas as profissões, priorizar a formação integral da pessoa, preparando-a para se relacionar com os outros e com o mundo, convivendo e respeitando as diferenças, singularidades e opiniões (Wilges, 2007).

6. Considerações Finais

Cientistas da computação lidam direta ou indiretamente com atividades e desenvolvimento de tecnologias que afetam a vida e o modo de ser de pessoas, grupos culturais e da sociedade como um todo. Portanto, agem dotados de um componente moral, mesmo que inconsciente ou despercebido. A aproximação com a Bioética pode proporcionar aos futuros profissionais bases teóricas para uma reflexão qualificada e subsídios para a ação prudente, baseada no respeito e na compreensão das consequências atuais e futuras de suas decisões.

Referências

Adinolfi, V. T. S. (2014) "Educação em valores e bioética – formação de engenheiros". Tese (Doutorado) – Universidade de São Paulo, Faculdade de Educação, São Paulo.

ANSA, Brasil (2017) "Mega ataque cibernético atinge mais de 70 países". Disponível em: http://ansabrasil.com.br. Acesso em 10 mai. 2018.

Beauchamp, T.; Childress, J. F. (2002) "Princípios de ética biomédica". São Paulo: Loyola.

BSA (2018) "Business Software Alliance". Disponível em: http://www.bsa.org. Acesso em: 10 mai. 2018.

Caramico, H. J; Zaher, V. L.; Rosito, M. M. B. (2007) "Ensino da bioética nas faculdades de medicina do Brasil". Bioœthikos, v. 1, n. 1, p. 76-90.

Camargo, A.; Almeida, M. A. S.; Morita, I. (2014). "Ética e bioética: o que os alunos do sexto ano médico têm a dizer". Revista Brasileira de Educação Médica, v. 38, n. 2, p. 182-189.

Carneiro, Larissa A.; Porto, Celmo C.; Duarte, Soraya B. R.; Chaveiro, Neuma; Barbosa, Maria A. (2010). "O ensino da ética nos cursos de graduação da área de saúde". Revista Brasileira de Educação Médica, v. 34, n. 3, p. 412-421.

Carvalho, Marcelo Sávio Revoredo Menezes de. (2006) "A trajetória da Internet no Brasil: do surgimento das redes de computadores à instituição dos mecanismo de governança. Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, COPPE, Engenharia de Sistemas e Computação, Rio de Janeiro.

CFMV (2018) “IV Congresso Brasileiro de Bioética e Bem-Estar Animal”. Disponível em: http://bioeticaebea.cfmv.gov.br. Acesso em: 10 mai. 2018.

Cunha, T.; Lorenzo, C. (2014) “Bioética global na perspectiva da bioética crítica”. Rev. bioét. (Impr.) v. 22, n. 1, p. 116-125.

Couto Filho, J. C. F; Souza, F. S.; Da Silva, S. S.; Yarid, S.; Sena, E. L. S. (2013) “Ensino da bioética nos cursos de Enfermagem das universidades federais brasileiras”. Revista Bioética, v. 21, p. 179-185.

Dória, T. A. F.; Moreira, L. M. A. (2011) “A bioética na formação do biólogo: um desafio contemporâneo”. Revista Entreideias: educação, cultura e sociedade, n. 20, p. 99-122.

Farinaccio, Rafael (2016) “Afinal, quanta energia elétrica a internet utiliza para funcionar?”. Disponível em: https://www.tecmundo.com.br. Acesso em: 10 mai. 2018.

Fischer, M. L.; Cunha, T. R.; Roth, M. E.; Martins, G. Z. (2017) “Caminho do Diálogo: uma experiência bioética no ensino fundamental”. Revista Bioética, v. 25, n. 1, p. 89-100.

Loch, J. A. (2003) “Como analisar conflitos em bioética clínica”. In: Urban, C. A. Bioética clínica. Rio de Janeiro: Revinter, p. 48-54.

Mann, Ian (2011) “Engenharia Social - Série Prevenção de Fraudes”. São Paulo: Blucher, 236 p.

Maluf, Fabiano; Azambuja, Letícia E. O. (2015) “Bioética e Odontologia: considerações sobre a relação profissional-paciente”. Revista Odontológica de Araçatuba, v. 36, n. 2, p. 61-65.

Mascarenhas, N. B.; Rosa, D. O. S. (2010) “Bioética e Formação do Enfermeiro: uma interface necessária”. Texto & Contexto – Enfermagem, v. 19, n. 2, p. 366-371.

MS (2018) “Ministério da Saúde – DATASUS – Meu digiSUS”. Disponível em: https://play.google.com. Acesso em: 10 mai. 2018.

Neves Júnior, W. A.; Araújo, L. Z. S.; Rego, S. (2016) “Ensino de bioética nas faculdades de medicina no Brasil”. Rev. Bioética v. 24, p. 98-107.

Otero, L. D. (2009) “Bioética: El concepto relegado”. Interciencia v. 34, n. 1, p. 71-76.

Pereira, Bernardo Augusto da Costa (2015) “O Biodireito brasileiro, seus princípios e a Bioética”. Revista Contribuciones a las Ciencias Sociales, n. 29.

Pessini, Leo. (2013) “As origens da bioética: do credo bioético de Potter ao imperativo bioético de Fritz Jahr”. Revista Bioética, v. 21, n. 1, p. 09-19.

Potter, V. R. (1970) “Bioethics: the science of survival”. Persp Biol Med., v. 14, p. 127-153.

Potter, V. R. (1971) “Bioethics: bridge to the future”. Prentice Hall Inc. Englewood Cliffs New Jersey.

Potter, V. R. (1988) “Global bioethics: building on the Leopold legacy”. East Lansing: Michigan State University Press.

Potter, V. R. (1998) "Script do vídeo elaborado e apresentado para o IV Congresso Mundial de Bioética". O Mundo da Saúde, v. 22, n. 6, p. 370-374.

Raymond, Gerson (2015) "Profissional de TI, Profissional Multitarefa ou Profissional Super-Homem?" Disponível em: https://www.tiespecialistas.com.br/2015/03/. Acesso em: 10 mai. 2018.

Roso, C. C.; Auler, D. (2016). "A participação na construção do currículo: práticas educativas vinculadas ao movimento CTS". Ciência & Educação (Bauru), v. 22, n. 2, p. 371-389.

Siebert, P. R. (2015) "Bioética para estudantes de Ciências Biológicas: investigação sobre os fundamentos que compõem ou deveriam compor a área". Tese (Doutorado) - Universidade Estadual Paulista, Faculdade de Ciências, Bauru.

Silva, P. F.; Krasilchik, M. (2013) "Bioética e ensino de ciências: o tratamento de temas controversos - dificuldades apresentadas por futuros professores de ciências e de biologia". Ciência & Educação (Bauru), v. 19, n. 2, p. 379-392.

Siqueira-Batista, R; Gomes, A. P.; Maia, P. M.; Costa, I. T.; Paiva, A. O.; Cerqueira, F. R. (2014) "Modelos de tomada de decisão em bioética clínica: apontamentos para a abordagem computacional". Revista Bioética, v. 22, n. 3, p. 456-461.

Siuly, S.; Huang, R.; Daneshmand, M. (2018) "Guest editorial: special issue on Artificial Intelligence in Health and Medicine". Health Information Science and Systems, v. 6.

Sitesa (2018) "Monitoramento de Empregados – Possibilidade". Disponível em: http://www.sitesa.com.br/contabil/conteudo_trabalhista/procedimentos/p_trabalhista/m07.html. Acesso em: 10 mai. 2018.

Valladão, A. S. N.; Graciosa, L. K.; Silva, M. F.; Pecoraro, P. V. B. F. (2011) "A bioética odontológica contemporânea - ampliando concepções deontológicas". Revista Interdisciplinar de Direito, Faculdade de Direito de Valença.

Villas-Bôas, M. E. (2012) "Bioética e direito: aspectos da interface". Desafios e perspectivas de um chamado biodireito. Revista Bioethikos, v. 6, n. 2, p. 89-100.

Zanetti, M. B. F. (2009) "O uso experimental de animais como instrumento didático nas práticas de ensino no curso de medicina veterinária". IX Congresso Nacional de Educação - EDUCERE. III Encontro Sul Brasileiro de Psicopedagogia, PUCPR. Curitiba.

Wilges, L. B. M. (2007) "A bioética num enfoque educacional: implicações na formação de professores de ciências e biologia". Dissertação (Mestrado) – PUCRS, Pós-Graduação em Educação em Ciências e Matemática.

O desenvolvimento da identidade docente por professores de Computação não licenciados atuantes na Educação Profissional de Nível Médio

Ranansamir Sousa da Silva[1]; Ecivaldo de Souza Matos[1]; Monica de Sousa Massa[2]

[1]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

[2]Departamento de C. Exatas e da Terra – Universidade do Estado da Bahia (UNEB)
Salvador - BA - Brasil

{ranansamir.silva; ecivaldo}@ufba.br, monicamassa@gmail.com

Abstract. *Teacher identity is a process, whose social and professional dimensions are interrelated, being strongly affected by the school context. This paper presents results of an investigation about the development process of the teaching identity by Computing professionals with no pedagogical training in undergraduate course, but that act as teachers of Professional Education at a public school in the state of Bahia. The data were collected by interviews and participant observation of Computing classes during a school year. The data analysis revealed some aspects about the difficult to establish teaching identity by those subjects, and that difficulty may influence the success of their didactic action.*

Resumo. *A identidade docente é um processo de construção, cujas dimensões social e profissional estão inter-relacionadas, sendo fortemente afetado pelo contexto escolar. Este artigo apresenta resultados de uma investigação qualitativa sobre o processo de desenvolvimento da identidade docente por profissionais de Computação sem formação pedagógica em nível de Licenciatura, atuantes como professores da Educação Profissional de Nível Médio em uma escola pública do estado da Bahia. Os dados foram coletados por meio de entrevistas e observação-participante das aulas ministradas ao longo de um ano letivo, cuja análise de conteúdo revelou alguns aspectos que dificultam o estabelecimento da identidade profissional docente por esses sujeitos e que podem influenciar o sucesso da sua ação didática.*

1. Introdução

A partir de 2004, por meio do Decreto nº 5.154/04 (BRASIL, 2004), o Governo Federal iniciou uma série de ações que reformularam a Educação Profissional de Nível Médio no Brasil - EPT. Dentre os diversos cursos técnicos oferecidos no âmbito da EPT, alguns fazem parte da área de Computação, como: Redes de Computadores; Computação Gráfica; Desenvolvimento e Análise de Sistemas; Suporte e Manutenção de Computadores, entre outros. A oferta de vagas na EPT é destinada a jovens egressos do ensino fundamental que desejam ter, de imediato, contato com uma determinada

profissão. Trata-se de uma proposta de formação cidadã que prepara o jovem para o mundo do trabalho (RAMOS, 2008).

Essa formação citada por Ramos (2008) indica a necessidade de profissionais docentes habilitados para atuarem nas escolas de nível médio, de modo a atender aos objetivos da EPT. Para os cursos técnicos da área de Computação, a Licenciatura em Computação - LC - é o meio pelo qual os professores podem obter a referida habilitação para atuarem no ensino de disciplinas computacionais.

Durante o curso de LC o professor terá contato com os mais diversos conhecimentos didáticos e pedagógicos que o dotará de recursos/ferramentas próprias a sua atividade de ensino. Além disso, o contato com a profissão de professor, a partir da formação inicial (graduação), facilitará o desenvolvimento do processo de construção da identidade docente (NÓVOA, 2013).

Segundo Matos (2013), o egresso da LC estará capacitado para atuar na EPT de forma satisfatória, pois sua formação compreende, além da capacitação técnica, conhecimentos pedagógicos que irão habilitar o profissional docente para atuar na escola.

Todavia, ainda há poucos profissionais formados em cursos de Licenciatura em Computação (LC) para suprir a demanda por professores da EPT (JUNIOR, 2014). Ao passo que a EPT se expande, a formação de professores na LC não segue no mesmo ritmo (GARIGLIO E BURNIER, 2012).

A carência de professores licenciados, isto é, habilitados ao exercício da docência na educação profissional de nível médio, aliada ao entendimento de que para ser professor basta o conhecimento técnico na área de Computação, tem sido uma das principais justificativas dos gestores públicos para contratação de profissionais de Computação sem formação em curso de licenciatura para atuar em sala de aula da EPT (GARIGLIO; BURNIER, 2012).

Nesse sentido, este artigo apresenta resultados de uma pesquisa sobre o desenvolvimento da identidade de professor por profissionais sem formação pedagógica em curso de licenciatura, mas que atuam como professores da educação profissional de nível médio na rede pública estadual da Bahia.

A seguir, na Seção 2 será apresentada os principais conceitos relacionados ao tema desta pesquisa, como “identidade docente” e “mediação didática”. Na Seção 3 são explicitados os passos metodológicos de coleta e análise dos dados. A Seção 4 apresenta alguns dos resultados acerca do desenvolvimento da identidade docente dos sujeitos investigados. Na Seção 5 são tecidas as considerações finais acerca dos achados da pesquisa e, por conseguinte as referências utilizadas neste artigo.

2 Fundamentação teórica

2.1 Identidade docente

Em contraponto ao entendimento de alguns gestores, para Tardif (2012) o professor é “alguém que deve conhecer sua matéria, sua disciplina e seu programa, além de possuir certos conhecimentos relativos às ciências da educação e à pedagogia e desenvolver um saber prático baseado em sua experiência cotidiana com os alunos” (p. 39). Nesse

sentido, o autor apresenta uma lista com a pluralidade de saberes que devem fazer parte da formação profissional docente: (i) saberes da formação profissional, desenvolvidos pelas instituições de formação de professores; (ii) saberes disciplinares, desenvolvidos no âmbito das disciplinas científicas; (iii) saberes curriculares, apresentam-se concretamente sob a forma de programas escolares que os professores devem aprender a aplicar; e (iv) saberes experienciais, construídos ao longo da experiência dos sujeitos e são por eles validados (TARDIF, 2012).

Segundo Pimenta (1997), esses saberes citados por Tardif (2012) estão relacionados à construção da identidade profissional docente. Saberes que são constituídos em três dimensões: (i) da experiência, que seria aquele desenvolvido pelo professor desde quando aluno, com os seus antigos professores que marcaram aquela época, do mesmo modo, o que é produzido na prática em um processo de reflexão e troca com os pares; (ii) do conhecimento, que abrange a revisão da função da escola na mediação dos conhecimentos e as suas especialidades em um contexto contemporâneo; e (iii) dos saberes pedagógicos, aqueles que abrangem a questão do conhecimento juntamente com o saber da experiência e os conteúdos específicos e que será construído a partir das necessidades pedagógicas reais.

De acordo com Cardoso *et al.* (2012), os saberes curriculares (saberes referentes à prática pedagógica) são fundamentais na formação da identidade do professor, que se trata de um processo "complexo e marcado por diferentes períodos, diferentes vivências e experiências" (p. 11). Nesse sentido, a formação docente compreende os saberes desenvolvidos durante o curso destinado à formação de professores - formação inicial, o aprendizado da vida social e familiar, contempla o período enquanto estudante, as relações que se estabelecem no local de trabalho (escola) e através das relações estudantes e seus pares (professores). Além disso, essa formação está inter-relacionada à identidade docente e, portanto, devem ser analisadas em conjunto.

Segundo Nóvoa (2013), o comportamento do sujeito está aliado à sua atuação profissional na medida que sua identidade será conhecida através da forma como "vive" a profissão docente. Desse modo, o autor apresenta o que denomina de triplo A (AAA), para caracterizar o processo de construção identitária dos professores: adesão, ação e autoconsciência.

Essa abordagem apresentada pelo professor Nóvoa é claramente definida nos dizeres de Pimenta (1997): "A identidade não é um dado imutável nem externo que possa ser adquirido. É um processo de construção do sujeito historicamente situado" (p. 6).

2.2 Mediação didática

No entendimento de Freire (2000), a mediação didática está fundamentada no diálogo enquanto condição básica para o conhecimento. O ato de conhecer faz parte do processo social e o diálogo é uma forma de mediação deste processo. Por outro lado, o processo de mediação didática está relacionado à forma como o professor afeta seus alunos ao ponto de lhes provocar o desejo pelos estudos.

Para D'Ávila (2005), a mediação didática é o processo de descobrir o que os estudantes sabem e como sabem acerca de determinado conteúdo/tema, tendo o professor enquanto mediador dos saberes e tradutor. O professor torna-se um "mediador

entre as ideias dos estudantes e os objetos de conhecimento" (D'ÁVILA, 2005, p. 236). O importante, segundo a autora, é considerar o estudante dotado de concepções culturais; além disso, a responsabilidade de mediar, com base no conhecimento prévio do sujeito, é do professor.

3. Metodologia

Pela necessidade de investigação em profundidade do desenvolvimento de um processo que envolve a análise do sujeito em sua relação com o ambiente educacional, optou-se pelo uso de método qualitativo de análise, conforme sugere Gatti (2013).

A decisão pela abordagem qualitativa também está relacionada às características do problema investigado (subjetividade, compreensão como base para obter o conhecimento, coleta de dados, descobertas teóricas, enfoque indutivo para análise dos dados e outros), conforme orientação de Ludke e André (2005). A coleta de dados foi realizada por meio de um estudo de caso ao longo de um ano letivo completo, com a participação de três professores de Computação (sujeitos da pesquisa) atuantes em um curso técnico em informática de um Centro Estadual de Educação Profissional - CEEP.

A amostra foi selecionada por conveniência e aceitação ao convite realizado à escola. Garantiu-se que os três sujeitos possuíam a característica fundamental única para participação na pesquisa: ser professor de Computação na EPT, atuando em sala de aula, mas sem formação pedagógica habilitadora (curso superior de licenciatura na área); de modo que o problema de pesquisa estivesse inscrito no caso investigado, justificando a opção pelo método de estudo de caso.

3.1. Coleta de dados

Dada a natureza da pesquisa qualitativa (avaliação em profundidade), o estudo de caso foi realizado com apenas três sujeitos, cujos dados foram coletados por meio de entrevistas e observação-participante nas aulas dos três professores (sujeitos de pesquisa) ao longo de todo o ano letivo.

A observação-participante, com o objetivo de acompanhar a atividade didática dos professores, resultou na produção de 97 relatórios de observação diária em todas as séries (1º ao 4º ano) em que o grupo de professores leciona disciplinas técnicas de Computação. A maioria das aulas/turmas observadas se concentrou no início e no final do curso, 1º e 4º ano, respectivamente (cf. Quadro 1).

Quadro 1.Frequência de observação por ano/série

Ano/Série	Frequência
1º	36%
2º	10%
3º	25%
4º	29%

O formulário de relatório de investigação era composto por um cabeçalho para identificar a aula observada. Alguns campos do relatório foram: data, local, disciplina, professor (nome civil), turma, quantidade de alunos, duração da aula e nome do

pesquisador. No corpo do relatório era escrita os acontecimentos da aula e as inferências do pesquisador.

Em cada uma das séries (quatro séries ao total), foram observadas diversas disciplinas com o objetivo de conhecer e acompanhar a atuação do professor em contextos diferentes, conforme apresentado no Quadro 2.

Quadro 2. Disciplinas observadas durante a pesquisa

Disciplinas
Arquitetura de Computadores
Banco de Dados
Análise e Projetos de Sistemas
Programação Visual
Lógica de Programação
Informática Básica
Segurança de Redes
Sistema Operacional
Redes de Computadores
Segurança de Sistemas

A hora-aula no contexto investigado foi composta de 50 minutos. Todavia, o tempo contabilizado durante a observação considerou os minutos efetivamente gastos em aula; assim, o tempo em que os estudantes se deslocavam de uma sala para outra e se acomodavam foi desconsiderado. De acordo com os registros da pesquisa, foram observadas 97 aulas, que somaram aproximadamente 74 horas de efetiva observação.

Além da observação, foram realizados dois blocos de entrevistas com o grupo de professores. O primeiro bloco ocorreu no início do estudo de caso e o segundo próximo ao final do período letivo.

O objetivo do primeiro bloco de entrevistas foi coletar dados pessoais, profissionais e da formação acadêmica dos professores (perfil básico). O segundo bloco de entrevistas buscou atualizar os dados coletados na primeira entrevista, além de levantar dados sobre as concepções dos professores acerca da identidade docente, o ambiente de trabalho, a vivência na educação profissional, a mediação didática e outros elementos relevantes à teorização quanto ao desenvolvimento da identidade desses professores.

3.2. Análise de dados

Para analisar os dados coletados nas entrevistas e nos relatórios diários de investigação, utilizou-se a técnica de Análise de Conteúdo (BARDIN, 1979).

A Análise de Conteúdo (AC) é “um conjunto de técnicas de análise das comunicações, visando obter, por procedimentos sistemáticos e objetivos de descrição do conteúdo das mensagens, indicadores que permitam a inferência de conhecimentos relativos às condições de produção/recepção destas mensagens.” (BARDIN, 1979, p. 42), organizada em três fases: pré-análise, exploração do material e tratamento dos resultados.

A partir da macrocategoria definida na pré-análise voltada ao objetivo da pesquisa ("desenvolvimento da identidade docente estabelecida por profissionais da Computação que atuam na docência"), definiu-se as três categorias utilizadas na análise dos dados: *formação docente; identidade docente e mediação didática.* Essas categorias foram extraídas dos estudos de Nóvoa (2013), Pimenta (1997), D'ÁVila (2008) e Tardif (2012), cujo objeto de investigação se aproxima do objeto desta pesquisa.

Na categoria *Formação Docente* investigou-se a relação da formação inicial e continuada (todo e qualquer curso que tenham participado após a conclusão do curso de graduação) com ocupação laboral no magistério. Informações acerca da formação docente (inicial e continuada - forma ou informação/não-formal) foram encontradas nos dados levantados nos dois blocos de entrevistas.

Fundamentada em Nóvoa (2013), a Formação Docente foi dividida nas seguintes subcategorias: *formação inicial*, *formas de atualização docente* e *formação pedagógica continuada*. Essas subcategorias estão alicerçadas na necessidade de formação inicial e continuada como professor para que seja possível ao sujeito refletir sobre sua identidade docente.

A segunda categoria analisada foi a *Identidade Docente*. Investigou-se como o sujeito se percebe na carreira profissional do magistério e algumas relações que se estabelecem neste processo de reconhecimento da "nova" profissão. Para tanto foram consideradas duas subcategorias: a autopercepção como professor *versus* profissional de mercado; e a autoidentificação e autovalorização. Essas subcategorias são oriundas de Nóvoa (2013), D'ÁVila (2008), Pimenta (1997), Pimenta e Anastasiou (2014) e Tardif (2012), quanto à percepção própria do sujeito de si e sobre si, por seus pares; a experiência laboral na docência como fator preponderante para desenvolvimento da identidade docente e os saberes adquiridos dentro/fora do ambiente escolar.

Quadro 3. Categorias e subcategorias de análise

<table>
<tr><th colspan="3">MACROCATEGORIA
Desenvolvimento da identidade docente estabelecida por profissionais da Computação que atuam na docência</th></tr>
<tr><th>FORMAÇÃO DOCENTE</th><th>IDENTIDADE DOCENTE</th><th>MEDIAÇÃO DIDÁTICA</th></tr>
<tr><td>Formação inicial</td><td>Autoidentificação e autovalorização</td><td>Modelos de mediação didática estabelecidos</td></tr>
<tr><td>Formas de atualização docente</td><td rowspan="2">Autopercepção como professor versus profissional de mercado</td><td rowspan="2">Principais estratégias utilizadas pelo professor em uma aula de Computação</td></tr>
<tr><td>Formação pedagógica continuada</td></tr>
</table>

A terceira categoria analisada foi *Mediação Didática*. Investigou-se a atuação em sala a partir das percepções do professor e das inferências relatadas, nos relatórios de investigação, pelo pesquisador. As subcategorias analisadas foram os modelos de mediação didática e as principais estratégias utilizadas pelo professor em aulas de Computação. A análise das subcategorias foi realizada com base nos estudos de D'ÁVila (2008) e Pimenta (1997) em seus estudos sobre os modelos de mediação didática: modelo artesanal, modelo academicista ou conteudista, modelo tecnicista ou

instrumental e modelo reflexivo. O Quadro 3 apresenta as categorias e suas respectivas subcategorias utilizadas na pesquisa.

Com base nessas categorias, foram desenvolvidos os códigos que serviram de indicadores de cada uma das subcategorias.

Para organizar o material, especificar as subcategorias e seus códigos, visando dar suporte à análise, utilizou-se a versão livre (*free*) do aplicativo QDA Miner[1] (QDA).

Durante a fase de pré-análise, percebeu-se a necessidade de tratamento dos dados coletados, preparando-os para uso adequado do QDA e, consequentemente, provendo mais clareza na relação entre as informações significativas e os código criados. Para isso, foi criado um documento denominado *extrato de investigação*.

O *extrato de investigação* é um resumo da transcrição das entrevistas, contendo apenas os dados significativos acerca do perfil do professor e suas concepções, com base nas entrevistas e nos relatórios diários de investigação.

A partir do extrato de investigação, com o QDA devidamente configurado com as subcategorias e seus respectivos códigos, quantificou-se as ocorrências dos códigos nos respectivos extratos de investigação de cada sujeito participante da pesquisa. Quantificou-se também as ocorrências dos códigos (indicadores) definidos para cada uma das subcategorias. Os indicadores poderiam validar ou não as suspeitas previamente encontradas na etapa de pré-análise ou ainda indicar novas suspeitas. Essas quantificações não serão apresentadas neste texto, mas podem ser consultadas em Silva (2017).

Utilizando o extrato de investigação foi possível apurar alguns resultados que, combinados, indicam os desafios enfrentados pelos professores de Computação atuantes na EPT no estado da Bahia e as implicações diretas e indiretas no desenvolvimento da identidade de professor por esse profissional.

Considerando que a identidade docente tem seu processo estabelecido nos anos em que o sujeito começa a estudar; que a evolução desse processo perpassa pela formação inicial docente e continuada (todo e qualquer tipo de capacitação ou estudo sobre temas educacionais) durante a carreira do professor; e, além disso, é refletida pela sua atuação didática, buscou-se aprofundar esta pesquisa a fim de investigar para refletir sobre como este processo identitário acontece para/com os profissionais da Computação que atuam como professores da EPT (NÓVOA, 2013; BARRETO, 2009).

Desse modo, durante a análise dos dados, identificou-se respostas às questões relacionadas à formação inicial e continuada. Investigou-se também quais são as concepções acerca da profissão docente desse professor e como elas são refletidas na sua prática cotidiana, de modo a contribuir para a definição da sua identidade de professor de Computação (NÓVOA, 2013; BARRETO, 2009).

4. A construção da identidade docente: alguns resultados

Esta seção apresenta alguns resultados acerca da análise dos dados coletados durante o estudo de caso, cujo objetivo foi responder à seguinte questão norteadora: "c*omo é*

[1]http://provalisresearch.com/products/qualitative-data-analysis-software/freeware/i.

estabelecido o processo de desenvolvimento da identidade docente de profissionais da Computação que não possuem formação inicial na Licenciatura em Computação e que atuam na EPT?"

4.1 A formação inicial dos sujeitos

Considerando que todos os sujeitos possuem formação inicial na área de Computação e que ingressaram na docência aproveitando-se de uma oportunidade para maximizar a renda mensal, torna-se importante saber como a formação em Computação auxilia na atuação docente. Quanto à formação inicial dos participantes, é possível observar no Quadro 4.

Quadro 4. Perfil da amostra

Professor	Graduação	Especialização	Tempo de experiência laboral	
			Computação	Educação
F	Bach. Análise de Sistemas	Metodologia do Ensino Superior	Entre 1 e 2 anos.	Entre 5 e 10 anos.
J	Bach. Análise de Sistemas	Formação Docente para Bacharéis	Entre 5 e 10 anos.	Entre 2 e 5 anos.
I	Bach. Análise de Sistemas	Segurança de Redes	Mais de 10 anos.	Entre 5 e 10 anos.

Considerando que a formação inicial dos sujeitos não possui características que, inicialmente, os habilite a lecionar (curso de Licenciatura), investigou-se como a formação que possuem contribui na prática docente, assim, os participantes foram questionados com a seguinte pergunta: "C*omo sua formação em Computação auxilia na sua atuação enquanto professor?"*.

As respostas foram divididas em duas dimensões para melhor compreensão quanto ao conteúdo e quanto à prática. Quanto ao conteúdo, os sujeitos responderam:

> *"Sim. Porque ensino disciplinas específicas." Prof. F*
>
> *"De forma positiva." Prof. J*
>
> *"Na verdade a minha formação é um ponto positivo por conta das práticas." Prof. I*

Os depoimentos dos professores encontram ressonância nos estudos de Veiga (2012) quanto à supervalorização do conhecimento técnico sobre o pedagógico.

Observa-se que o sujeito identificado como Prof. F é o que possui menos tempo atuando na sua área de formação, por outro lado, é o que está há mais tempo na docência e participou de um curso de Especialização em Metodologia do Ensino Superior. O Prof. J, egresso de um curso de formação docente, acompanha o entendimento dos seus pares, ratificando o entendimento do grupo que, no que diz respeito ao conteúdo, o conhecimento técnico é o foco principal da sua atuação. Outro fato que maximiza essa evidência quanto à supervalorização citada consta dos relatórios de investigação que apontam a preocupação dos sujeitos em cumprir com o programa de conteúdos das disciplinas.

A segunda parte analisada foi a dimensão quanto à prática, a qual os participantes responderam:

> *" ...Eu trago muito para a sala de aula as minhas vivências. [...] trago tudo que já passei atuando no suporte. Eu acho que isso é um ponto positivo." Prof. I*

> *"Hoje eu não penso em ficar apenas na parte acadêmica, para pôr em prática o que vi na minha formação e passar o que é visto no mercado de trabalho para o estudante." Prof. J*

Os sujeitos participantes da pesquisa atuam diariamente com Computação fora da escola: uma parte do dia atuam no ambiente empresarial e noutra no ambiente escolar. O Prof. J concluiu um curso de 40 horas na área de Educação; o sujeito Prof. I não participou de curso semelhante, no entanto, leciona em outra instituição de ensino profissionalizante.

Os depoimentos dos professores sugerem que ambos possuem a concepção de que a sua prática em sala de aula está diretamente relacionada à sua atuação na indústria. Esses sujeitos defendem o sucesso de uma prática baseada em "contação de casos" que ocorrem no seu cotidiano laboral fora da escola.

O foco nos relatos de experiências e nas aulas práticas, com simulação de problemas reais, maximizam as suspeitas de que a formação técnica aliada à experiência laboral em Computação dão suporte às suas estratégias didáticas. Essas concepções ratificam os entendimentos de alguns autores como Veiga (2012) e Pereira (2009), quanto à valorização dada a dimensão técnica em detrimento da reflexão sobre a prática docente, também discutidas por Pimenta & Ghedin (2002) e D'ÁVila (2013).

Essa aparente evidência apontada nos depoimentos, onde o processo de ensino é baseado no conhecimento técnico sobre o conteúdo a ser lecionado e a experiência profissional na área de Computação, fortalece o estabelecimento de um processo de formação baseado no empirismo, além de reforçar o domínio do técnico e do científico sobre o escolar (PIMENTA & ANASTASIOU, 2014; D'ÁVILA, 2008).

Segundo Tardif (2012), essa formação empírica é importante no processo de formação docente. O que se discute, no entanto, é o cuidado em não mecanizar o "fazer docente" ao distanciar-se de processos pedagógicos que atuam como facilitadores do processo de ensino.

4.2 O desenvolvimento de uma *identidade do docente*

Os professores quando arguidos quanto a sua profissão atualmente, assim foram perguntados: "*Qual é a sua profissão?*". Tinha-se o objetivo de entender como o sujeito se identifica profissionalmente. Pimenta (1997) já indicava que a sua autoconcepção profissional implica na sua atuação enquanto profissional de mercado que agrega à docência as suas atividades laborais.

As respostas foram as seguintes:

> *"Eu sou Professora."(Prof. F)*

> *"Eu sou Professora."(Prof. I)*

> *"Eu sou Analista de Suporte"(Prof. J)*

O sujeito identificado como Prof. J está há menos tempo atuando como professor (cerca de dois anos) e, historicamente, sempre atuou no campo empresarial.

Quanto aos demais sujeitos, observa-se a autoidentificação latente em suas respostas. De acordo com os relatórios de investigação, ambos atuam há mais de cinco anos na docência e também nas suas respectivas áreas de formação inicial; todavia, refletem a profissão docente em seu cotidiano, a ponto de se identificarem enquanto professores de Computação.

Essa autoidentificação se fortalece quando os sujeitos elencam algumas características que acreditam ser fundamentais a um professor da área de Computação, como: preparar aulas, realizar aulas práticas em sala, manter boa relação com os alunos, ter comprometimento, lealdade, preocupação com o conteúdo, profissionalismo, conhecimento técnico de Computação, e conhecimento Pedagógico.

Diante das características elencadas pelos sujeitos, foi importante saber como eles se percebem em relação a elas. Assim, foram solicitados a listar até três pontos fortes que acreditavam ter e, em seguida, até três pontos que precisavam melhorar.

Quanto aos pontos fortes, listaram organização, paciência, conhecimento técnico, comprometimento, lealdade, profissionalismo, preocupação como conteúdo, espontaneidade e manter boa relação com alunos.

Por outro lado, compreendem que precisam melhorar o trato emocional com os estudantes, a prática docente, a parte afetiva (*dialogar mais com os estudantes*), enaltecer suas qualidades positivas, minimizando as ações negativas, melhorar a qualidade do trabalho pedagógico e buscar novas formas de conhecimento.

Segundo Nóvoa (2013), essa autoanálise dos professores reflete o grau de pertencimento à docência que o grupo já alcançou e o modo como se percebem profissionalmente, demonstrando a capacidade de agir como um profissional da docência.

Ao tempo que caracterizam o professor de Computação, refletem sobre os mesmos aspectos e buscam se perceber ou não fazendo parte desse conjunto. Com base nos relatórios de investigação, muitas dessas características, tanto pontos fortes como pontos fracos, foram registradas em diversos momentos sob situações adversas. Em alguns momentos, de acordo com os registros, a falta do apoio pedagógico (presencial) evidenciou os pontos fracos de alguns professores: estudantes desinteressados com a aula, supervalorização do conhecimento técnico, centralização do diálogo no professor, inibindo os estudantes de interagirem.

Importante observar que, mesmo tendo a concepção sobre algumas práticas que precisam melhorar, os professores não conseguem avançar por si só, alegam a necessidade de suporte pedagógico, quando afirmam que:

> *"Faz falta um acompanhamento pedagógico." Prof. I*

> *"...Acho que falta um pouco de técnica." Prof. J*

Reconhecem que somente o conhecimento técnico acerca da Computação não é suficiente para sua atuação em sala de aula (TARDIF, 2012; PIMENTA & GHEDIN,

2002; D'ÁVILA, 2013), que sua prática não pode estar baseada somente no "empirismo" (D'ÁVILA, 2008; PIMENTA e ANASTASIOU, 2014).

4.2.1 A importância do ambiente escolar e do projeto político pedagógico

Considerando que os sujeitos de pesquisa, segundo eles, aproveitaram uma "oportunidade de maximizar a renda mensal" como professor e que já atuam por algum tempo na profissão docente, acredita-se que seja importante saber o nível de conhecimento que o grupo possui sobre planos daquele espaço de ensino. Assim, foram questionados: *"Você tem ou teve acesso ao projeto político pedagógico do colégio?"*

Todos os participantes responderam que nunca tiveram qualquer contato com nenhum documento que tratasse do projeto político pedagógico ou do planejamento da escola. De acordo com os registros, até aquele momento, por diversas razões, a escola não possuía um Projeto Político Pedagógico (PPP), mas que estava em processo de construção.

Toda escola precisa de um PPP que oficialize a organização do seu trabalho pedagógico e da sua unidade escolar. Segundo Veiga (2002, p. 1), "a escola é o lugar de concepção, realização e avaliação de seu projeto educativo, uma vez que necessita organizar seu trabalho pedagógico com base em seus alunos".

O PPP é o documento orientador de toda a comunidade escolar, de modo a alcançar os objetivos ali definidos. Em geral, trata-se de um processo democrático onde todos podem contribuir; é político por se comprometer com os interesses reais e coletivos daquela comunidade. Talvez seja possível fazer uma analogia do PPP com uma "bússola", comumente utilizada por barcos para se orientarem.

A falta de um planejamento político-pedagógico, que indique as características identitárias, traz desafios também para a construção da identidade profissional docente dos sujeitos que ali atuam.

4.3 Processos de mediação didática

Nesta etapa de análise, tratando especificamente da categoria *Mediação Didática*, investigou-se a atuação em sala a partir das percepções do professor e das inferências relatadas, nos relatórios de investigação, pelo pesquisador.

As subcategorias analisadas foram os modelos de *mediação didática* estabelecidos e principais estratégias utilizadas pelo professor em uma aula de Computação. A análise das subcategorias foi realizada com base nos estudos de D'ÁVila (2008) e Pimenta (1997) em seus estudos sobre os modelos de mediação didática: modelo artesanal, modelo academicista ou conteudista, modelo tecnicista ou instrumental e modelo reflexivo.

As características encontradas na prática cotidiana dos professores, com base nas inferências registradas nos relatórios de investigação, indicaram quais as características dos modelos de mediação didática estão mais presentes na prática docente, de modo a refletir a identidade de professor.

Percebeu-se os grandes desafios que precisam ser superados cotidianamente pelos professores participantes do grupo; como falta de formação inicial que os capacite

para a docência; carência de suporte pedagógico presencial; ausência de documento formal construído pela comunidade que indique a identidade da escola; e a missão dada ao professor: ser o responsável por contribuir na formação de jovens para o mundo do trabalho por meio da Computação.

Com o objetivo de entender como esses profissionais atuam e como suas práticas refletem sua identidade profissional de professor da área de Computação, analisaram-se-se os depoimentos, amparados pelos dados constantes nos extratos de investigação. A pergunta chave a ser analisada foi: *"A sua forma de ensinar, no seu entender, tem alguma relação com a forma como você aprendeu?"*

Dois dos sujeitos tanto demonstraram como afirmaram que lecionam com base nos exemplos de seus antigos professores. Assim, conservavam as práticas que consideram boas e evitam aquelas que consideram ruins.

Essa forma de atuar não é uma surpresa, considerando a formação inicial do sujeito; todavia, é preciso refletir que as práticas de tempos passados já não se enquadram perfeitamente nos tempos atuais, do mesmo modo o perfil dos estudantes sofreu bastante diferenças (D'ÁVILA, 2008). Percebe-se que, embora alguns desses professores atuem há algum tempo em sala de aula e tenham participado de cursos que tangenciam a Educação, impressão sobre a prática docente está mais inclinada para uma atuação tradicional e bastante voltada ao conteúdo que lecionam, inclusive, em seus depoimentos, todos trazem como principal preocupação, o conteúdo.

A falta de suporte pedagógico fica muito evidenciada neste ponto. Em muitos momentos os professores aparentavam a necessidade de respostas para determinadas questões ocorridas em sala de aula: dificuldade de entendimento da turma, baixo índice de aproveitamento nas avaliações, planejamento deficiente das disciplinas, o foco da atuação do professor em preparar o estudante para a indústria ao invés de proporcionar uma formação crítica, reflexiva e humanística visando preparar o aluno para o mundo do trabalho de forma emancipatória, ao mesmo tempo, desenvolvendo suas habilidades de docente de Computação de forma a favorecer a construção da identidade de professor (LIMA, 2001; NÓVOA, 2013; PIMENTA, 1997).

4.4. Síntese da análise

De acordo com os dados encontrados no âmbito da categoria *Formação Docente*, os três professores possuem formação inicial em cursos superior de Bacharelado em Análise de Sistemas, além de certificados de cursos de pós-graduação *lato sensu*, sendo um em Segurança de Redes, outro em Metodologia do Ensino Superior e o terceiro em Formação Docente.

Além da formação docente, analisaram-se dados referentes à categoria *Mediação Didática*. Considerando os dados encontrados nessa categoria, é possível observar que a ausência de uma formação inicial que os habilite para lecionar, culmina por dotar o sujeito de uma certa autonomia, que suspeita-se seja bastante preocupante na medida que suas práticas são baseadas no empirismo, ou seja, uma prática de tentativa de erros e acertos, carente de um suporte pedagógico adequado.

De acordo com os dados encontrados no âmbito da categoria *Identidade Docente*, constata-se que os sujeitos não ingressaram na carreira do magistério como

acontece comumente com aqueles que desejam ser professor: estudar em cursos de licenciatura. Assim, um dos passos iniciais para o desenvolvimento da identidade de professor (NÓVOA, 2013), não foi cumprido, pois, como já informado antes, a formação inicial dos sujeitos não contemplou aspectos didáticos-pedagógicos. Neste sentido, se mostrou importante investigar como aconteceu o ingresso na carreira do magistério. Ao serem perguntados sobre "*Como você se tornou-professor?*", responderam:

> *"Por acaso. ... surgiu a oportunidade de vir trabalhar..." (Prof. F)*
>
> *"A partir de uma oportunidade que surgiu, por acaso...." (Prof. J)*
>
> *"Comecei na docência meio que por acaso, eu estava trabalhando em uma empresa como analista de suporte e surgiu a oportunidade." (Prof. I)*

Os depoimentos dos sujeitos parecem revelar que inicialmente não havia o desejo de lecionar. Os sujeitos ingressaram na docência aproveitando-se da oportunidade de melhorar sua renda mensal por meio de um segundo vínculo laboral com o magistério. Essa suspeita surgiu no fato de um dos sujeitos trabalhar apenas como professor, mas sua vida profissional ter se iniciado como analista de sistemas, atuando na indústria de software; outro sujeito está há mais de cinco anos trabalhando como professor, trabalha em duas unidades de ensino distintas e atua como profissional liberal nas horas vagas; e o terceiro sujeito, atua como professor há pouco mais de dois anos, mas também exerce a atividade de Analista de Suporte.

Suas práticas cotidianas corroboram com as reflexões de Nóvoa (2013) quanto aos símbolos observados no sujeito, características claras de um profissional da docência. Ainda que não possuam formação inicial na Educação (curso de licenciatura), atuam profissionalmente como professores, não mais com o simples desejo de maximizar a renda mensal, mas com o desejo de refletir sobre a profissão docente e aprofundar o seu aprendizado, a fim de melhorar sua prática cotidiana, conforme demonstrado nas observação e entrevistas.

O processo de construção da identidade docente, no entanto, não depende apenas da formação inicial do sujeito. Conforme entendimento de Pimenta (1997), a identidade docente é parte de um processo de construção do sujeito historicamente situado. Assim, requer aprofundamento nos dados coletados para investigar como a atuação dos professores participantes, a sua relação com seus colegas professores, com o local de trabalho, com seus gestores e suas concepções sobre o estabelecimento do processo identitário.

Para determinar o alcance das possíveis respostas, retorna-se à questão norteadora: "c*omo é estabelecido o processo de desenvolvimento da identidade docente de profissionais da Computação que não possuem formação inicial na Licenciatura em Computação e que atuam na EPT?*". Embora os sujeitos participantes não possuam formação inicial (graduação) em curso de Licenciatura, apurou-se que eles sentiram a necessidade de buscar conhecimentos pedagógicos. Dois sujeitos concluíram cursos de pós-graduação *lato sensu* cujo conteúdo, conforme informaram, tangencia a Educação e o fizeram de livre vontade.

Quanto à atuação dos professores, apurou-se que suas práticas são baseadas nas tentativas de erros e acertos, focalizadas principalmente no conteúdo das disciplinas de

Computação, de modo a inviabilizar que o estudante possa se apropriar de forma autônoma e reflexiva acerca do assunto estudado, para enfim produzir novos conhecimentos.

Refletem sobre suas práticas "empíricas" e compreendem a necessidade de mudanças, reconhecem que precisam de suporte pedagógico e orientações basilares acerca dos objetivos que orientam a escola onde atuam.

A carência de suporte pedagógico, aliada à falta de documentos formais que reflitam a identidade da escola, culminam por prejudicar o desenvolvimento da identidade docente dos professores de computação sem formação inicial na Licenciatura atuantes na EPT.

5. Considerações Finais

Este artigo apresentou alguns resultados de uma investigação sobre o processo de desenvolvimento da identidade docente por professores de Computação sem formação pedagógica em nível de Licenciatura, atuantes na Educação Profissional de Nível Médio em uma escola pública do estado da Bahia.

Apurou-se que os desafios para desenvolvimento da identidade de professor por esses profissionais minimizam as chances de sucesso didático-pedagógico, por causa de diversos fatores (causais) relacionados ao ambiente, às concepções de educação e à ausência de formação inicial no campo da Licenciatura. Esses fatores se refletem (consequência) nos modelos de mediação didática adotados por eles.

A falta de formação inicial que capacite o profissional a lecionar se mostra como um desafio a ser superado. Neste sentido, é possível que a Licenciatura em Computação contribua de forma significativa por formar profissionais qualificados e habilitados para o exercício do magistério. Além disso, é preciso mais estudos acerca do tema, de modo a alargar o campo epistemológico sobre a formação de professores de Computação, em um diálogo constante com outras áreas do conhecimento, estabelecendo a Educação em Computação como de fato um campo de pesquisa (MATOS & SILVA, 2012).

Outra desafio a ser superado é a falta da identidade na escola. Uma escola, conforme afirma Veiga (2002), necessita de um planejamento formal, mas que não seja apenas um arranjo para cumprir as determinações dos órgãos superiores, mas um documento construído democraticamente por toda comunidade atuante de forma a refletir sua identidade e, assim, proporcionar aos seus professores o desenvolvimento de suas próprias identidades profissionais através da vivência, segundo as orientações definidas para o coletivo escolar.

Como trabalho futuro, sugere-se ampliar a investigação acerca do desenvolvimento da identidade docente em Computação, a partir das concepções filosóficas, sociais e políticas que permeiam o ambiente escolar e os professores que atuam na Educação em Computação.

Referências

Bardin L. (1979). *Análise de conteúdo*. Lisboa: Edições 70.

Barreto, M. A. M. (2009). Ensinando a ensinar - a importância do modelo na formação de professores. *Revista Práxis*, v. 1, n. 1. Disponível em: https://goo.gl/aXRvb9. Acessado em : 21/04/2018.

Brasil (2004). *Decreto n° 5154*, de 23 de julho de 2004. Disponível em: https://goo.gl/YXTDdk. Acessado em: 16/04/2018.

Cardoso, A. A.; Pino, M. D.; Dorneles, C. (2012). Os saberes profissionais dos professores na perspectiva de Tardif e Gauthier: contribuições para o campo de pesquisa sobre os saberes docentes no Brasil. In: *Anais da IX ANPED Sul, Seminário de Pesquisa em Educação da Região Sul*. Disponível em: https://goo.gl/iPQfe1. Acessado em: 17/04/2018.

D'ÁVila, C. (2005). A mediação didática na história das pedagogias brasileiras. *Revista FAEEBA*, v. 14, n. 24, p. 217–238.

_____. (2008). Formação docente na contemporaneidade: limites e desafios. *Revista FAEEBA*, v. 17, n. 30, p. 33-41.

_____. (2013). *Decifra-me ou te devorarei*: o que pode o professor frente ao livro didático?. Salvador: Eduneb.

Freire, P. (2000). *Pedagogia do oprimido*. 29ª. ed. São Paulo, SP: Paz e Terra.

Gariglio, J. A.; Burnier, S. (2012). Saberes da docência na Educação Profissional e Tecnológica: um estudo sobre o olhar dos professores. *Educação em Revista*, Belo Horizonte, v. 28, n. 01, p. 211-236.

Gatti, B. A. (2013). Implicações e perspectivas da pesquisa educacional no Brasil contemporâneo. *Cadernos de Pesquisa*, Fundação Carlos Chagas, n. 113, p. 65-81.

Junior, E. V. B.;Lopes, K. M.; Balduino, A. R.;de Sousa, J. M. (2014). O desafio da prática docente sem a formação de licenciatura em computação. In: *Anais da*, *V Jornada de Iniciação Científica e Extensão (*V JICE), Tocantins. Disponível em: https://goo.gl/xBPWXS. s/p. Acessado em: 20/03/2018.

Lima, A. A. B. (2001). *Educação Profissional para quê?*: construindo a formação dos trabalhadores para além do falso consenso. Salvador: UFBA/FACED.

Ludke, M.; André, M. E. D. A. (2005). *Pesquisa em educação*: abordagens qualitativas. São Paulo, SP: E.P.U.

Matos, E.; Silva, G. (2012). Currículo de licenciatura em computação: uma reflexão sobre perfil de formação à luz dos referenciais curriculares da SBC. In: *Anais do XXXII Congresso da Sociedade Brasileira de Computação*. XX Workshop sobre Educação em Computação (WEI). s/p.

Matos, E. (2013). Identidade profissional docente e o papel da interdisciplinaridade no currículo de licenciatura em computação. *Revista Espaço Acadêmico,* v. 13, n. 148, p. 26-34.

Moura, D. H (2015). A formação de docentes para a educação profissional e tecnológica. *Revista Brasileira da Educação Profissional e Tecnológica*, v. 1, n. 1, p. 23-38.

Nóvoa, A. (Org.) (2013). *Vidas de Professores*. 11ª ed. Lisboa: Porto Editora.

Pereira, L. A. C. (2009). A formação de professores e a capacitação de trabalhadores da educação profissional e tecnológica. Portal MEC, v. 3. p. 1-9. Disponível em: https://goo.gl/4nNYpx. p. 1-6. Acessado em: 20/03/2018.

Pimenta, S. G. (1997). Formação de professores - saberes da docência e identidade do professor. *Nuances*, v. 3, p. 5-14.

Pimenta, S. G..; Anastasiou, L. G. C. (2014). *Docência no Ensino Superior*. 3ª ed. São Paulo: Cortez.

Pimenta, S. G.; Ghedin. E. (2002). *Professor reflexivo no Brasil*: gênese e crítica de um conceito. São Paulo: Cortez.

Ramos, M. N. (2008). Concepção do ensino médio integrado. In: *Seminário sobre o ensino médio.* RN: Secretaria Estadual de Educação - Natal Disponível em: https://goo.gl/1ne8TW. Acesso em: 09/05/2018.

Silva, R. S. (2017). *Formação docente e as práticas do professor de computação na educação profissional*. Dissertação (Mestrado em Ciência da Computação), Universidade Federal da Bahia, Salvador/BA.

Tardif, M. (2012). *Saberes Docentes e Formação Profissional*. Petrópolis, RJ: Vozes.

Veiga, I. P. A. (2012). Didática: uma retrospectiva histórica. In: VEIGA, I. P. A. (*Org*.). *Repensando a Didática*. 29ª ed. Campinas, SP: Papirus.

_____ (2002). *Projeto Político Pedagógico da Escola*: uma construção possível. 14ª. ed. Campinas, SP: Papirus.

Idealizando Jogos Digitais de Pensamento Computacional a Partir do *Bebras Challenge*: Um Estudo Exploratório.

Jéssica Laisa[1], Eduardo Henrique [1], Wendell Oliveira[1]
[1]Programa de Pós-Graduação em Sistemas Computacionais – PPgSC
Universidade Federal do Rio Grande do Norte – UFRN
Campus Universitário Lagoa Nova –Natal/RN – Brasil

jessicalaisajl@gmail.com, eduardoaranha@dimap.ufrn.br,

wendell.cmd@gmail.com

Abstract. *National and international initiatives have been incorporating computational thinking in basic education. One of them is the Bebras Challenge, which presents a series of questions that exercise and evaluate computational thinking. In the context of this initiative, we present an exploratory study carried out with 19 students graduating from a federal university in order to investigate whether the Bebras issues are good sources for game design construction. With this, this work intends to answer the following research questions: What is the quality of game designs produced? What is the effort required to produce game designs? What are the difficulties presented by the students? The results show that it is possible to develop good game designs based on Bebras issues, which are a good source of inspiration.*

Resumo. *Iniciativas nacionais e internacionais vêm inserindo pensamento computacional no ensino básico. Uma delas é o Bebras Challenge, que apresenta uma série de questões, que exercitam e avaliam o pensamento computacional. No contexto dessa iniciativa, apresentamos um estudo exploratório realizado com 19 alunos graduandos de uma universidade federal, com o objetivo de investigar se as questões do Bebras são boas fontes para construção de game design. Com isso, este trabalho pretender responder as seguintes questões de pesquisa: Qual a qualidade dos game designs produzidos? Qual o esforço necessário para realizar a produção dos game designs? Quais as dificuldades apresentadas pelos alunos? Os resultados alcançados mostram que é possível desenvolver bons game designs a partir das questões do Bebras, mostrando-se estas uma boa fonte de inspiração.*

1. Introdução

Cada vez mais, a tecnologia adentra nas instituições de ensino e é uma forte ferramenta para estimular o desenvolvimento dos alunos, propiciando engajamento, motivação e auxiliando os professores nos trabalhos em sala de aula. O autor Kenski (2013) ressalta que assim como a educação, a tecnologia também é poder. E dentre as tecnologias usadas nas escolas uma que tem grande potencial de engajamento e participação dos alunos estão os jogos digitais. [Prensky, 2001] relata que a utilização de jogos digitais na educação traz grandes vantagens, como a possibilidade de motivar o jogador, pois um aluno se divertindo aprende melhor.

Porém, antes dos jogos educativos digitais serem desenvolvidos, eles passam por um processo de criação e uma delas é logo no inicio, o desenvolvimento do documento de game design. [Leite e Mendonça, 2013] afirmam a importância da definição do documento de game design, o qual contém as especificações do jogo e onde são descritas e explicadas às características principais do jogo, como jogabilidade, controles, interfaces, personagens, armas, golpes, inimigos, fases e demais aspectos gerais do projeto.

O objetivo deste artigo é trazer um estudo exploratório que verificar como atividades ou testes de habilidades podem ser boas fontes para construção de game design para se trabalhar pensamento computacional. Nesse contexto, este trabalho levanta as seguintes questões: Qual a qualidade dos game designs produzidos? Qual o esforço necessário para realizar a produção dos game designs? Quais as dificuldades apresentadas pelos alunos?

Para atingir esse objetivo, realizamos neste artigo um estudo exploratório com alunos de graduação que idealizaram jogos digitais educativos a partir de questões do teste internacional BEBRAS, que trabalha com pensamento computacional.

Sendo assim, as próximas seções deste trabalho estão organizadas da seguinte maneira: a Seção 2 discute sobre Game Design, enquanto a Seção 3 apresenta o Pensamento Computacional. Já a Seção 4 descreve a metodologia utilizada, enquanto os resultados são apresentados na Seção 5. Por fim, as considerações finais são tratadas na Seção 6.

2. Game Design

Conforme o autor de [Pedersen, 2013], o documento de game design do jogo, ou GDD (*game design documen*t) é uma ferramenta textual que descreve todas as características de um jogo, desde informações básicas, conceitos, passando por personagens e cenários, até informações mais detalhadas sobre o jogo.

Perucia *et al.* (2005) afirmam que o GDD pode ser definido como "um documento que descreve as características do game design de modo abrangente". Já o autor de [Schuytema, 2008, p. 100] considera o game design como o coração e a alma de todos os documentos que giram em torno de um game em desenvolvimento e ressalta a importância dele, por apresentar uma descrição detalhada do jogo. Vale ressaltar que o game design é uma das primeiras etapas no desenvolvimento de jogos e que este documento pode ser modificado em qualquer momento da produção dos jogos.

3. Pensamento Computacional

O conceito de Pensamento Computacional foi inicialmente relatado por [Wing, 2006] para tratar da Ciência da Computação e de suas aplicações, que está relacionado desde a estruturação do raciocínio lógico até o comportamento humano e na resolução de problemas. Este tipo de pensamento pode ser visto nos processos de leitura, escrita e matemática como parte integrante da habilidade analítica das crianças desde a idade infantil. Outros autores como [Gomes e Melo, 2013] afirmam que o Pensamento Computacional tem uma contribuição na resolução de problemas, possibilitando que os indivíduos possam utilizar a computação nas diversas ações do cotidiano.

Assim, os conceitos do Pensamento Computacional estão nos princípios da computação e não em suas tecnologias. Peter Lee em *National Research Council* (2010) descreveu o Pensamento Computacional como o estudo de mecanismos da inteligência humana que podem descrever aplicações e modelos que ajudem a tratar a complexidade.

O Pensamento computacional também pode ser definido como o pensamento analítico, compartilhando: com a matemática, a resolução de problemas; com a engenharia, modelagem e projeto; e com a ciência, a compreensão sobre computabilidade, inteligência, mente e comportamento humano [Wing 2008].

A *Computer Science Teachers Association* [CSTA 2011] ressalta um conjunto de competências que são trabalhadas nas atividades do pensamento computacional. Essas competências incluem a confiança em lidar com a complexidade; persistência ao trabalhar com problemas difíceis; tolerância em lidar com ambiguidade; capacidade de lidar com problemas em aberto; capacidade de se comunicar e trabalhar em grupo para atingir um objetivo.

3.1 Testes de habilidades – *Bebras Challenge*

O "*Bebras*" é uma iniciativa internacional com o objetivo de promover a Informática e o pensamento computacional entre estudantes de todas as idades. Os participantes geralmente são supervisionados por professores que podem integrar o desafio da Bebras em suas atividades de ensino. O desafio é realizado em escolas usando computadores ou dispositivo móvel [Bebras 2017].

A ideia do "*Bebras*" surgiu na Lituânia, pela Prof. Valentina Dagiene da Universidade de Vilnius. *Bebras* é uma palavra lituana para "castor". Um dos objetivos de Valentina Dagiene foi estabelecer a Bebras como uma iniciativa internacional em informática nas escolas. Desde do início, vários países europeus se juntaram ao "Bebras". De acordo com os fundadores, este teste pretende interessar crianças, jovens e adolescentes em problemas típicos de Ciência da Computação, não requerendo qualquer conhecimento como pré-requisito [Dagiene 2008].

Atualmente o desafio é organizado em mais de 30 países e tem o objetivo de levar estudantes de todo o mundo a serem entusiasmados com a computação. Cada participante do desafio recebe 45 minutos para responder 15 questões de múltipla escolha que se concentram no tema de pensamento computacional e lógico.

4. Metodologia

Para a execução deste trabalho, realizou-se inicialmente uma pesquisa de caráter exploratório de instrumentos e testes avaliativos que trabalhassem o pensamento computacional. O desafio *Bebras* foi o selecionado por ser um teste renomado e que já é adotado por mais de 30 países no Mundo.

Realizou-se então um estudo exploratório na produção de game designs que trabalhassem pensamento computacional. Documentos de game design foram produzidos e analisados, junto com formulários e observações realizadas. Na seção

seguinte, é apresentado o planejamento do estudo exploratório como também as fases realizadas.

4.1 Fases do estudo

O estudo exploratório foi organizado nas seguintes fases:

1. Chamada para os alunos para a aplicação da pesquisa;
2. Selecionar questões do Teste *Bebras*;
3. Explicação e apresentação do Documento de *Game Design*;
4. Criar o documento de *Game Design* Educativo;

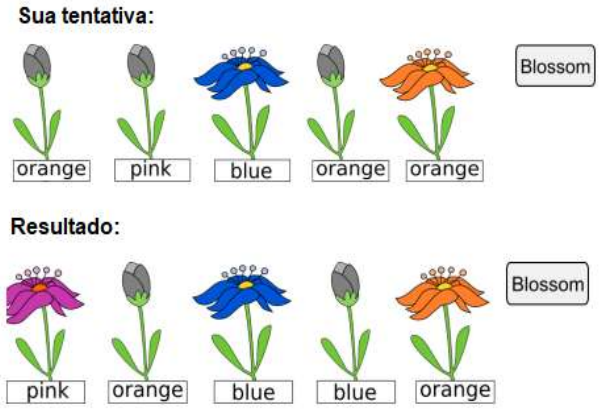
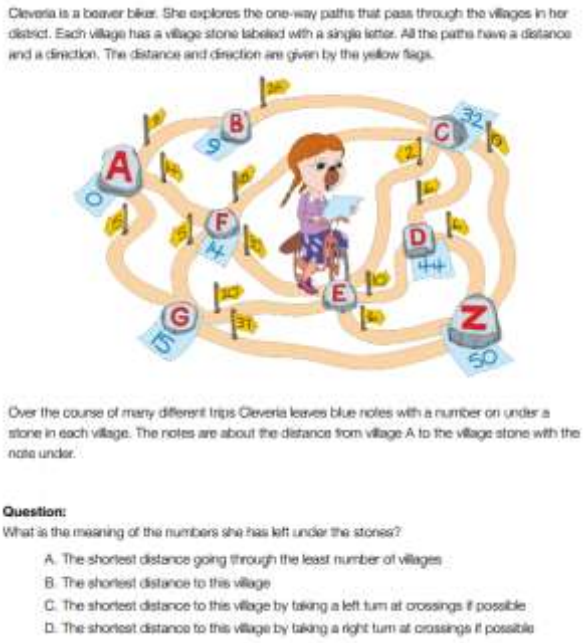
As fases são descritas a seguir. A fase 1 e 2 ocorreu em paralelo ao longo de duas semanas, enquanto as fases 3 e 4 ocorreu de forma sequencial na manhã de um único dia, ou seja, em 4 horas contínuas.

Fase 1: Chamada para os alunos para a aplicação da pesquisa: Nesta etapa os pesquisadores do laboratório divulgaram no portal da universidade uma chamada para alunos dos cursos de graduação que tivessem interesse em criar o game design para Jogos Digitais de Pensamento Computacional. Como resultados inscreveram-se e compareceram 19 alunos de graduação de diferentes áreas, conforme descrito na próxima seção.

Fase 2: Selecionar questões do Teste *Bebras*: Nesta etapa, dois pesquisadores selecionaram três questões do desafio *Bebras* referentes ao ano de 2016, com graus de dificuldades diferenciados: fácil, médio e difícil, classificação dada pelo próprio *Bebras*. As questões selecionadas foram: Bike Paths, Broken Window e Blossom. Para cada questão existem respostas e comentários disponibilizados pelo caderno de resposta do teste *Bebras*, os quais foram utilizados no estudo para facilitar o entendimento da questão e consequentemente a criação do game design. Na Tabela 1 são apresentadas as questões selecionadas com seus respectivos enunciados.

Tabela 1: Questões selecionadas do desafio *Bebras*.

Nome da questão	Enuciando da questão	Descrição
Broken Window		A questão se baseia em dar determinadas Características, para descobrir quem quebrou a janela.

Blossom		Descobrir as cores de cada flor com um número x de tentativas, ao acertar uma cor a flor desabrocha com a cor correspondente.
Bike Paths		Descobrir o menor caminho, dado as distâncias, em cada ponto é colocado a distância do ponto A até as demais.

Fase 3: Explicação e apresentação do Documento de *Game Design*: Esta etapa foi aplicada no início do dia marcado com os alunos inscritos para o desenvolvimento do estudo. Por ter alunos de diferentes graduações, foi explicado como fazer um documento de Game Design apresentando um modelo de exemplo utilizado pelos os pesquisadores, facilitando assim a compreensão dos alunos.

Fase 4: Criar o documento de *Game Design* Educativo: Nesta etapa cada aluno de forma individual elaborava suas ideias de jogos a partir das questões e produziam os documentos de *Game design.*

4.2 Perfis dos alunos

Um formulário foi passado no início do estudo para verificar o perfil e as experiências prévias dos estudantes. Os detalhes são apresentados a seguir. O estudo contou com a participação de 19 estudantes com previsão de término de curso entre 2017.2 e 2020.

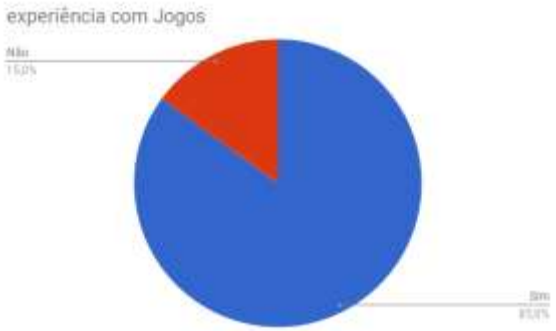
Dos 19 alunos participantes, 17 são homens e 2 são mulheres. O perfil dos alunos que se interessaram por participar da pesquisa era diferente, pois os inscritos eram graduandos de diferentes cursos. A seguir é apresentada a tabela 2 referentes à distribuição dos alunos entre seus cursos.

Tabela 2. Cursos respectivos dos alunos inscritos.

	Curso	Quantidade alunos
Perfil dos Alunos	Tecnologia da Informação	14
	Química	1
	Licenciatura em Filosofia	1
	História	2
	Licenciatura em Física	1
	Total de Alunos	**19**

Sobre experiência prévia, os estudantes poderiam relatar se possuíam algum contato ou experiência com jogos digitais, seja em disciplinas ou até mesmo na produção de jogos comerciais. Dos 19 alunos inscritos, apenas 3 relataram não possuir nenhuma experiência, ou seja, nunca tiveram contanto com universo dos jogos digitas. O gráfico abaixo mostra esse percentual.

Gráfico 1: Experiência com Jogos.



5. Análise dos resultados

Nesta seção são apresentados os resultados referentes à aplicação do estudo no laboratório com os alunos. Os resultados são provenientes das analises realizadas por meio de observações do comportamento dos alunos, da produção dos documentos de game design e nos formulários aplicados.

Qual a qualidade dos *game design* produzidos?

Os documentos de game design produzidos foram avaliados por 5 pesquisadores do laboratório com experiência na criação de game design. Foram verificados os seguintes aspectos de avaliação: baseados nas questões do *Teste Bebras;* criatividade na criação do enredo do jogo; preenchimento adequado do *Game Design (GDD).*

Tabela 3.Critérios de avaliação do Game design

Número do Critério	Critérios de avaliação do game design
1	Baseados nas questões do Teste *Bebras*
2	Criatividade na criação do enredo do jogo
3	Preenchimento adequado do G*ame Design*

No critério 1, verificamos se os game designs realmente adotaram como base o Teste *Bebras.* Nesse ponto, 14 alunos conseguiram realizar a criação do game design inspirados nas questões do teste. Entretanto, quatro alunos não compreenderam as explicações e acabaram levando em consideração outros instrumentos na criação do seu game design. Alguns alunos ao invés de criar os *Games Designs* das questões a partir do Teste *Bebras* procuraram outros instrumentos para realizar o game design, como por exemplo: questão do Enem, outros testes de avaliações e jogos já conhecidos no mercado.

Com relação ao critério 2, de criatividade, percebesse que relacionando o contexto de seus conhecimentos específicos em disciplinas como física, matemática, meio ambiente, contribui no processo criativo de desenvolver ideias para a criação do jogo. Com isso, acabou por agregar valor criativo e pedagógico aos game design das questões.

No critério 3, preenchimento do Documento de *Game design*, todos os alunos realizaram o preenchimento adequado do documento. Neste campo, foi analisado também a organização, explicação e a qualidade no preenchimento. Abaixo, segue exemplo de um o campo descrição, referente a o documento do game design realizado por um aluno.

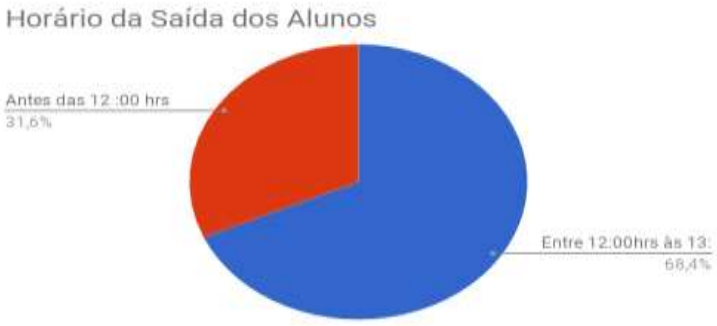
Tabela 4. Exemplo do campo "descrição" do *Game Design* da questão *Broken Window*.

DESCRIÇÃO:	O jogador será um membro da Polícia Espacial. Uma jovem alien será assaltada e dirá as características do assaltante. Essa informação aparecerá no canto direito inferior da tela. Aparecerão, então, vários E.T.s num cenário, que são os "suspeitos". Clicando num suspeito, o jogador irá abordá-lo. Se ele for o criminoso, será preso; mas se for um civil inocente, o jogador perderá 1 chance de abordagem. Se ele abordar 3 inocentes, será demitido (Game Over). Para achar o criminoso, o jogador deverá encontrar o suspeito que tem características que batem com a descrição dada pela vítima.

Qual o esforço necessário para realizar a produção dos game design?

Para avaliar o esforço dos alunos na criação do documento de game design foi dado inicialmente aos alunos o tempo de 4 horas para realizar o game design das três questões passadas. Porém, dentre os 19 alunos, apenas 31,6% conseguiram terminaram no tempo estipulado. Os demais 68,4% dos alunos solicitaram mais tempo para terminar os game designs, alegando pouco tempo para a atividade. O gráfico a baixo ilustra o horário e a porcentagem referente à quantidade de alunos.

Gráfico 2: Horários da Saída dos alunos.



A partir desta questão, pode-se perceber que o tempo determinado para o estudo foi insuficiente, pois a maioria dos alunos como mostra o gráfico necessitaram de mais tempo. O estudo inicialmente foi estipulado de 4 horas, iniciando às 8 horas e terminando às 12 horas, porém durante a aplicação dos estudos alguns alunos solicitaram mais tempo e os pesquisadores dispuseram de mais 1 hora e com isso, dando assim um tempo de 5 horas. Além disso, mais tempo deve implicar em uma melhoria na qualidade do game design ou de viabilizar melhor o processo de criatividade, sem a pressão do tempo.

Nesta questão analisamos também o relato dos alunos, que ressaltaram como as questões selecionadas do Teste *Bebras* contribuíram para que não houvesse tanto esforço para delimitar o escopo a qual o game design seria criado. Desta forma, as questões ajudaram no processo de criação do game design e até mesmo no processo de criatividade. Porém, em contrapartida, alguns alunos não conseguiram concluir os 3 (GDD) questões, mesmo com a hora a mais adicionada.

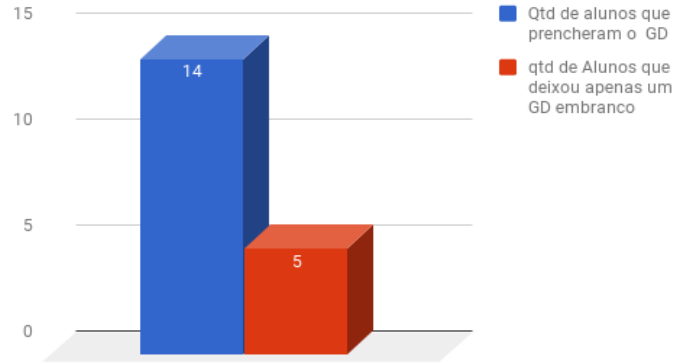
Quais as dificuldades apresentadas pelos alunos?

Por meio das observações na aplicação do estudo como também, pelo formulário respondido e pelos *game designs* produzido, foi visto que alguns alunos apresentaram dificuldade como:

- Compreender o objetivo do estudo, que era de utilizar as questões do Teste *Bebras* como base para realizar os games design. Dos 19 alunos 4 acabaram por pesquisar em outros instrumentos avaliativos questões para produzir o *Game Design*. Vale ressaltar que estes quatros apresentavam a semelhança de não serem do curso de Tecnologia da Informação, podendo assim, ser uma relação que pudessem tê-los prejudicado na compreensão.
- Durante estudo alguns alunos apresentaram dúvidas e solicitaram o esclarecimento dos pesquisadores mediadores. Cinco alunos fizeram em vários momentos a mesma pergunta sobre como usar as questões para elaborar o *game design*.
- Alguns alunos solicitaram mais tempo para realizar os games design, alegando que tempo estava insuficiente.
- Apresentaram dificuldade com alguns campos do game design, acabando por deixá-los sem preenchimento.
- Dentre as questões apresentadas, os alunos relataram maior dificuldade de compreensão para criar o game design com a questão 3 - Bike Paths.

- Dos 19 alunos, cinco alunos deixaram de fazer o Game Design de um dos três jogos que deveria ser feito, como apresentado no gráfico 2.

Gráfico 3: Quantidade alunos que realizaram a criação dos Games Designs.



6. Conclusões

Este trabalho apresentou um estudo exploratório de como produzir a criação de game design de Jogos Digitais de Pensamento Computacional a partir de questões do Teste internacional *Bebras*. Podemos perceber que alunos de diferentes graduações se interessaram por participarem do convite do estudo e como a utilização de uma base para gerar o game design pode ajudar aos alunos a criarem o game design.

Com a utilização de questões do Teste *Bebras,* pode-se garantir a criação dos game design voltados ao pensamento computacional. Essa fonte de inspiração ajudou os alunos a produzir de maneira mais concisa e eficaz os documentos, pois muitos alegaram a contribuição das questões selecionadas, que os ajudaram no processo. De fato, os participantes relataram que se não houvesse às questões levariam mais tempo e consequentemente mais esforço para estabelecerem um escopo para criação dos seus games designs.

A contribuição deste trabalho se dá no contexto atual de cada vez mais trabalhar os aspectos educacionais e estimular o pensamento computacional na produção dos jogos, tendo em vista sua importância no ambiente educacional de estimular a resolução de problemas e desenvolver raciocínio lógico. O trabalho também investiga a produção de game design a partir de questões de testes avaliativos reconhecidos, como uma forma de facilitar esse processo.

Esperamos que este estudo possa de alguma forma servir de auxílio para os atuais e futuros envolvidos com o tema, bem como, contribuir para o avanço de buscar métodos para produção de game design na produção de jogos digitais de pensamento computacional. Pretendemos assim, que com este trabalhos incentive pesquisas para difundir o pensamento computacional como possibilitar à aproximação o universo dos jogos educativos, consequentemente de meios para aplicações da tecnologia aos alunos de outras áreas.

Como trabalhos futuros, a proposta é elaborar um estudo mais completo selecionando mais questões do teste *Bebras* como também de outros testes avaliativos.

Pretende-se também aumentar o número de alunos para realização de pesquisa como estender o estudo em outras instituições, bem como fomentar não só a criação de game designs, mas também do desenvolvimento e aplicação desses jogos na educação.

Referências

Balasubramanian, Nathan; Wilson, Brent G. Games and Simulations. In: Society For Information Technology And Teacher Education International Conference, 2006. Proceedings…v.1. 2006.

CSTA&ISTE (2011). Computational thinking teacher resources. http://csta.acm.org/curriculum/sub/compthinking.html. acesso: outubro/2017

Gomes, T. e Melo, J. (2013). O pensamento computacional no ensino medio: Uma ´ abordagem blended-learning. In Anais do 21o WEI -XXXIII CSBC. Maceio, AL– Brasil.

Kenski, Vani Moreira. (2013) Tecnologias e tempo docente. São Paulo: Papirus Editora

Prensky, M., 2001. Digital game-based learning. New York, McGraw-Hill.

R. Pedersen. Game design foundations. 1.ED. Sudbury: Wordware publishing, INC. 2003.

A.S. Perucia, A. C. Berthêm, G. L. Bertschinger e R. R. C. Menezes. Desenvolvimento de Jogos Eletrônicos: teoria e prática. São Paulo: Novatec Editora, 2005.

V. Dagiene and G. Futschek, "Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks," in Informatics Education - Supporting Computational Thinking, ISSEP 2008, Torun, Poland, July 1-4, 2008: Springer, 2008, pp. 19-30

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3):33–35.

Estágio Docente de Licenciatura de Computação: Um Ensaio do Ensino de Computação no Ensino Fundamental

José Andersson S. da Silva[1], Hérikles Vinícyus F. Cordeiro[1], Sônia R. Fortes da Silva[1], Anselmo L. Gomes[1]

[1]Universidade de Pernambuco, Campus Garanhuns (UPE)
CEP 55.294-902 – Garanhuns – PE – Brasil

{anderssonsoares97, heriklescordeiro, fortes.sonia, anselmodsi }@gmail.com

Abstract. *This article reports on the experience of undergraduates in Computing at the University of Pernambuco, Multicampi Garanhuns, during the course of Supervised Internship II, that was held at the School of Application Professora Ivonita Alves Guerra, at the Campus University of Pernambuco in Garanhuns. It describes the activities carried out in the classroom by these authors who acted as teachers, with the use of expository methodology, the classes consisted of theoretical explanations on topics related to computing, followed by practical activities. Allied to the use of the Construct 2 tool, we worked on the development of games, using the Problem Based Learning (PBL) methodology during the classes that constituted the Extension dimension. This experience verified the importance of the Supervised Internship for the teaching of Computing, thanks to the professional experience provided, regarding the learning developed by the students.*

Resumo. *Este artigo relata a experiência vivenciada pelos licenciandos em Computação da Universidade de Pernambuco - UPE, Multicampi Garanhuns, durante a disciplina de Estágio Supervisionado II, realizada na Escola de Aplicação Professora Ivonita Alves Guerra, no Campus da UPE em Garanhuns. Descreve as atividades realizadas em sala de aula por estes autores que atuaram como professores, com o uso de metodologia expositiva. As aulas consistiram em explicações teóricas sobre tópicos relacionados à computação, seguidas de atividades práticas. Aliados à utilização da ferramenta Construct 2, trabalhamos o desenvolvimento de jogos, utilizando a metodologia Problem Based Learning (PBL) durante as aulas que constituíram a dimensão Extensão. Esta experiência constatou a importância do Estágio Supervisionado para o ensino de Computação, graças a experiência profissional proporcionada, quanto à aprendizagem desenvolvida pelos estudantes.*

1. Introdução

O ensino de Computação é indicado em todos os níveis da Educação Básica, da Educação infantil até o último ano do ensino médio, de acordo com a *Computer Science Teacher Association* (CSTA, 2005). Existe atualmente uma discussão no Brasil envolvendo a Sociedade Brasileira de Computação (SBC), e o Grupo de Trabalho de Ensino de Computação e Informática na Educação Básica (GT3), para a implementação do componente curricular da Computação na Base Nacional Comum Curricular[1] (BNCC)

[1] http://basenacionalcomum.mec.gov.br/

área de conhecimento que ajuda a construção do Pensamento Computacional em estudantes e a inserção do jovem no Ecossistema e na Cultura Digital. Para tal competência e habilidade na construção desses eixos pelos educandos, existe no Brasil o Curso de Licenciatura em Computação para fomentar e contribuir para a consolidação do que seria Educação e Computação, bem como, o seu papel para a formação de uma sociedade 4.0, será exigido dos profissionais o desenvolvimento de uma visão multidisciplinar (Freitas, 2016).

Considera-se que a introdução de Computação, enquanto ciência, na Educação Básica é de fundamental importância e relevância (CSTA, 2011) para o desenvolvimento do cidadão da sociedade contemporânea. A sua inexistência limita a capacidade dos educandos de lidar com os avanços da sociedade cada vez mais globalizada e tecnológica, privando os educandos de múltiplos caminhos profissionais futuros, que orientam inovação, autonomia, competência e habilidades tecnológicas para programar e criar artefatos que auxiliem o ser humano em educação, saúde e outros setores de produção. A inexistência para um grupo e a existência para outro grupo de educandos desta formação cria um fosso entre a exclusão e inclusão dos formandos para o seu exercício profissional. Aqueles que tiveram contato com o ensino de computação no ensino fundamental terão facilidade no curso técnico estadual e federal, sendo um diferencial para o êxito na última etapa da Educação Básica, na educação superior e na vida social e profissional.

Porém, no Brasil, esta área de conhecimento não faz parte da BNCC conforme a proposta da SBC, pelo desconhecimento dos educadores sobre esta área, pela quantidade de propostas existentes na BNCC para o currículo escolar e pela Lei Fiscal que impossibilita Estado e Município de aumentarem seus custos com os professores, impossibilitando contratações de profissionais de Licenciatura em Computação. A sociedade solicita a existência de cursos de Licenciatura em Computação na graduação, mas o Estado não materializa o seu exercício, excluindo a sociedade brasileira deste ensino na Educação Básica. Expropria-se o estudante deste conhecimento, excluindo-o da evolução tecnológica e do êxito profissional, aumentando a força de trabalho desqualificada e barata.

A partir destas visões, muitas vezes antagônicas, o Estágio Supervisionado dos cursos de Licenciatura em Computação é considerado um dos principais articuladores de formação profissional do ensino de Computação no ensino fundamental e médio. Em diversas vezes, o Estágio Supervisionado proporciona aos educandos das escolas públicas, o contato inicial com o ensino de Computação, ministrado por professores da área de Educação, e de Computação, sendo muitas vezes sua proposta interdisciplinar ou com base no Referencial de Ensino de Computação na Educação Básica (SBC, 2017).

Com isto, a partir da experiência docente orientada, estudantes da Licenciatura em Computação lançam mão dos seus conhecimentos, para exercer a profissão e averiguar os problemas deste exercício, resultando em futuros docentes mais autônomos, que estruturam soluções de situações-problema, construindo conteúdos educacionais e artefatos de auxílio para tratar temas atuais, apresentando inovações na forma de aprender e de compreender os fenômenos dos saberes escolares.

Nesse contexto, o presente artigo relata a experiência de um grupo de discentes do Curso de Licenciatura em Computação da Universidade de Pernambuco, Multicampi Garanhuns, que através do projeto de Estágio Supervisionado, atuaram como docentes na Escola de Aplicação Professora Ivonita Alves Guerra, no Campus da Universidade de Pernambuco. As atividades do Estágio Supervisionado foram subdivididas por meio das

atividades de Ensino e Extensão, cada uma contendo um objetivo específico a ser alcançado, integradas nas atividades da docência.

2. Estágio em Licenciatura em Computação

De acordo com Kulcsar (2012), os estágios supervisionados são considerados uma parte importante da relação trabalho-escola e da teoria-prática, por propiciar ao docente em formação uma visão da realidade a qual será submetido após sua graduação, possibilitando com os conhecimentos aprendidos na graduação e na vida escolar, exercer a profissão em sala de aula, o que, até então, fora transmitido no decorrer do curso somente no aspecto teórico (PIMENTA, 2001).

De acordo com a Lei Nº11.788, de setembro de 2008, conceitua-se estágio, sendo

> o ato educativo escolar supervisionado, desenvolvido no ambiente de trabalho, que visa à preparação para o trabalho produtivo de educandos que estejam frequentando o ensino regular em instituições de educação superior, de educação profissional, de ensino médio, da educação especial e dos anos finais do ensino fundamental, na modalidade profissional da educação de jovens e adultos. (BRASIL, 2008, Parágrafo 1º).

Este expõe o objetivo de proporcionar ao aluno o exercício de suas habilidades, aplicando seus conhecimentos acadêmicos em situações práticas, adquirindo uma experiência que sirva de base para a construção de uma visão crítica do exercício de suas habilidades como destaca Oliveira e Cunha (2006). Espera-se que, com isso, o aluno tenha a opção de incorporar atitudes práticas e adquirir uma visão crítica de sua área de atuação profissional.

Neste âmbito, percebe-se a importância na formação do licenciando em Computação, ao inseri-lo na posição de professor de Computação no ensino fundamental, tendo em vista as discussões pelo qual passa o país, com propostas de implementações do ensino de conceitos básicos nas escolas, que tende a seguir alguns países que têm implantado um currículo mínimo de Computação em suas escolas (CSTA, 2005).

Porém, no Brasil, esses debates sobre a implantação de um currículo de Computação na Educação Básica ainda não tiveram avanços significativos desde seu início. Devido a isso, este artigo relata a execução do projeto da disciplina de Estágio Supervisionado, que viabilizou de forma experimental a implantação da disciplina no ensino fundamental do 6º ao 8º ano, visando a importância de projetos como esse, que geram resultados importantes para discussões sobre a implementação na BNCC.

3. Computação e Educação

Com o advento da computação seu impacto na sociedade proporcionou transformações sociais, mesmo com apenas 100 anos de existência como ciência. A Computação tem como ponto principal a ferramenta do raciocínio lógico para o tratamento de situações-problemas, está que não se restringe apenas no âmbito computacional. Tendo conhecimento dos benefícios proporcionado por essa ciência, vários pesquisadores estão direcionando e publicando seus trabalhos a fim relatar as experiências e corroborar com a importância da inclusão da computação na Educação Básica.

Segundo França (2014) a sala de aula não condiz com a necessidade que a sociedade moderna exige de seus cidadãos. A educação brasileira contém indícios de que mantém o padrão de 50 anos atrás: Carteiras enfileiradas, quadro branco, livros, e o professor atuando como centro das aulas, e como única fonte de conhecimento. Tal

metodologia não ajuda o aluno a construir a capacidade de aprender por si próprio, buscando soluções para suas dúvidas e dificuldades.

França e Amaral (2013) ao realizarem um mapeamento sistemático dos trabalhos apresentado no Simpósio Brasileiro de Informática na Educação (SBIE), o Workshop de Informática na Escola (WIE) e o Workshop sobre Educação em Computação (WEI), demonstraram que o interesse no tema "Ensino de Computação na Educação Básica" está em ascensão - mesmo sendo bastante recente se comparado ao ensino de outras ciências como Matemática, Biologia, Física, entre outras.

4. Método e Operacionalização

O método foi qualitativo e de pesquisa exploratória, quando se estuda e aplica referenciais de ensino de Computação entre escola e profissionais de Licenciatura em Computação, acompanhando e analisando cada passo em sala de aula. Realizou-se um plano de trabalho para a orientação dos envolvidos nas atividades do Projeto na Escola de Aplicação Ivonita Alves Guerra.

Os estagiários, juntos aos professores orientadores, construíram um programa de componente curricular para cada turma do sexto ao oitavo ano, compondo a disciplina de Computação Educacional, após discussões sobre as regras da escola o programa foi elaborado com intuito de desenvolver o pensamento computacional dos educandos, contextualizando com a sociedade no qual está inserido, abrangendo três pilares: Pensamento Computacional (PC), Ecossistema Digital (ED) e Cultura Digital (CD).

Em seguida, a construção do programa iniciou um processo de estudo dos conteúdos, preparação de planos de aula, material de estudo para os educandos, atividades, avaliações, com cada passo acompanhado pelos professores no campo de estudo da escola de aplicação.

Na dimensão Ensino realizou-se atividades sobre: Introdução a Computação, Pensamento Computacional, História da Computação, Sistemas Operacionais, A evolução da Cultura Digital na humanidade, Representação de Algoritmos usando pseudocódigo, Introdução ao *HyperText Markup Language 5* (HTML5), Introdução ao *Cascading Style Sheets* (CSS 3) e Continuação ao HTML5, Princípios de Programação ao decorrer de nove aulas, abordando os assuntos descritos, das quais, seis trataram do Pensamento Computacional, por meio de assuntos relacionados, sendo as três primeiras deste pilar, trabalhadas com atividades desplugadas. Estas aulas com intuito de criar uma familiarização dos estudantes com a história e importância do uso do computador para a atual sociedade.

Como exemplo, a atividade desplugada "Cidade Enlameada", do livro Computer Science Unplugged, que mostra como os computadores são usados para encontrar as melhores soluções para os problemas da vida real, tais como conectar linhas elétricas entre casas, encontrando menor caminho que ligasse todos os pontos.

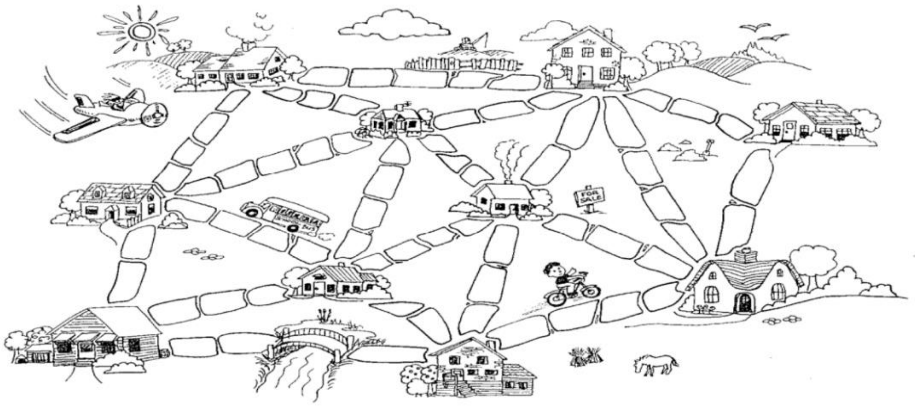


Figura 1. Planilha de Atividade: O problema da Cidade Enlameada.
Fonte: Computer Science Unplugged (1995)

As três demais aulas ocorreram no laboratório, onde as atividades desenvolvidas em conjunto ao computador, consistiram em conceitos de programação web.

Para a dimensão Extensão projetamos o desenvolvimento de um jogo com elementos básicos para que de maneira lúdica fosse ensinados os conceitos de lógica de programação, ao longo do projeto, onde o auxílio dos estagiários poderia ser solicitado, quando necessário para sanar dúvidas. Optamos por usar a ferramenta *Construct* 2, um programa que criar jogos de computador utilizando HTML5, de forma *offline*, possuindo uma versão gratuita. A atividade ocorreu em três aulas, dedicadas ao desenvolvimento dos jogos, sendo estas, divididas em quatro momentos, que podem ser observados na Figura 2.

Figura 2. Fluxograma apresentando as etapas da Extensão.

A imagem da Figura 2 demonstra as etapas da dimensão Extensão, tais etapas ocorreram nas três aulas previstas, sendo a terceira aula, utilizada para realizar as duas últimas etapas, Conclusão do Desenvolvimento do Projeto e Demonstração dos Resultados, respectivamente.

A Demonstração dos Resultados ocorreu de forma expositiva dialógica, onde os alunos puderam apresentar seus jogos para os demais colegas, discutindo sobre as ideias que basearam a criação.

Estes jogos criados, eram compostos por apenas uma fase, devido ao curto espaço de tempo que tínhamos disponível. Visando a criação do projeto, de forma atrativa, optamos por usar a metodologia *Problem-Based Learning* (PBL) durante as aulas da Extensão, tratando os alunos como sujeitos ativos, da seguinte maneira: após apresentação da ferramenta a ser utilizada e os conceitos iniciais de utilização, era proposto uma atividade relacionada ao jogo, a qual, ao ser realizada, contribuiria para a construção do projeto, os mesmos contavam com acesso à internet para realizar buscas por soluções, e podiam solicitar ajuda aos estagiários para solucionar dúvidas.

5. Resultados da experiência no ensino de Computação

O propósito do Estágio Supervisionado II foi criar um ambiente de ensino de Computação no ensino fundamental, o mais real possível, contribuindo para a formação de educandos e estagiários de Licenciatura em Computação, em decorrência da inexistência do ensino de Computação na Educação Básica do município de Garanhuns. Objetivou-se implementar uma vivência da computação no ensino fundamental, conforme as discussões anteriormente citadas.

Os tópicos a seguir descrevem as etapas que foram realizadas nessa experiência tão significativa nos âmbitos de Ensino e Extensão, desde o planejamento, preparação do material, até a finalização da execução e os resultados obtidos com este trabalho.

5.1 Ensino

Os estagiários receberam a informação que não teriam a supervisão e colaboração de um professor de Computação nas atividades em sala de aula, pois a instituição não possuía tais profissionais. Todavia os professores supervisores da disciplina de Estágio II, que acompanharam e contribuíram na construção do planejamento. Assim, nesta primeira etapa realizamos algumas reuniões técnicas e psicopedagógicas sobre as aplicações das atividades de estágio supervisionado. Após esta etapa, iniciamos a dimensão de Ensino

Através de reuniões realizadas antes do início da dimensão de Ensino, este grupo ficou responsável por ministrar aulas nas turmas do 7º (sétimo) ano do ensino fundamental. Ciente dessa informação e partindo da diagnose de que os educandos não tinham base de conhecimento sobre Computação e a abordagem dos eixos do Referencial, foi proposto um plano de trabalho entre a realidade e o proposto pela SBC para o ensino de Computação.

A escola não ofertava a disciplina Computação ou componente curricular semelhante, que construísse uma base sobre o assunto, sendo organizada uma sequência de aulas e atividades para inicialmente fazer com que os aprendentes descobrissem o que é a computação, assim incentivando a curiosidade na área, e subsequentemente os mesmos tivessem base para que o programa de atividade previsto fosse concluído.

No momento inicial do primeiro encontro realizou-se uma sondagem nos discentes para observar seus conhecimentos em Computação a fim de construir uma base necessária e confluir as aulas seguintes. Na segunda e terceira aulas contou com uma explanação do que é Computação e uma introdução ao PC e atividades como as do livro Computer Science Unplugged (1995) para melhor compreensão, nas três aulas seguintes, os planos de aula davam-enfoque em explanar a CD e ED.

Figura 3. Aula exploratória sobre Computação e Pensamento Computacional.

Enquanto CD abordou o que era um computador, a evolução dos computadores ao longo do tempo e os impactos dessa evolução na sociedade, como um todo. Quando o grupo pensou na execução dessa aula o foco era identificar o uso de tecnologia nas diferentes dimensões da vida escolar, social e profissional e riscos acarretados por vírus, tudo isso através de linhas do tempo da CD. A aula com foco no ED foi montada para que os educandos compreenderem as diferenças e a relação entre o software e hardware, logo, nessa aula o assunto explanado: Sistema operacional (S.O), camadas de S.O e o que é *Hardware e Software*.

É possível dividir a dimensão Ensino em 5 fases para a melhor compreensão do que foi realizado:

Tabela 1: Fases executadas na Dimensão Ensino.

Fase 1	Planejamento e organização dos assuntos para o ano escolar referente
Fase 2	Primeira aula: Sondagem do conhecimento que os alunos já tinham sobre os assuntos, explanação sobre esses conhecimentos, breve revisão.
Fase 3	Segunda e terceira aulas: Introdução ao PC e realização e explicação da atividade “O problema da Cidade Enlameada”, respectivamente.
Fase 4	Quarta e quinta aulas: Introdução a CD, riscos e impactos de CD para a sociedade.
Fase 5	Sexta aula: Introdução a ED e inicialização do assunto de *Hardware e Software*

5.2 Extensão

O eixo de Extensão teve início logo após as análises das atividades avaliativas realizadas no eixo de Ensino. Baseado nos resultados obtidos nos exercícios avaliativos e na observação que os estagiários viveram nas aulas anteriormente ministradas - os discentes demonstraram grande entusiasmo quando explanado como era construído um *software* – foi planejado um minicurso de desenvolvimento de conceitos iniciais e desenvolvimento

de jogos usando a ferramenta Construct 2, editor de jogos 2D baseado em HTML5. A escolha dessa ferramenta ao invés de uma linguagem de programação de alto nível como JavaScript ou frameworks, como o Allegro-biblioteca para jogos 2D feita em C, dá-se ao fato que estes itens citados rogam pela necessidade de explicações sobre conceitos que influenciariam a aprendizagem dos discentes, devido ao curto espaço de tempo disponível, tal prática se tornaria inviável.

Após um levantamento feito, o minicurso ocorreu no período de três encontros com a carga horária de duas horas cada, assim atendendo as necessidades dos alunos. Na primeira aula tratamos sobre os conceitos básico de planejamento e construção de jogos, desde os *sprites, sheet* até o que são motores gráficos. Já introduzido a ferramenta de criação e edição de jogos a ser utilizada no decorrer do minicurso o Construct 2.

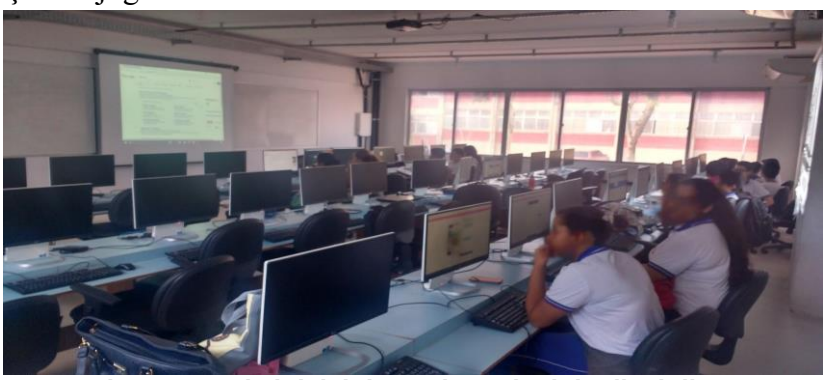


Figura 4. Aula inicial do Projeto Final da disciplina.

Na segunda aula os estagiários continuaram a apresentação da ferramenta usando uma abordagem prática. Ao decorrer de toda aula os discentes, com auxílio e supervisão dos estagiários, participaram da criação de um jogo de plataforma que abordou conceitos como: jogabilidade de plataforma, inserção de personagem, movimentação 2D, troca de fases e inserção de música. Toda a base inicial para a criação dessa categoria. A última aula foi dedicada para os participantes criassem seus próprios jogos usando o aprendizado da aula anterior e informações contidas na documentação do Construct 2, sendo este jogo a avaliação final do minicurso.

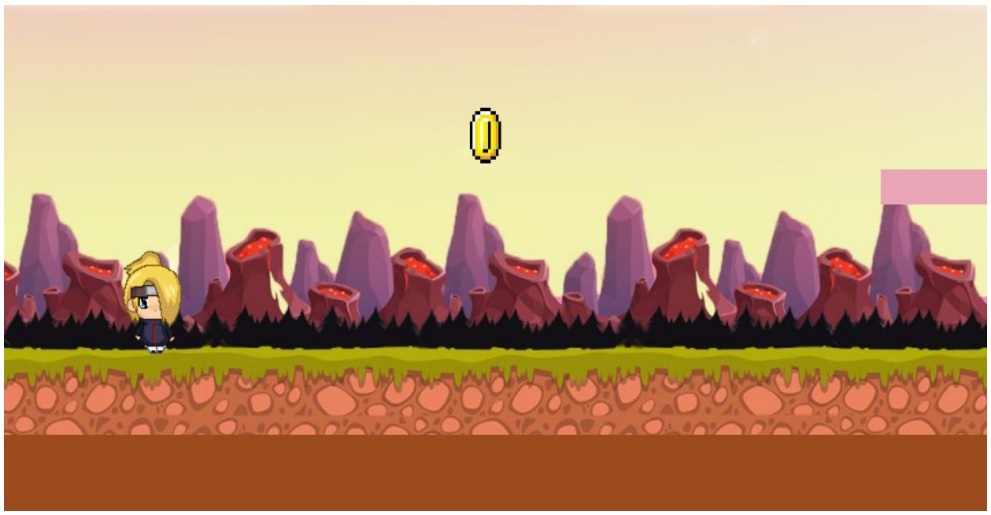


Figura 5. Projeto finalizado por um dos grupos de alunos.

A Figura 5 apresenta o resultado dos projetos de uma das equipes. Nela, é possível observar que mesmo em um curto espaço de tempo e dispondo de pouca experiência com esta ferramenta, os resultados obtidos foram surpreendentes, estes possibilitados pelo uso da ferramenta, junto ao site Game Art 2D[1] que possibilita o *download* de *sprites* gratuitos.

6. Considerações finais

A sociedade contemporânea demonstra uma demanda crescente pelo domínio dos conceitos de Computação, sejam eles no âmbito escolar ou no meio profissional, para suprir as mais variadas necessidades, derivadas de uma sociedade cada vez mais conectada e interativa que preza pela produtividade e domínio da capacidade de adaptação às novas tendências tecnológicas nos meios de produção.

É papel da escola fornecer a devida formação aos estudantes que fazem parte desta sociedade, que junto ao Licenciado em Computação, devem compor um sistema de ensino capaz de oferecer condições para que estes estudantes desenvolvessem as capacidades necessárias nos pilares que compõem à Computação Educacional.

Diante o que ocorreu no estágio, podem ser elencados pontos relevantes para esse relato, os resultados obtidos tanto no Ensino quanto na Extensão foram gratificantes, pois mesmo com a dificuldade dos alunos, por não possuir o arcabouço de conhecimento necessário para o que foi estabelecido naquele nível escolar trabalhado, os mesmo, conseguiram realizar as atividades previstas, sem estagnar em determinado ponto do cronograma, mostrando que o aprendizado ocorreu de forma intuitiva ao decorrer das aulas, e das pesquisas realizadas, estimulando-os a construção das capacidades de raciocínio, pesquisa, e solução de problemas.

Considerando que no Brasil o ensino de Computação não faz parte da BNCC e que as discussões avancem lentamente, concluímos que trabalhos como este, que relatam os excelentes resultados que projetos experimentais exercem, proporcionando aos alunos envolvidos o desenvolvimento do pensamento computacional e conceitos diretamente e indiretamente relacionados, são de profunda importância para o desenvolvimento de discussões que propiciem a implementação deste componente curricular na BNCC.

Referências

Allegro. Disponível em: <http://desenvolvimentodejogos.wikidot.com/allegro>. Acesso em 12 de maio de 2018.

Bell T.C.G., Witten, I. (1995).“Computer Science Unplugged: Capturing the interest of the uninterested”. Anais do NZ Computer Conference,Wellington, Nova Zelândia.

BRASIL. Constituição (1988). Constituição da República Federativa do Brasil. Brasília, DF: Senado Federal, 1988. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11788.html> Acesso em: 09/12/2017.

Construct 2 Disponível em: <https://www.scirra.com/>. Acesso em 10 de dezembro de 2017.

CSTA - Computer Science Teacher Association. (2005) “The New Educational Imperative: Improving High School Computer Science Education”. Final Report

[1] www.gameart2d.com

of the CSTA. Curriculum Improvement Task Force. ACM - Association for Computing Machinery.

CSTA - Computer Science Teacher Association. (2011) "CSTA K-12 Computer Science Standards". CSTA Standards Task Force. ACM - Association for Computing Machinery.

Freitas (2016). "Alunos e Escolas para a Sociedade 4.0". Revista Linha Direta, 15 de Julho.

França R.S., Ferreira V.A.F., Almeida L.C.F., Amaral H.J.C. (2014). "A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação" XXXIV Congresso da Sociedade Brasileira de Computação – CSBC.

França R.S. Amaral H.J.C. (2013). "Ensino de Computação na Educação Básica no Brasil: Um Mapeamento Sistemático" Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2013/009.pdf>. Acesso em 21 março 2017.

JavaScript. Distonivel em : < http://tableless.github.io/iniciantes/manual/js/ >. Acesso em 12 de maio de 2018.

Kulcsar, Rosa (2012). O estágio supervisionado como atividade integradora. In: Fazenda, Ivani Catarina Arantes [etall]; Piconez, Stela C. Bertholo (Coord.). A prática de ensino e o estágio supervisionado. Campinas-SP: Papirus.

Nunes, D. J. (2011). "Ciência da Computação na Educação Básica". Jornal da Ciência, 09 de Setembro.

Oliveira, E.S.G.; CUNHA, V.L. O estágio Supervisionado na formação continuada docente à distância: desafios a vencer e Construção de novas subjetividades. 2006. In: BERNARDY, Katieli1; PAZ, Dirce Maria Teixeira. Importância do estágio supervisionado para a formação de professores. 2012. Disponível em: Acesso em 09/12/2017.

Pimenta, S. G. O estágio na formação de professores: unidade teórica e prática? 4. ed. São Paulo: Cortez, 2001.

As Aventuras de Ada e Turing: Apoiando o Desenvolvimento de Habilidades do Pensamento Computacional por Meio de um Jogo

Géssica Neves Sodré da Silva[1], **Letícia Lopes Leite**[1], **Ana Carolina Lopes de Jesus**[1]

[1]Departamento de Ciência da Computação
Universidade de Brasília(UnB) - Brasília, DF - Brasil

gnsodre,lopesleiteleticia,carolinalj94@gmail.com

Abstract. *Several countries are investing in educational policies to insert content that promotes the development of Computational Thinking (CP) in the early school years. Considering that in Brazil we have few of these initiatives directed to Elementary School I, it is proposed with this work to support the development of CP skills in these students. From the research question: "How can Computational Thinking skills be applied in the development of a game for Elementary School I?", the game "**The Adventures of Ada and Turing**" was developed, it seeks to favor the development of CP skills in Elementary Students, and contribute to researchs related to this topic.*

Resumo. *Diversos países estão investindo em políticas educacionais para inserir conteúdos que promovam o desenvolvimento do Pensamento Computacional (PC) nos primeiros anos escolares. Tendo em vista que no Brasil temos poucas destas iniciativas direcionadas ao Ensino Fundamental I, propõe-se com este trabalho apoiar o desenvolvimento de habilidades do PC nestes alunos. A partir da questão de pesquisa: "Como as habilidades do Pensamento Computacional podem ser aplicadas no desenvolvimento de um jogo para o Ensino Fundamental I?", foi desenvolvido o jogo "**As Aventuras de Ada e Turing**", que busca favorecer o desenvolvimento das habilidades do PC em estudantes do Ensino Fundamental I, além de contribuir com pesquisas relacionadas a este tema.*

1. Introdução

Apesar da inegável e crescente disseminação dos computadores nos últimos anos, um estudo realizado entre 2011 e 2015 pela *Organisation for Economic Cooperation and Development* [OECD 2016] em 33 países, com indivíduos com idades entre 16 e 65 anos de idade, mostrou que cerca de 26% da população adulta não sabe utilizar computadores e, que apenas 5% é capaz de conceber soluções consideradas mais complexas e que exigem o uso tanto de tecnologias computacionais genéricas quanto mais específicas. Isso mostra que a maioria dos usuários de computadores os utilizam de forma superficial e poucos sabem aplicá-los em tarefas mais complexas, o que pode se tornar um problema no futuro, devido ao rápido crescimento da área de tecnologia. Essa expansão pode ser evidenciada por um dos relatórios da [Burning Glass Technologies 2016], empresa especializada em estudos analíticos voltados ao mercado de trabalho, que afirma que empregos que exigem conhecimentos em programação estão crescendo 12% mais rápido que a média geral do mercado para os demais tipos de ocupações.

No contexto dos avanços tecnológicos em que vivemos, surgiu o tema Pensamento Computacional (PC), termo que ganhou importância ao ser apresentado por Jeannette [Wing 2006]. O PC representa um conjunto de habilidades que nos possibilita dividir problemas complexos em subproblemas menores por meio de uma sequência de passos, revisar como uma solução pode ser aplicada em problemas similares e, por fim, determinar se um computador pode ajudar na resolução do problema.

Dados os benefícios oferecidos pelos conhecimentos computacionais à inserção social de um indivíduo, tais como vantagem competitiva no mercado de trabalho e aquisição de habilidade de lidar com problemas complexos - entendendo o que é o problema e desenvolvendo soluções para ele, de maneira computacional ou não - que podem ser utilizadas nos mais diversos campos de atuação, o foco deste trabalho será no Pensamento Computacional. Busca-se, mais especificamente, uma forma de apoiar o desenvolvimento do PC nas séries finais do Ensino Fundamental I, no Brasil.

A metodologia utilizada envolve pesquisa exploratória para estudo das abordagens relacionadas ao desenvolvimento de habilidades do PC na educação básica, a implementação de um jogo digital que desenvolva um subconjunto dessas habilidades e sua avaliação. Esta última etapa, a avaliação, foi realizada com base no *framework* Octalysis e em testes de usabilidade com o público-alvo.

2. Pensamento Computacional

A Ciência da Computação é utilizada de forma transversal em todas as áreas do conhecimento. Seu caráter interdisciplinar auxilia na evolução das mais diversas atividades humanas, como a construção de edifícios, a análise de DNA e a criação de medicamentos. Em 2008, Blikstein destacava a época de transição em que vivíamos, apontando como exemplo cientistas que passavam a maior parte do tempo construindo modelos computacionais ao invés de perpetuarem o clássico estereótipo do cientista do século XX, que passava horas em um laboratório com tubos de ensaio, e os engenheiros, que também utilizavam modelos computacionais além de papel e lápis ao projetarem linhas de produção. Além disso, o autor também afirma que precisamos de mais pessoas que não sejam apenas usuários de tecnologia, mas também criadores de soluções [Blikstein 2008].

Neste contexto, situa-se o PC, termo que vem ganhando destaque pelo mundo desde a década passada e que gerou diversas discussões quanto à sua inclusão no currículo escolar básico de forma direta ou transversal às demais disciplinas. Seu o objetivo é preparar indivíduos capazes de analisar problemas e projetar suas soluções, sejam computacionais ou não.

O termo Pensamento Computacional em si passou a ser discutido a partir de 2006, no artigo *Computational Thinking*, de Jeannette [Wing 2006]. Nele a autora afirma que o "Pensamento Computacional é construído com base no poder e nos limites dos processos de Computação, sejam eles executados por humanos ou por máquinas". Apesar da disseminação do PC, não houve uma definição concreta do termo, sendo a maior contribuição de Wing a respeito do tema a distinção entre PC, programação e outros tipos de pensamentos analíticos, além de destacar a importância dele para qualquer pessoa em todos os campos de atuação. Em 2008, Wing voltou a fazer contribuições acerca do assunto, prenunciando o PC como um instrumento para a descoberta e a inovação em várias áreas, como parte integral da educação infantil e, consequentemente, do ensino do PC

para crianças [Wing 2008].

Influenciados por Wing, diversos autores deram suas próprias definições de PC. Dentre eles, Yadav, Hong e Stephenson afirmam que o PC envolve: dividir problemas complexos em subproblemas mais simples (decomposição de problemas) por meio de uma sequência de passos (algoritmos), revisar como a solução pode ser aplicada em problemas similares (abstração) e, por fim, determinar se um computador pode ajudar na resolução do problema [Yadav et al. 2016]. Paulo Blikstein, por sua vez, destaca que o PC não é saber utilizar o computador em tarefas do dia a dia, como enviar e-mails e navegar na Internet, mas sim usufruir do computador como "um instrumento de aumento do poder cognitivo e operacional humano" para aumentar a criatividade e a produtividade [Blikstein 2008].

2.1. Pensamento Computacional na Educação Básica

Conforme apresentado anteriormente, dada a importância da Computação e dos seus desdobramentos como ferramenta indispensável ao avanço da sociedade, é necessário preparar indivíduos capazes de criar soluções. [Wing 2006] foi a primeira autora a sugerir a necessidade de que as bases do Pensamento Computacional (PC) tornem-se comuns a todas as áreas, assim como outros tipos de pensamento, como o matemático e o científico, que são desenvolvidos durante toda a vida acadêmica. Outrossim, diversos autores vêm defendendo a introdução do PC como uma disciplina obrigatória da Educação Básica. [Barr and Stephenson 2011] defendem que não é mais suficiente esperar que os estudantes cheguem até o ensino superior para serem expostos aos conceitos do PC.

No contexto escolar, em 2014, Judith Gal-Ezer e Chris Stephenson contaram a história de sucesso e os desafios enfrentados no desenvolvimento do currículo/padrões do ensino de Ciência da Computação em Israel e nos Estados Unidos, destacando o PC como uma habilidade da Ciência da Computação que os estudantes devem desenvolver e, para tanto, recomendam a sua inclusão como conteúdo curricular [Gal-Ezer and Stephenson 2014]. Além de Israel e dos Estados Unidos, outros países também vêm se empenhando na expansão do PC, como a Nova Zelândia, que irá incorporá-lo ao seu currículo como disciplina essencial a todas as crianças nos 10 primeiros anos escolares [Parsons 2016] e o Reino Unido, que sofreu uma alteração curricular em 2014, na qual propôs a inserção do PC no ensino secundário [CAS 2014].

No Brasil, assim como nos Estados Unidos, também vêm ocorrendo esforços conduzidos pela academia com o objetivo de disseminar o PC. Em documento gerado pela SBC, aprovado pela Comissão de Educação e apresentado no CSBC 2017 durante as Assembleias do WEI e da SBC, o PC é descrito como um dos três eixos dos conhecimentos da área de Computação, juntamente com o Mundo Digital e a Cultura Digital [Raabe et al. 2017]. Ainda nesse documento, os autores defendem a inserção da Computação no currículo da Educação Básica como forma de oportunizar a formação de habilidades e competências. Nesse sentido, o PC se destaca como potencializador da capacidade de solucionar problemas para criar processos e produtos por desenvolver competências nomeadas como: abstração, automação e análise.

Ainda assim, no Brasil, os projetos normalmente utilizam materiais baratos como papelão e garrafas em atividades lúdicas no intuito de promover habilidades do PC [Pinho et al. 2016]. Embora o ideal fosse utilizar os recursos tecnológicos em tais ativi-

dades a fim de que os alunos vivenciassem as tecnologias que irão encontrar fora da sala de aula, é preciso observar que a realidade brasileira torna inviável que isso ocorra em todas as escolas. Uma pesquisa da [Fundação Victor Civita 2009] aponta que "questões de infraestrutura, como o número reduzido de computadores e a falta de um laboratório de informática, são vistos como um dos principais problemas no uso pedagógico", além do despreparo dos professores para lidar com a tecnologia.

No contexto da Educação Básica, deve-se observar que o desenvolvimento de um currículo que inclua o PC em sua base, ou mesmo o uso de forma independente em instituições que não o tenham em seu currículo formal, exige a distinção das habilidades trabalhadas por ele.

2.2. Habilidades do Pensamento Computacional

Com a popularização do Pensamento Computacional e o reconhecimento da comunidade acadêmica quanto à sua importância, existem diversos esforços com o objetivo de conceber uma lista que abarque as suas habilidades básicas. Tendo em vista a criação de uma estrutura e de vocabulário de apoio que tornasse os conceitos do PC acessíveis aos educadores, a [ISTE/CSTA 2011] propôs uma definição operacional do PC. Esta definição considera um conjunto composto por seis habilidades, tais como:

1. elaborar problemas de forma que seja possível utilizar um computador e outras ferramentas para resolvê-los;
2. organizar e analisar dados logicamente;
3. representar dados por meio de abstrações, como modelos e simulações;
4. automatizar soluções por meio de pensamento algorítmico;
5. identificar, analisar e implementar possíveis soluções com o intuito de atingir a combinação mais eficiente e efetiva de passos e recursos;
6. generalizar e transferir esse processo de resolução de problemas para outros problemas.

Formuladas durante um *workshop* promovido também pela CSTA, [Barr and Stephenson 2011] apresentaram listas resultantes de discussões dos conceitos centrais do Pensamento Computacional no contexto de habilidades, disposições e pré-disposições, e cultura de sala de aula. O resultado das discussões acerca das habilidades incluem:

1. projetar soluções para problemas utilizando abstração, automação, criação de algoritmos, coleta de dados e análise;
2. implementar modelos;
3. realizar testes e depuração;
4. modelar, executar simulações e analisar sistemas;
5. refletir sobre prática e comunicação;
6. usar o vocabulário;
7. reconhecer abstrações e ser capaz de trabalhar com diferentes níveis de abstração;
8. inovar, explorar e exercer a criatividade interdisciplinarmente;
9. resolver problemas em grupo;
10. empregar diferentes estratégias de aprendizagem.

No âmbito deste trabalho, o conjunto de habilidades apresentados por [Barr and Stephenson 2011] serão utilizados no desenvolvimento de um jogo digital direcionado a crianças cursando o Ensino Fundamental I (a partir dos 7 anos). O uso de habilidades discutidas em um contexto de profissionais da área de educação justifica-se pela importância de uma base teórica adequada que possa guiar o desenvolvimento de um jogo educativo. Ademais, a escolha da lista que possui um número maior de habilidades se deve ao fato de que ela inclui a comunicação e o trabalho em grupo, habilidades muito valorizadas no mercado de trabalho atual e que poderão incentivar o engajamento dos alunos durante as atividades do jogo.

3. Jogos Digitais na Educação

O advento e a popularização dos jogos digitais em diversas faixas etárias, e principalmente entre os mais jovens, proporcionou sua utilização como ferramenta potencializadora de novas e enriquecedoras oportunidades de aprendizagem. Hoje, temos exemplos de algumas abordagens para trazer os jogos para a sala de aula, principalmente como instrumento que engaje o aluno e aumente sua motivação para aprender.

Para além do uso como ferramenta, tanto em sala de aula quanto em outros contextos educativos, os fundamentos dos jogos também surgem como artefatos para a construção de novas formas de motivar estudantes e aprimorar suas habilidades. Neste sentido, o conceito de gamificação foi trazido para o âmbito educacional. Segundo Chou, a gamificação consiste na utilização dos elementos divertidos e atrativos encontrados nos jogos, aplicando-os ao mundo real para atividades produtivas [Chou 2017]. Desta forma, ao invés do foco estar somente na eficiência do sistema, a gamificação é um processo de *design* que enfatiza o humano no sistema, o que Chou denomina como "Design Focado no Humano". A partir dessa análise, Chou empregou esses elementos para criar o Octalysis um *framework* que auxilia na identificação do quão gamificado um produto ou processo está.

Segundo Ramos, Lorenset e Petri, as características e elementos presentes nos jogos, como regras e restrições, narrativa, objetivos, interação, desafio, competição e conflito, resultados, recompensas e *feedback* contribuem à aprendizagem [Ramos et al. 2016]. Tais aspectos podem ser avaliados através do *framework* Octalysis, que identifica oito direcionadores principais da gamificação apontados por Yu-kai [Chou 2015], formando uma estrutura de octógono (Figura 1).

No desenvolvimento do *framework* Octalysis, Yu-kai Chou [Chou 2015] indica que, em relação à diversão em um jogo, existem oito direcionadores principais que motivam a realização de atividades. Esses direcionadores são a base do octógono que forma o *framework* Octalysis:

1. Significado Épico & Chamado: o motivador é a crença de que se está fazendo algo maior ou que foi o "escolhido" para fazer algo.
2. Desenvolvimento & Realização: o motivador é o desenvolvimento de habilidades e, eventualmente, superação de desafios.
3. Empoderamento da Criatividade & *Feedback*: o motivador é o envolvimento em um processo criativo onde os usuários devem descobrir coisas repetidamente e tentar diferentes combinações.
4. Propriedade & Posse: o motivador é o sentimento de que se possui algo.

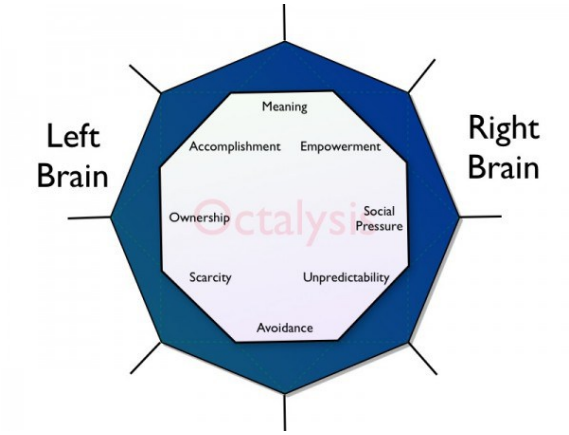


Figura 1. Direcionadores do Octalysis

5. Influência Social & Relacionamento: os motivadores são elementos sociais que direcionam as pessoas, incluindo mentoria, aceitação, respostas sociais, companhia, competição e inveja.
6. Escassez & Impaciência: o motivador é o desejo por algo que não se pode ter.
7. Imprevisibilidade & Curiosidade: o motivador é o desejo de descobrir o que vai acontecer em seguida.
8. Perda & Evasão: o motivador é a busca por evitar que algo negativo aconteça.

Segundo [Chou 2015], além dos oito direcionadores descritos, o octógono gerado pelo *framework* Octalysis também pode ser dividido em lado direito e esquerdo, e gamificadores superiores e inferiores. Os direcionadores do lado direito estariam relacionados à criatividade, autoexpressão e aspectos sociais, e o lado esquerdo mais associado à lógica, cálculos e pertencimento. Do mesmo modo, os direcionadores superiores são considerados motivadores muito positivos, enquanto os inferiores são classificados como negativos.

Os direcionadores descritos são a base da Octalysis Tool e, a partir dela, é possível avaliar se um produto ou processo apresenta características gamificadas. Cada um dos direcionadores é pontuado com um valor entre 0 e 10, baseado em julgamento pessoal, dados e experiência de uso. Quando toda pontuação for adicionada, é gerada a pontuação final do Octalysis. Essa avaliação é importante para apontar o que está faltando no jogo e o que poderia ser trabalhado para melhorar a experiência do jogador e alcançar o objetivo pretendido com o produto ou projeto.

3.1. O Jogo Digital como Recurso para o Desenvolvimento do Pensamento Computacional

Na literatura referente ao Pensamento Computacional, é possível encontrar diversos exemplos de softwares educativos que utilizam abordagens gamificadas e que podem auxiliar no desenvolvimento do PC. Dentre esses exemplos, estão os ambientes gráficos de programação Scratch, Alice, Game Maker e Kodu, além de ferramentas de simulação e criação como o AgentSheets e AgentCubes [Grover and Pea 2013].

Em 2016, Zanetti, Borges e Ricarte fizeram uma revisão de trabalhos relacionados ao PC, publicados em eventos brasileiros entre os anos de 2012 a 2015

[Zanetti et al. 2016]. Todavia, dos 16 artigos adequados ao contexto da pesquisa apenas seis trabalham o tema utilizando jogos digitais, dentre os quais apenas um tinha como foco o Ensino Fundamental. Ainda, todos os jogos digitais avaliados nestes artigos trabalham o PC no contexto de programação, característica que se repete em outros trabalhos.

Partindo do contexto do apoio ao ensino e à aprendizagem infantil, iniciativas como o Play Code Dog [Santos et al. 2017] e o Fantastic Pirates [Abreu et al. 2017] destacam não só a relevância de se trabalhar temas como lógica de programação e tecnologias digitais, como celulares e tablets, aplicadas ao processo de ensino e aprendizagem, mas também evidenciam a importância de se observar as necessidades educacionais específicas de alunos do Ensino Fundamental. Contudo, esses trabalhos não têm como objeto o desenvolvimento das habilidades do PC.

Em trabalhos mais recentes, [Zanetti et al. 2017] usam a programação como ferramenta para explorar o conteúdo de Lógica de Programação e de PC, fazendo uso do Scratch com o objetivo desenvolver uma abordagem motivadora e criativa para o nível fundamental. Outra pesquisa, proposta por [Pessoa et al. 2017], utiliza as habilidades do PC de forma mais definida e emprega uma abordagem gamificada em um aplicativo que propõe uma série de desafios ao usuário, o T-mind, no entanto é direcionado ao Ensino Médio.

Diante da escassez de trabalhos direcionados às séries iniciais do Ensino Fundamental onde as habilidades do PC fossem trabalhadas de forma sistemática, trazemos uma abordagem gamificada diferente para trabalhar o PC, focando nas habilidades que podem ser desenvolvidas transversalmente no currículo do Ensino Fundamental I, denominada "**As Aventuras de Ada e Turing**".

4. As Aventuras de Ada e Turing

Ainda que poucas, as pesquisas sobre o uso de jogos digitais apontam sua efetividade como apoio educacional para as crianças e adolescentes (Seção 3). Muito além da eficácia do jogo como ferramenta educacional, é preciso também observar a necessidade da inclusão da tecnologia na educação numa sociedade como a atual, onde o progresso tecnológico é constante e está presente no dia a dia da população.

No contexto das discussões acerca da inclusão do desenvolvimento do PC na Educação, em especial no Ensino Fundamental I, este trabalho apresenta uma proposta de jogo digital que está alinhado a este cenário. Para tanto, cada uma das fases do jogo tem como objetivo expor o jogador a uma situação que promova o desenvolvimento de algumas das habilidades do PC citadas por [Barr and Stephenson 2011].

Dentre os desafios de se propor uma solução para o desenvolvimento do PC em ambiente escolar, a sua não inclusão no currículo da Educação Básica figura entre os maiores. Para contornar esse problema, nos baseamos no livro "*Computer Science Unplugged*: Ensinando Ciência da Computação sem o uso do computador" [Bell et al. 2011]. Nele, os autores propõem uma coletânea de atividades "desplugadas" para que qualquer pessoa aprenda conceitos da Ciência da Computação através de jogos e enigmas de vários tipos.

Nesse cenário, nos embasamos para o desenvolvimento do jogo, na atividade 12, intitulada "Seguindo Instruções - Linguagens de Programação" [Bell et al. 2011, p.101].

Nela, os autores apresentam o conceito “linguagem”, da Ciência da Computação, como um “vocabulário limitado de instruções que devem ser obedecidas” pelo computador. Essa habilidade, explicada pelos referidos autores como "dar e receber instruções", pode ser identificada como “usar o vocabulário”, presente na lista de habilidades do PC. A partir disso, é proposta às crianças uma experiência sobre esse conceito da programação.

A atividade 12 foi escolhida por manifestar dois importantes aspectos relacionados ao desenvolvimento de um jogo digital para crianças no Ensino Fundamental I:

1. ensinar conceitos de programação sem utilizar a programação em si;
2. ter como público-alvo crianças a partir dos 7 anos de idade.

A partir do conceito de dar e seguir instruções apontado pela atividade 12, foi necessário identificar as habilidades que queríamos desenvolver: usar o vocabulário; projetar soluções para problemas utilizando análise; projetar soluções para problemas utilizando análise e criação de algoritmo; e realizar testes e depuração [Barr and Stephenson 2011]. Buscamos relacionar em maiores detalhes cada uma destas habilidades às fases específicas do protótipo desenvolvido.

Tendo como base os fundamentos supracitados e relacionados ao Pensamento Computacional e, tendo como inspiração dois personagens importantes da história da Computação (Ada Lovelace e Alan Turing), criamos o protótipo do jogo, chamado “**As Aventuras de Ada e Turing**”. Buscamos, através dele, mostrar que é possível trabalhar algumas das habilidades do PC, utilizando o apelo lúdico do jogo, sem necessariamente exigir do aluno conhecimentos de programação.

4.1. Implementação

O jogo “**As Aventuras de Ada e Turing**” foi implementado utilizando o *framework* Corona SDK. Esta escolha deve-se ao fato dele fornecer soluções para a maioria dos obstáculos para o desenvolvimento de um software educativo, entre eles: desenvolvimento rápido, gratuidade e multiplataforma. Os cenários do jogo foram criados utilizando o editor de mapas Tiled[1], pois ele possibilitou criar o cenário, importando-o posteriormente para o Corona SDK.

O código e versão final do protótipo do jogo “**As Aventuras de Ada e Turing**” pode ser acessado pela plataforma GitHub por meio do link: `https://github.com/gessikete/As-Aventuras-de-Ada-e-Turing/releases/tag/v1.0`.

4.2. Jogabilidade

O enredo foi pensado para contemplar parte do dia a dia de estudantes a partir dos 7 anos de idade, passando pelas fases denominadas: **Casa**, **Escola** e **Restaurante**.

Hoje é mais um dia normal na vida de Ada, porém sua mãe a presenteia com uma bicicleta e um desafio: percorrer a cidade ajudando algumas pessoas e voltar para **Casa** no menor tempo possível para ganhar uma surpresa. Entretanto, seu irmão Turing ouviu a proposta da mãe e resolveu criar uma competição. Será que Ada conseguirá chegar em **Casa** antes de Turing? No caminho, ela terá que ajudar o professor a organizar a bagunça deixada pelos alunos na **Escola** e também ajudar o cozinheiro a preparar uma receita bem gostosa. Isso tudo usando o que as pessoas a ensinam para chegar sempre na frente do irmão.

[1]Disponível para *download* no site `http://www.mapeditor.org/`

O jogo é composto por quatro cenas: a **Casa** do personagem principal, a **Escola**, o **Restaurante** e o **Mapa da Cidade**, de forma que este último agrega todos os locais anteriores. O **Mapa da Cidade** (Figura 2) representa uma cena de transição e funciona como tela de controle do progresso do jogador.

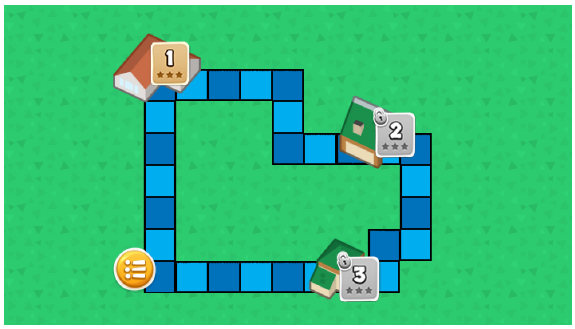


Figura 2. **Mapa da Cidade**

Cada elemento do jogo "**As Aventuras de Ada e Turing**" foi planejado com o intuito de estimular o jogador e desenvolver habilidades do PC sem perder o aspecto lúdico de um jogo. O protótipo do jogo envolve três fases, de forma que quatro das dez habilidades listadas por [Barr and Stephenson 2011] foram utilizadas:

1. Usar o vocabulário: com o intuito de alinhar o jogo com o objetivo da atividade 12, "vocabulário limitado de instruções que devem ser obedecidas".
2. Projetar soluções para problemas utilizando análise: com o intuito de incentivar o jogador a analisar um problema antes de projetar uma solução.
3. Projetar soluções para problemas utilizando análise e criação de algoritmo: com o intuito de, junto à análise, criar uma sequência lógica de passos para solucionar o problema.
4. Realizar testes e depuração: com o intuito de encorajar o jogador a aprender com os erros e continuar tentando para melhorar seu desempenho.

4.2.1. Fase 1: Casa

O objetivo desta fase é desenvolver a habilidade de usar o vocabulário. Para tanto, o jogador passará por um treino interativo (Figura 3), expondo as instruções de direção e o conceito de repetição representado por uma bicicleta, de forma que a quantidade de pedaladas é visualizada quando a criança gira a roda o número de vezes correspondente à quantidade de passos a serem dados no tabuleiro. Ainda neste cenário a mãe irá instruir o personagem sobre o funcionamento dos comandos do jogo, enquanto uma animação destaca os elementos que devem ser utilizados.

4.2.2. Fase 2: Escola

A fase 2 se passa na **Escola** (Figura 4) e continua utilizando os conceitos demonstrados na fase 1, porém limita as ações do jogador.

Na fase 1, o jogador pode cumprir o objetivo em quantas etapas desejar. No entanto, nesta fase, ele deverá completar o objetivo utilizando um único conjunto de instruções, porém na ordem

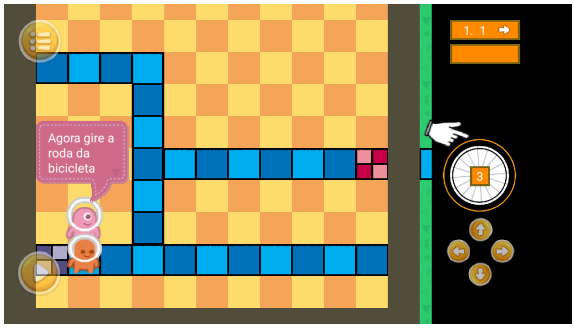


Figura 3. Interface que representa a fase 1 do jogo "**As Aventuras de Ada e Turing**"

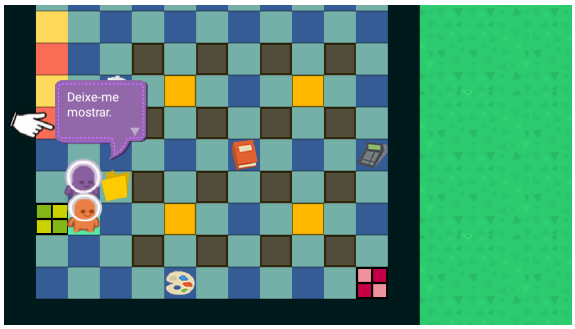


Figura 4. Interface que representa a fase 2 com os objetos espalhados pela sala de aula

mais adequada, ou seja, elaborar a melhor estratégia para pegar todos os materiais espalhados pela sala e depositá-los no organizador utilizando um único conjunto de instruções. Assim, pretendemos que o jogador projete soluções para problemas utilizando análise, mais uma habilidade do PC.

4.2.3. Fase 3: Restaurante

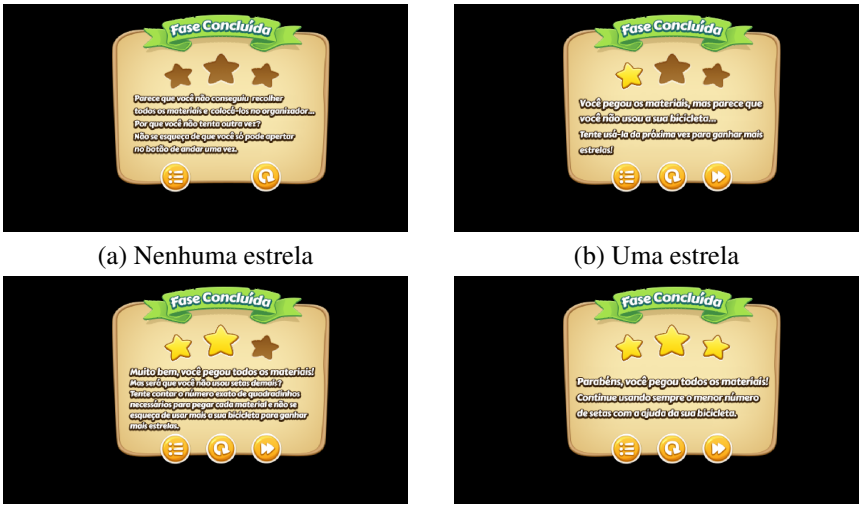
A fase 3 foi desenvolvida para que o jogador projete soluções para problemas utilizando análise e criação de algoritmo. Com o intuito de desenvolver essa habilidade do PC, o jogo propõe que o jogador complete uma receita em duas etapas. Para tanto, ele deve coletar os ingredientes espalhados pelo **Restaurante** de forma ordenada - obedecendo uma lista - e retornar cada uma das vezes com um conjunto de ingredientes ao cozinheiro (Figura 5).

Figura 5. Interface que representa a fase 3 com os objetos espalhados pelo **Restaurante**

Novamente, o jogador estará utilizando todos os conceitos que aprendeu anteriormente, porém deverá seguir a ordem definida pelo cozinheiro. No **Restaurante** (Figura 5) é possível observar os ingredientes espalhados e a ordem em que a primeira lista deve ser entregue ao cozinheiro.

O jogo "**As Aventuras de Ada e Turing**" chega ao fim quando o jogador retorna para **Casa** após ajudar todas as pessoas da cidade. Nesse momento, caso tenha conseguido chegar antes do irmão em pelo menos duas fases, o jogador receberá a surpresa (premiação final) que foi prometida pela mãe. Caso contrário, perderá a aposta com o irmão e ficará sem a surpresa. No entanto, o jogo permite que o jogador refaça todas as fases para tentar terminar o jogo recebendo a premiação final.

Ao finalizar cada uma das fases, o jogo apresentará quatro possíveis *feedbacks* (Figura 6):



(a) Nenhuma estrela (b) Uma estrela

(c) Duas estrelas (d) Três estrelas

Figura 6. Telas de *feedback*

Cada nível de *feedback* pretende informar o usuário o seu desempenho na fase jogada:

1. Nenhuma estrela (Figura 6a): o jogador não alcançou o objetivo da fase.
2. Uma estrela (Figura 6b): o jogador completou o objetivo, porém não utilizou a habilidade do PC desenvolvida na fase.
3. Duas estrelas (Figura 6c): o jogador completou o objetivo, porém não utilizou a habilidade do PC desenvolvida na fase tanto quanto poderia.
4. Três estrelas (Figura 6d): o jogador completou o objetivo da fase e utilizou a habilidade do PC desenvolvida na fase de forma adequada.

É preciso que os jogadores sejam capazes de aprender com os próprios erros, revendo e analisando seus passos, para que seja possível traçar uma nova estratégia em caso de fracasso, ou simplesmente para compreender por que obteve sucesso [Paula and Valente 2016]. Desse modo, mais uma habilidade do PC é incluída ao jogo: realizar testes e depuração, pois o jogador pode observar que poderia ter se saído melhor naquela fase e de que forma poderia melhorar, tendo a

opção de jogar a mesma fase novamente para tentar obter um desempenho melhor. Ainda trabalhando a habilidade de realizar testes e depuração, a movimentação do jogador é destacada à medida que o personagem se desloca criando um rastro verde. Dessa forma é possível que ele reveja os pontos onde acertou ou errou.

5. Avaliação e Testes

O trabalho desenvolvido foi submetido à avaliação por especialistas em desenvolvimento de jogos e pelas autoras do trabalho utilizando o *framework* Octalysis. Ainda, foi realizada a avaliação da jogabilidade utilizando um conjunto heurísticas proposto por [Barcelos et al. 2011]; testes de usabilidade realizados diretamente com o público-alvo do jogo; e a aplicação de questionário pós-teste feito com este público após utilizarem o aplicativo. Salienta-se, neste caso, que os responsáveis pelos avaliadores assinaram o Termo de Consentimento Livre e Esclarecido, pois tratam-se de crianças com idades entre 5 e 8 anos.

A avaliação a partir do Octalysis indicou que os direcionadores do jogo são Realização e Evasão e apontou a carência de aspectos sociais no jogo, fato que fez com que uma das extremidades ultrapassasse a parte interna do octógono, apontando um aspecto fraco e que precisa ser trabalhado no aplicativo (Figura 7). Esta avaliação ainda apontou que há uma tendência pela motivação mais extrínseca que, conforme [Chou 2015] tende a ser mais efetiva em um jogo.

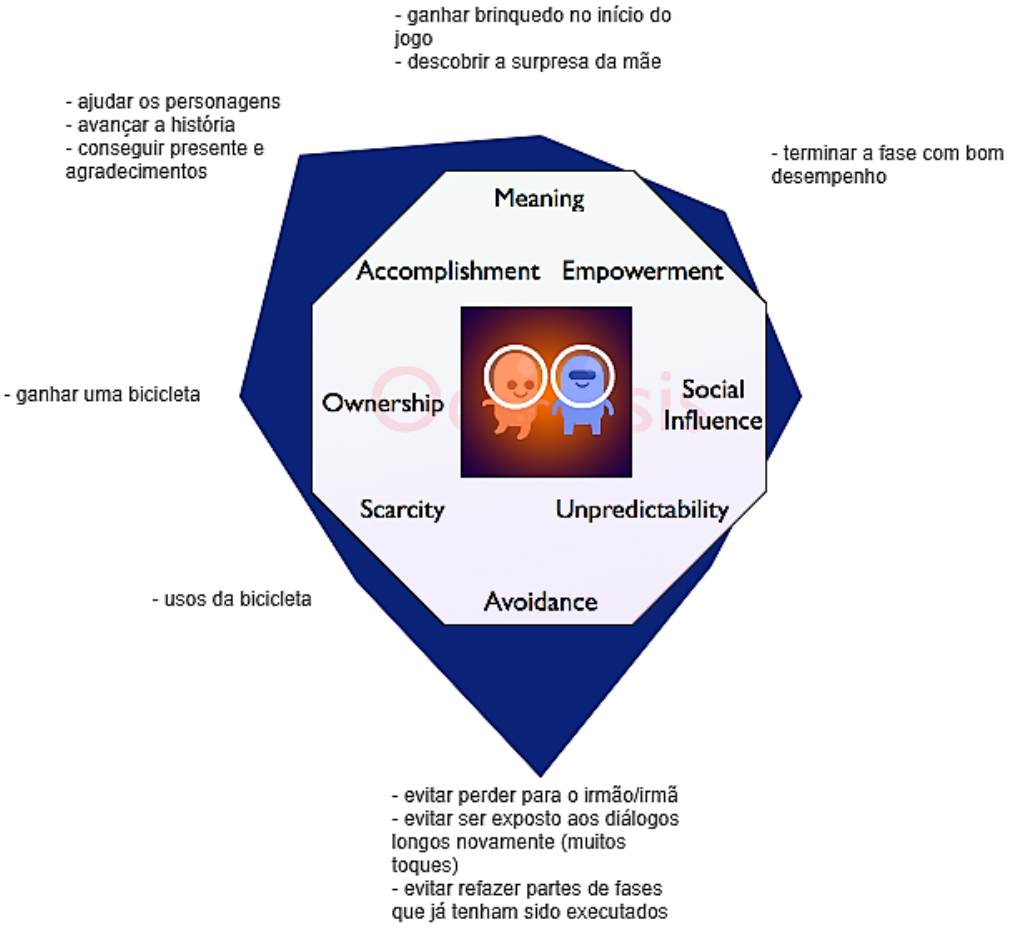


Figura 7. Avaliação do jogo "**As Aventuras de Ada e Turing**" realizada pelo especialista em desenvolvimento de jogos utilizando o *framework* Octalysis

No contexto da jogabilidade de "**As Aventuras de Ada e Turing**", a avaliação utilizando o conjunto de heurísticas foi utilizada para detectar falhas a serem corrigidas em trabalhos futuros. Os resultados desta avaliação apontaram que aspectos relacionados aos gráficos do jogo precisam ser melhor explorados como, por exemplo, a interface em 2D sobre a qual o aplicativo foi desenvolvido. Entretanto, características como o salvamento automático da trajetória do jogador, o mapeamento de situações reais, a apresentação clara dos objetivos e a possibilidade de o jogador utilizar diferentes estratégias para a resolução dos problemas propostos são destacados como diferenciais do jogo.

Os testes de usabilidade indicaram que alguns aspectos da interface precisam ser trabalhados como, por exemplo, a metáfora da roda de bicicleta para representar a repetição. O questionário pós-teste demonstrou que as crianças sentiram-se motivadas a utilizar o jogo, salientando que haviam gostado da interface, do enredo proposto e afirmando que gostariam de utilizar novamente o aplicativo. De forma geral, observou-se que as crianças mais velhas gostaram mais do jogo e tiveram um melhor desempenho, tanto com relação ao entendimento do que era esperado delas, quanto em relação à execução das instruções.

6. Conclusão

O Pensamento Computacional envolve a decomposição de problemas em subproblemas mais simples por meio de algoritmos e a capacidade de aplicar essas soluções em problemas similares, de forma que essas soluções possam ser apresentadas em um formato computacional ou não. Essas habilidades, que se tornam pré-requisito em uma sociedade cada vez mais dependente da tecnologia devem, idealmente, ser desenvolvidas desde os primeiros anos do ensino, em destaque o Ensino Fundamental, período escolar que, por lei, é responsável pela formação básica do cidadão, especialmente quanto à inclusão tecnológica.

Além da influência direta do Pensamento Computacional no desempenho do indivíduo quanto à inclusão tecnológica, sua importância também é destaque quanto à formação do aluno. A influência no desempenho de outras disciplinas, como matemática e física, evidencia-se no desenvolvimento do raciocínio lógico, decomposição de problemas, algoritmos, abstração, etc.

Tendo em vista o exposto, foi criado um protótipo de jogo como apoio ao desenvolvimento de habilidades do Pensamento Computacional. O jogo "**As Aventuras de Ada e Turing**" buscou, em cada uma das suas fases, promover o desenvolvimento das habilidades do PC através de atividades que estavam inseridas dentro de uma narrativa simples, mas que, juntamente com os elementos visuais, procurava engajar o usuário a concluir as dinâmicas propostas. Dentre as habilidades que se busca promover estão: uso de vocabulário, desenho de soluções por meio de análise e de criação de algoritmos e realização de testes e depuração.

Para aferir a usabilidade do protótipo, foram realizados testes com especialistas e com o público-alvo da ferramenta. E, que embora precisem ser ampliados, possibilitaram a identificação de vários aspectos de usabilidade que podem ser melhorados para desenvolver um jogo que se aproxime ainda mais das necessidades e limitações do usuário de forma a criar uma interface transparente, que não demande esforço do jogador para seu entendimento e utilização.

Consideramos que o jogo demonstrou ser uma abordagem adequada quanto à aplicação das habilidades que compõem o Pensamento Computacional em uma ferramenta de apoio, prática que favorece o desenvolvimento do tema mesmo em contextos onde ele não esteja incluído no currículo. Ademais, cria subsídios para novas pesquisas que utilizem essa e outras formas de desenvolver o Pensamento Computacional.

Dentre as possíveis melhorias e trabalhos futuros, além dos testes de usabilidade, é necessário que sejam realizados testes com amostras maiores e com o intuito de validar a eficácia da

ferramenta como forma de promover o desenvolvimento das habilidades do Pensamento Computacional (testes cognitivos). Além de testes mais aprofundados, também deve ser realizada uma reformulação do tutorial, para que este se torne mais interativo e possibilite um treinamento guiado mais extenso.

E, por fim, acreditando na importância do trabalho realizado, consideramos que a ampliação do jogo para apoiar outras habilidades do Pensamento Computacional, como resolver problemas em grupo, seja um dos próximos passos a ser realizado. Também devem ser criados níveis para a diferenciação entre as idades das crianças, e a implementação de um mecanismo de registro do percurso do aluno, de forma a gerar um *feedback* para o professor, de maneira que este possa orientar os alunos nos tópicos que apresentarem mais dificuldades.

Referências

Abreu, C., Rosa, J., and Matos, E. (2017). Fantastic Pirates: Software de Apoio ao Ensino e à Aprendizagem Infantil. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 6(1):252.

Barcelos, T. S., Carvalho, T., Schimiguel, J., and Silveira, I. F. (2011). Análise Comparativa de Heurísticas para Avaliação de Jogos Digitais. *Proceedings of the 10th Brazilian Symposium on Human Factors in Computing Systems and the 5th Latin American Conference on Human-Computer Interaction*, pages 187–196.

Barr, V. and Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *acm Inroads*, 2:48–54.

Bell, T., Witten, I. H., and Fellows, M. (2011). *Computer Science Unplugged: Ensinando Ciência da Computação sem o uso do computador*. csunplugged.org. Tradução coordenada por Luciano Porto Barreto.

Blikstein, P. (2008). O Pensamento Computacional e a Reinvenção do Computador na Educação. `http://www.blikstein.com/paulo/documents/online/ol_pensamento_computacional.html`. Acesso em 30 Junho de 2017.

Burning Glass Technologies (2016). Beyond point and click: the expanding demand for coding skills. *Burning Glass Technologie*.

CAS (2014). Computing in the National Curriculum: A Guide for Secondary Teachers. `http://www.computingatschool.org.uk/data/uploads/cas_secondary.pdf`. Acesso em 08 de Agosto de 2017.

Chou, Y. (2015). Octalysis – complete Gamification framework. `http://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/`. Acesso em 22 Novembro de 2017.

Chou, Y. (2017). What is Gamification. `http://yukaichou.com/gamification-examples/what-is-gamification/`. Acesso em 30 Junho de 2017.

Fundação Victor Civita (2009). O Uso dos Computadores e da Internet nas Escolas Públicas de Capitais Brasileiras. `http://www.smeduquedecaxias.rj.gov.br/nead/`

Biblioteca/GestÃčo/pesquisa_computadores.pdf. Acesso em 04 de Setembro de 2017.

Gal-Ezer, J. and Stephenson, C. (2014). A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States. *ACM Transactions on Computing Education*, 14(2):1–18.

Grover, S. and Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, 42(1):38–43.

ISTE/CSTA (2011). Operational Definition of Computational Thinking for K–12 Education. http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf. Acesso em 02 de Julho de 2017.

OECD (2016). Skills matter. http://dx.doi.org/10.1787/23078731. Acesso em 30 Junho de 2017.

Parsons, D. (2016). Computational Thinking, Digital Fluency and the New Zealand Curriculum. http://davidparsons.ac.nz/seminars/Computational\%20Thinking,\%20Digital\%20Fluency\%20and\%20the\%20New\%20Zealand\%20Curriculum.pdf.

Paula, B. H. d. and Valente, J. A. (2016). Jogos digitais e educação: uma possibilidade de mudança da abordagem pedagógica no ensino formal. *Revista Ibero-americana de Educação*, 70(1):9–28.

Pessoa, F. I. R., Araujo, A. L. S. O., Andrade, W., and Guerrero, D. (2017). T-mind: um Aplicativo Gamificado para Estímulo ao Desenvolvimento de Habilidades do Pensamento Computacional. *Simpósio Brasileiro de Informática na Educação - SBIE*, 28(1):645.

Pinho, G., Weisshahn, Y., Brum, C. F. d., Cavalheiro, G. G. H., and Cavalheiro, S. (2016). Proposta de Jogo Digital para Dispositivos Móveis: Desenvolvendo Habilidades do Pensamento Computacional. *Simpósio Brasileiro de Informática na Educação - SBIE*, 27(1):100.

Raabe, A. L. A., Zorzo, A. F., Frango, I., Ribeiro, L., Granville, L. Z., Salgado, L., da Cruz, M. J. K., Bigolin, N., Cavalheiro, S. A. C., Fortes, S., Malucelli, A., Zorzo, A. F., Nunes, D. J., de S. Matos, E., Steinmacher, I. F., Leite, J. C., de Araujo, R. M., Correia, R. C. M., and de L. Martins, S. (2017). Referenciais de Formação em Computação: Educação Básica. In *Workshop sobre Educação em Computação*, pages 1–9. Sociedade Brasileira de Computação (SBC).

Ramos, D. K., Lorenset, C. C., and Petri, G. (2016). Jogos Educacionais: Contribuições da Neurociência à Aprendizagem. *Revista X*, 2:17. Acesso em 30 de Novembro de 2017.

Santos, C. P., da Silva, J. L., and Genz, C. (2017). Lógica de Programação: Iniciação Lúdica com Play Code Dog. *Anais do Workshop de Informática na Escola*, 23(1):108.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3):33–35.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, pages 3717–3725.

Yadav, A., Hong, H., and Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *Springerlink*.

Zanetti, H. A. P., Borges, M. A. F., Leal, V. C. G., and Matsuzaki, I. Y. (2017). Proposta de ensino de programação para crianças com Scratch e Pensamento Computacional. *Tecnologias, Sociedade e Conhecimento*, 4(1):43–58.

Zanetti, H. A. P., Borges, M. A. F., and Ricarte, I. L. M. (2016). Pensamento Computacional no Ensino de Programação: Uma Revisão Sistemática da Literatura Brasileira. *Simpósio Brasileiro de Informática na Educação - SBIE*, 27(1):21.

Towards better tools and methodologies to teach computational thinking to children

Laís V. Minchillo[1], **Augusto Vellozo**[2], **Edson Borin**[1], **Juliana F. Borin**[1]

[1]Instituto de Computação – Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 – Cidade Universitária, Campinas/SP - Brasil

[2]TecSinapse
Av Dr. Chucri Zaidan, 940 - 16º andar - São Paulo/SP - Brasil

lminchillo2@gmail.com, augusto.vellozo@tecsinapse.com.br,
edson@ic.unicamp.br, juliana@ic.unicamp.br

Abstract. *Computational Thinking is a useful skill to solve problems in all areas of knowledge. Efforts around the world aim to teach children this skill and in some countries it is already part of the curriculum. In this paper, we (i) describe our experiments teaching computational thinking concepts to children, (ii) describe the insights derived from this work, and (iii) propose a set of new hypothesis that should be tested in order to guide the development of a methodology to teach computational thinking to children.*

1. Introduction

Computational thinking (CT) is a tool to solve problems that applies to all areas of knowledge. The term was made popular by Wing (2006), who defined it as solving problems, designing systems and understanding human behaviour through concepts fundamental to computer science (CS).

Computational thinking must not be seen as a technical skill, but rather as a way to organize thoughts and solve problems, and it is natural to consider teaching it in school, either as a separate course, or as a new tool to existing disciplines [Wing 2008, Barr and Stephenson 2011, Barcelos and Silveira 2012, França and Amaral 2013, Lye and Koh 2014, Code.org 2017, CSTA 2017, Buitrago Flórez 2017, Eloy et al. 2017]. In many countries this skill is already part of the basic curriculum [UK Department for Education 2013, Smith 2016], and it is expected that new generations have a better understanding of technology and its different applications.

Teaching computational thinking to children is not a recent idea - Papert (1972) published a paper on the subject. According to him, children learn by doing and by thinking about what they do, and innovation in teaching must bring better things to do and better ways to think. At that time, the author also claimed that computing was by far the richest innovation area for teaching. Papert was one of the creators of Logo, a programming language designed to provide a fun environment for children to study and learn math and programming concepts [Logo Foundation 1991], as well as author of the book Mindstorms: Children, Computers and Powerful Ideas (1980), in which he defends the benefits of teaching computer literacy.

Thanks to Prodecad for hosting our experiments and to TecSinapse for financing this project

Several countries have included computational thinking in their school curricula, as well as programming and CS concepts and other related subjects (such as logical thinking, problem solving, abstraction, planning, among others). The United Kingdom's government has included CT and CS concepts in their national curriculum, stating that a high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world [UK Department for Education 2013]. In the United States, ex president Obama launched an initiative to include Computer Science in the K-12[1] curriculum. In Brazil, the Brazilian Computing Society (SBC) is working to include Computer Science in the national curriculum [SBC 2017].

Over the past decade several authors have described what CT and CS concepts to teach and how to teach them, however, there is still no consensus on how to teach CT and how to include it in the schools. In this paper, we describe our experiments teaching computational thinking concepts to children using a mobile application paired with a physical robot and discuss the insights derived from this work. We also propose a new set of hypothesis that should be tested in order to guide the development of an effective methodology to teach computational thinking to children.

The rest of this paper is organized as follows: Section 2 presents the computational thinking concepts that are frequently discussed by key sources. Section 3 list the related work. Section 4 shows our experimental setup. Section 5 discusses the experimental results and, finally, Section 6 brings our conclusions and suggestions for future work.

2. Programming and CT concepts

Computational thinking is not the same as programming, but rather a skill programmers use in order to solve different problems. As a consequence, using programming as a way to teach CT is common in the literature [Lye and Koh 2014, Buitrago Flórez 2017, Eloy et al. 2017]. In this section we enumerate the main CT concepts cited by a few key sources: the Computer Science Teachers Association (CSTA), a membership organization that supports and promotes the teaching of computer science; Code.org, a non-profit organization dedicated to expand access to computer science in schools and increase participation by women and underrepresented minorities. They organize the annual Hour of Code campaign and provide a curriculum for K-12 computer science and they are supported by several companies including Amazon, Facebook, Google and Microsoft; SBC, Brazilian Computing Society, a non-profit organization whose goal is to encourage research and teaching in computing. We also chose three of the most cited papers in the CT topic - Barr and Stephenson (2011), Grover and Pea (2013) and Brennan, K. and Resnick, M. (2012) - as well as one paper well cited in Brazil - França and Amaral (2013) - one of the first works in the country to discuss teaching CT in schools.

These concepts are:

- Sequence: a series of individual steps.
- Algorithm: a sequence of instructions to solve a task.
- Loop: the execution of the same sequence multiple times.
- Event: an external action that triggers a command sequence.
- Conditional: making decisions based on predefined conditions.

[1] K-12 is the school curriculum for children aged up to 12 years old.

- Debugging and testing: executing an algorithm to find errors or to validate the proposed solution.
- Problem decomposition and modularization: divide a problem in smaller ones that can be solved more easily.
- Function: a sequence of instructions one can use with a given input to execute a task, possibly generating an output and modifying the original input to better suit it's purposes.
- Nested loop and conditional: a loop within a loop, or a conditional with another condition.
- Recursion: a function that calls itself.
- Parallelism: executing more than one instruction at a time, or execute more than a sequence of instructions at a time. Parallel tasks can be independent or not.

Table 1 shows the list of sources and concepts discussed by them.

Table 1. Computational thinking concepts by author

	CSTA 2017	Code.org 2017	SBC 2017	Barr and Stephenson 2011	Grover and Pea 2013	Brennan and Resnick 2012	França and Amaral 2013
Sequence		•	•	•	•	•	•
Algorithm	•	•	•	•	•		
Loop	•	•		•		•	•
Event	•	•			•	•	•
Conditional	•	•		•	•	•	•
Debugging	•	•	•	•	•	•	•
Test	•	•	•	•	•	•	•
Decomposition, functions, modularization	•	•	•	•	•	•	•
Nested for, nested if	•	•					
Parallelism	•			•	•		•
Recursion	•		•	•	•		•

Some sources also propose to group these concepts in modules, each one associated with a target age range. Tables 2 and 3 shows two of such module divisions.

Table 2. Computational thinking modules, Code.org

Level	Ages	Prerequisites	Concepts
1	4 to 6	Reading (basic)	Sequence, loop, event, problem solving
2	6 or more (Recommended 7 to 10)	Reading, math	Conditional, algorithm, debugging
3	6 or more (Recommended 9 or 10)	Completed module 2	Problem decomposition, functions, nested loop, nested conditional

Table 3. Computational thinking modules, CSTA K-12

Level	Ages	Concepts
1	8 to 11	Algorithm, problem solving, design and implementation, test, problem decomposition
2	11 to 14	Algorithm, design and implementation, parallelism, abstraction, problem decomposition, check different algorithms that solve the same problem
3	14 to 17	Functions, parameters, classes and pre-defined methods in problem solving, describing steps to developing software, explain how sequence and recursion are used to build algorithms, design and simulation of environments, abstraction, parallel programming

In Brazil, SBC (2017) has started to define how computing should be included (or modified) in the national curriculum. There are three main categories:

- Computational thinking: understand and use models and representation to describe information, processes and techniques to build algorithmic solutions; describe solutions though algorithms that can be executed in parts or in total by machines, as well as build computational models for complex systems; analyze problems and solutions to not only find automated solutions, but be able to evaluate their efficiency and correctness.
- Digital world: understand how information can be described and stored; understand how information is processed by computers and the relation between hardware and software; understand how digital devices communicate with each other, how the data is transmitted and how the integrity and safety of information is guaranteed.
- Digital culture: understand the impact of the digital revolution and advances in the digital world on humanity; utilize in an efficient and critical manner tools to help obtain, analyze, synthesize and communicate information of different formats and with different purposes; analyze ethical and moral questions created by the digital world.

Table 4 shows how SBC is grouping computational thinking concepts by school level.

Table 4. Computational thinking concepts by school level, SBC

Level	Concepts
Preschool Ages 3 to 5	Understand a problem and identify a sequence of steps to solve it. Represent these steps in an organized manner. Create steps to solve problems related to body movement and spatial trajectories.
Elementary school Ages 6 to 10	Abstraction to describe data such as lists and graphs. Identify the abstractions needed to build steps and to define algorithms that involve daily situations around the children. Use a visual language to represent algorithms. Understand problem decomposition.
Middle school Ages 11 to 14	Use visual and native languages to represent data and processes. Formalize the concepts of data structures. Use recursion to solve problems. Build new solutions by reusing solutions to problems of different context. Relate an algorithm in visual language to code in a programming language.
High school Ages 15 to 17	Work in groups designing solutions to problems integrated in other areas of the curriculum using computers, phones and other computing machines. Compare problems and reuse solutions. Analyze algorithm's cost and efficiency and justify if a solution is feasible and adequate. Argue about algorithm's correctness. Understanding the limits of computing to differentiate what can or can not be automated.

3. Related work

Several authors have described which CT and CS concepts to teach and how to teach them, and as we discussed in the previous sections, in some countries this has already been included in the national school curriculum offering every child the opportunity to learn and benefit from computational thinking. The fact that there is no consensus on how to teach CT and how to include it in the schools, especially in Brazil, has been part of our motivation for this work.

There has been several initiatives towards teaching CT in Brazil. Eloy et al. (2017) described an experience training teachers with the goal of promoting the practice of programming and development of CT in Brazilian public schools. In their pilot project they worked on four main areas: implementation in schools, curriculum design, teacher training and monitoring and evaluation. Their first guiding material to build the curriculum was the online platform Programaê[2]. Teachers were included in improving this curriculum through discussion sessions and questionnaires. They had over 500 students participating in the activities, and the sessions taking place in 2016 had an average of 80% student retention.

[2]http://programae.org.br/

Godinho et al. (2017) present a project to introduce CT and encourage children to become technology creators. The project was recognized by SBC in 2016 for bringing computing to children, teenagers and people who otherwise had little contact with the area. Their encounters included mini-courses, unplugged activities or tasks in Code.org's Hour of Code[3], Scratch[4], CodeMonkey[5], Monster Coding[6] or App Inventor[7]. Almost 300 students participated in their activities and through feedback questionnaires approximately 90% rated the experience as excellent or great.

Aono et al. (2017) use Scratch allied with an expository methodology to teach CT to elementary school students with ages 10 and 11. The children applied the concepts they learned into a project: building a "Flappy Bird" game. Every student that participated was able to build the game successfully, but all of them needed some help from the supervisors to implement the hardest parts, like the use of variables and counters.

Silva Junior and França (2017) discuss how existing tools are being used in the classroom in Brazil and their effectiveness. In total, 9 tools were analyzed for their interaction, platform, programming language and other characteristics. The tool found to be most adequate was Portugol Studio, since it is fully available in portuguese, it is appropriate for beginners and it features a user friendly interface. Besides that, it offers features to help teachers use it in their classes.

In this work, we aimed to teach CT to children using a mobile game paired with a physical robot, in an informal environment - unlike the regular classroom setting. Our goal was to have the children free to explore the application and learn from its use. The robot appears only as a motivating factor and as part of the play. The game itself saves logs of several actions, allowing us to analyze more than just the code the students developed, but rather see the interaction as a whole.

4. Experimental setup and methodology

We designed experiments with students from a local public school. In total there were twenty five children aged 9 to 11. The children were asked to solve problems by controlling a physical robot using a block-based visual programming language. The problems and the tools were designed to motivate the children to learn computational thinking skills. The tools and the methodology are described in the next sections.

4.1. Teaching tools

We designed and implemented an Android application and a physical robot to support our experiments. The application communicates with the robot via Bluetooth and provides a visual block-based programming interface through Google's Blockly library [Google for Education 2012]. Figure 1 shows a picture of the experiment's environment, including the robot and a tablet running the application.

In some of the activities, the user is required to solve a problem by programming the robot using a block-based visual programming language. The application provides a canvas in which the users can drag programming blocks and connect them to compose

[3] https://code.org/learn
[4] https://scratch.mit.edu/
[5] https://www.playcodemonkey.com/
[6] http://monstercoding.com/
[7] http://appinventor.mit.edu/

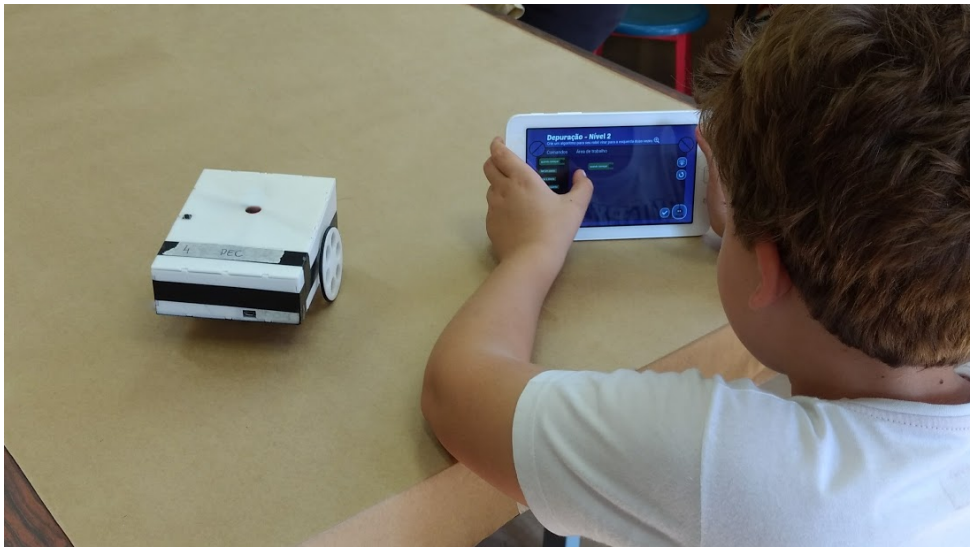
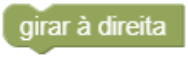


Figure 1. The experiment's environment, including both the robot and the application

their program. Blocks are shaped so that only connections that make sense are allowed. For example, the user may add a "Step forward" and a "Turn right" block and connect them to express a sequence of commands, however, a "Number" block may not be connected to a "Step forward" block. This feature minimizes issues associated with programming syntax, which are common in text based programming languages. Table 5 lists the programming blocks available in the application and Figure 2 shows a screenshot of the application, in which the user has a canvas on the right side and the blocks available for the given activity.

Table 5. Programming blocks

Step forward		Turn left	
Turn right		Number	
Repeat		Repeat a number of times	
Number comparison		Distance in front of the robot	
If		If else	

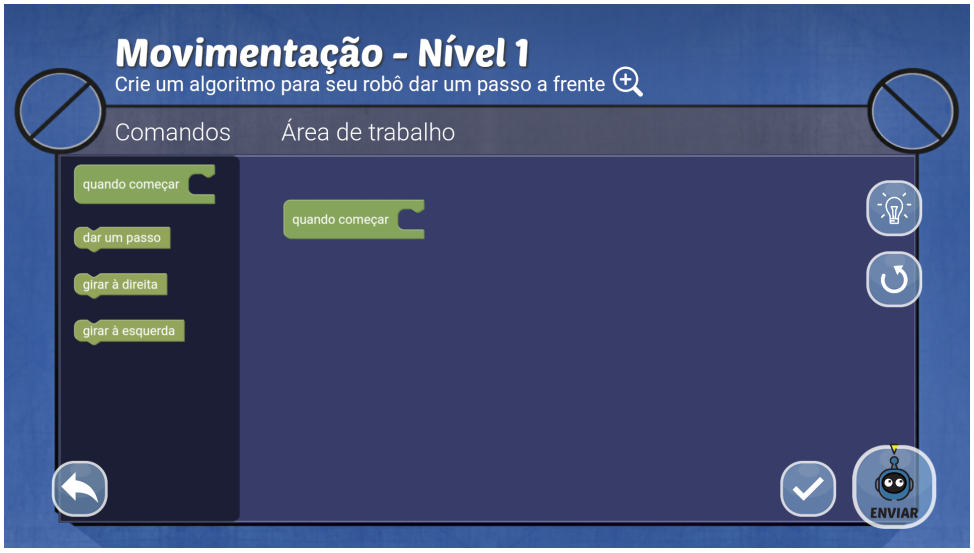


Figure 2. One of the challenges in the application

The Android application is a prototype of an educational game that has six levels, each one introducing new concepts and having multiple activities within it. Table 6 presents the levels and the CT concepts used in each one of them. The CT concepts were selected based on the age of the children and the amount of time we would have with them.

Table 6. Proposed levels and concepts

Levels	**Concepts**
Move the robot	Sequence
Have the robot make a path of a certain shape (e.g. square)	Sequence, algorithm
Find mistakes in given algorithms	Sequence, algorithm, testing, debugging
Have the robot repeat the same tasks multiple times	Sequence, algorithm, loop, problem decomposition
Have the robot decide on what action to take based on external conditions	Sequence, algorithm, conditional
Combine repetition and conditional scenarios	Sequence, algorithm, conditional, loop

There were three types of activities available: **tutorials** that explained what concepts the following activities would involve, and hints of how the user could solve the problems that would be presented next; **quizzes** to test the user knowledge, each one comprised by a question and three possible answers - of which only one was correct; **programming challenges** that involved either creating a new algorithm to solve a problem or to fix an existing algorithm.

In order to help our evaluation we had the application log some of the actions, as listed in Table 7, as well as the timestamps and the current activity identification number.

Table 7. Application logs

TUTORIAL-OPENED	User opened the tutorial
TUTORIAL-NEXT	Next tutorial page
TUTORIAL-PREV	Previous tutorial page
TUTORIAL-CLOSE	User closed the tutorial
QUIZ-OPEN	User opened the quiz
QUIZ-CORRECT	Correct quiz answer
QUIZ-INCORRECT	Incorrect quiz answer
QUIZ-CLOSE	User closed the quiz
ACTIVITY-OPEN	User opened the (programming) activity
ACTIVITY-HINT	User clicked hint
ACTIVITY-SEND	User sent the algorithm to the robot
ACTIVITY-CORRECT	User marked the current solution as correct
ACTIVITY-CLOSE	User closed the activity

4.2. Methodology

In total we had three sets of experiments - the first one with fourteen students, the second one with six students, and the third one with five students. In total there were ten girls and fifteen boys. One of our ideas was that children should work in pairs so that one could help the other. In the first experiment the children were divided in pairs by their teachers. This proved to be a poor strategy since some of the children were very uncomfortable with the person they were paired to. For this reason, in the next two experiments children were allowed to choose their pair and generally this proved to be a better approach.

For each experiment we had around seven encounters, one for introduction and the others for the actual activities. In the last session of each experiment we also asked the participants to answer a short feedback form - twenty two students filled this form out.

In the introduction session we provided a brief explanation of the research and interested students received a Consent Form to be signed by their parents or legal tutors. Also, as part of the introduction, each group was asked to name its robot.

After the children split in pairs and we handed over the robots and tablets, they were free to explore the application. The application itself is not capable of evaluating the user's solutions to programming challenges - there are several solutions to each of the proposed challenges, so simply having one correct algorithm and testing that it matches user's input would not suffice. To evaluate user's algorithms we would have needed a much more sophisticated setup so the application could have the robot's step-by-step information to only then determine if the desired solution was reached. We decided to let the students evaluate their own solutions by watching the robot to see if it behaved the way it was expected to. By doing so we could also observe whether the students had the ability to decide if their solution was correct.

We aimed at creating a playful and spontaneous environment, having as few evaluations and interventions as possible throughout the experiment, hoping it would encourage the children to *want to play*, rather than make them feel they *had to*. We could observe from the very beginning that competition was a much stronger motivating factor than collaboration. For that reason, in some occasions we proposed having the robots compete in a race with obstacles or an arena where the last robot standing would be the winner. The rules of the arena were simple: the robots had to keep moving and if it hit a wall or another robot, it was out for the round. Our goal was to combine the concepts of conditional (only move forward if there is no obstacle detected by the ultrasonic distance sensor, otherwise turn left or right) and repetition (never stop moving). In these competition environments it was clear that the students tried several solutions, reaching for the best one possible.

5. Results

At the introduction session we learned that most children were familiar with the use of smartphones, tablets and computers. They seemed very excited to work with the robots - particularly the boys. Asking each group to name its robot worked even better than anticipated because they developed a personal connection with it throughout the experiment.

During the experiments we aimed to intervene as little as possible while also being available to clear any doubts or to help if the children got stuck in any task. We observed that a very loose environment compromises their ability to focus on the proposed activities; however, it had a very positive effect on their will. This was confirmed by the teachers, who stated that the children were very excited to be participating in the experiments, especially considering they took place in their free period, when they could choose the activity they liked best.

One thing that stood out in their behavior was that they clearly preferred competition to collaboration. The pairs that were supposed to be working together divided the activities in a round robin fashion, and instead of helping each other they rushed their colleague so they could get to play with the robot faster. However, when an environment of team competition occurred - like the robot races - they would collaborate with their teammate to reach a better solution and try to win. In the regular activities, the pairs were not so motivated to keep trying different solutions when the first one didn't work, and would often lose attention in the current task and go see what other children were doing. Because of this behavior we expect that the children would reach their best when working individually or in a competition setting.

Another thing that stood out was that boys and girls showed a very distinct behavior: boys wanted to grab both the robot and the tablet straight away and go play with it - although not necessarily play within the scope of the proposed activities. Rather than that, most boys wanted to complete the tasks as fast as they could to either reach other groups that were in more advanced stages or to be the first to get to those stages. Girls on the other hand showed a much higher interest in reading the tutorials and completing the activities successfully. The children, in general, were very interested in discovering what the parts of the robot could do - like the "eyes" (an ultrasonic sensor) - and how to make it move or turn on the LEDs.

Because the application was unable to evaluate the user's solution, it was expected that the children themselves would figure out whether they had successfully completed the

given task or not. In many cases, they marked an activity as done even when they didn't program the robot as expected. There are some factors to consider:

- First, they could have thought that their solution was correct even if it wasn't, so it was an honest mistake.
- Being very familiar with other games and applications that can evaluate the solutions, they thought the application would only let them mark an activity as done if their solution was correct, so that's also an honest mistake.
- The third (and possibly worse) case is when the children mark the activity as done, when they knew it wasn't, so they could pass on to the next levels - to either reach their colleagues or to be the first to get to the next levels.
- There is also another extreme case: when the children's solution is correct but they aren't sure it is, so they keep trying to alter the code in order to see a better result in the robot.

Perhaps all of these issues can be solved by having a teacher or tutor check the child's solution before they can move on, and this certainly looks like a good solution to handle the aforementioned problems. However, the impact this aproach could have on the children's engagement should be evaluated.

We analyzed the application's logs in order to look for some correlations in the statistics. First, we graded the programming tasks using the following scale: 10 - completed the activity, 5 - partially completed the activity, and 0 - didn't complete the activity. Then, we compared the average grade in programming activities against the average number of wrong quiz answers. The result is presented in Figure 3, which contains a red line showing a linear fit of the data. Notice that the red line suggests that there is a negative correlation between these two metrics, i.e., the better the students scored in programming activities the less they select wrong quiz answers.

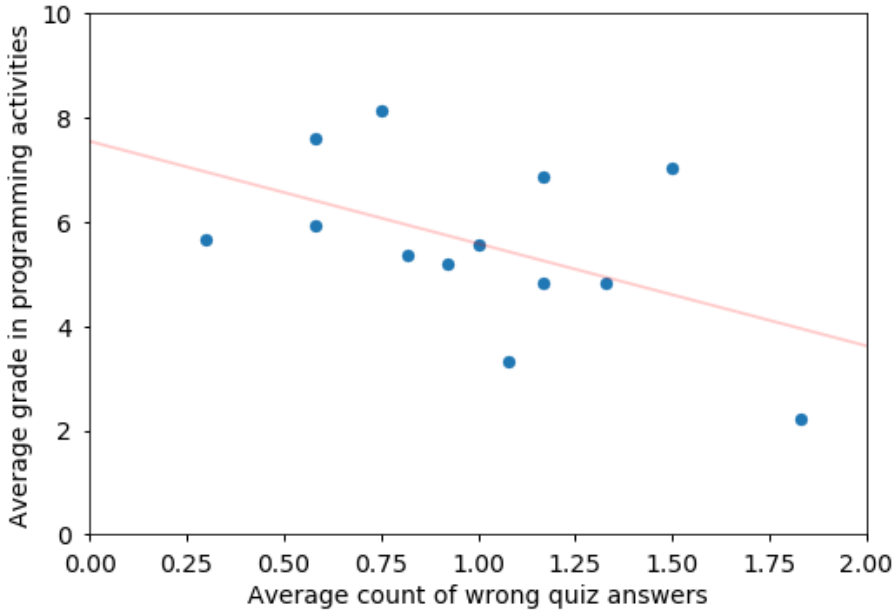


Figure 3. Grade in programming tasks versus count of wrong answers in quizzes

We also analyzed the percentage of the tutorials the children read. Each tutorial had a certain number of pages, and we checked the number of pages the children read.

Some children only opened the first page and already quit - meaning they never read the following pages.

Prior to the experiments, we anticipated that the children who read the tutorials would have a better performance in the quizzes and programming activities. As Figure 4 indicates, there seems to be a positive correlation between reading the tutorial pages and scoring higher on programming activities.

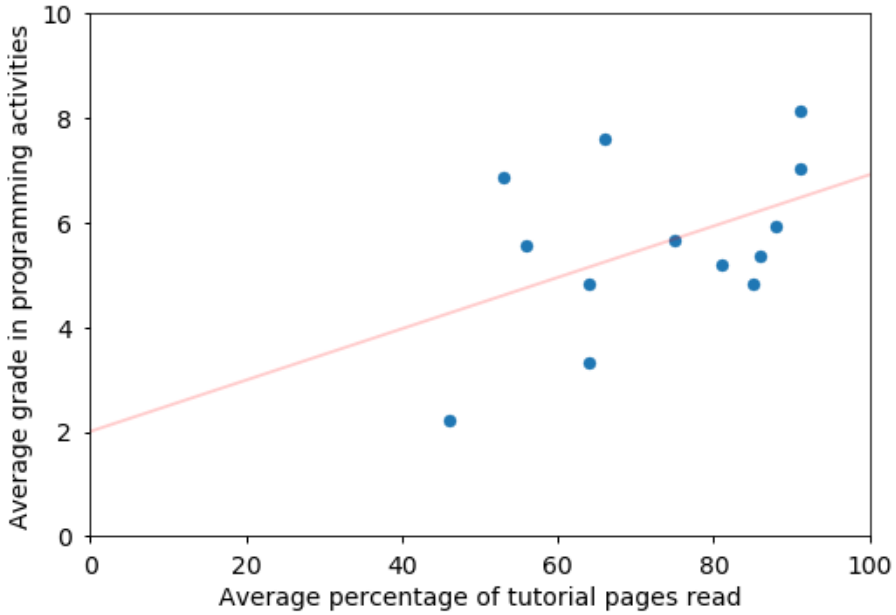


Figure 4. Grade in programming tasks versus percentage of tutorial pages read

Even though the linear regression indicates that there is a positive correlation between grades in programming activities and number of tutorial pages read, we noticed that different groups with very similar grades had read different amounts of tutorials pages. One of the possible reasons this happened is due to the different children's backgrounds. Some of them could already have more knowledge of similar games and activities, and therefore even not reading the tutorials - or reading less of them - had a better comprehension of the activities and how to solve them.

Figure 5 shows the average count of wrong quiz answers against the percentage of tutorial pages read. Again, as indicated by the linear regression, there seems to be a negative correlation between the percentage of tutorial pages read by the students and the amount of incorrect quiz answers.

In general, these results indicate that children who read more tutorials achieve higher grades in programming tasks and select fewer incorrect answers in quiz activities.

We also evaluated how well the children understood each concept by looking at the average grade that was obtained for the programming tasks involving that concept. Figure 6 shows a graph of average grades per concept. Activities involving the first concepts - movement, sequence and debugging - were completed with success by most groups, indicating that the students either learned the concepts or had previous knowledge of them. The activities that involved the loop concept were only completed partially, with a much lower grade than the previous ones. Finally, no child was able to achieve the

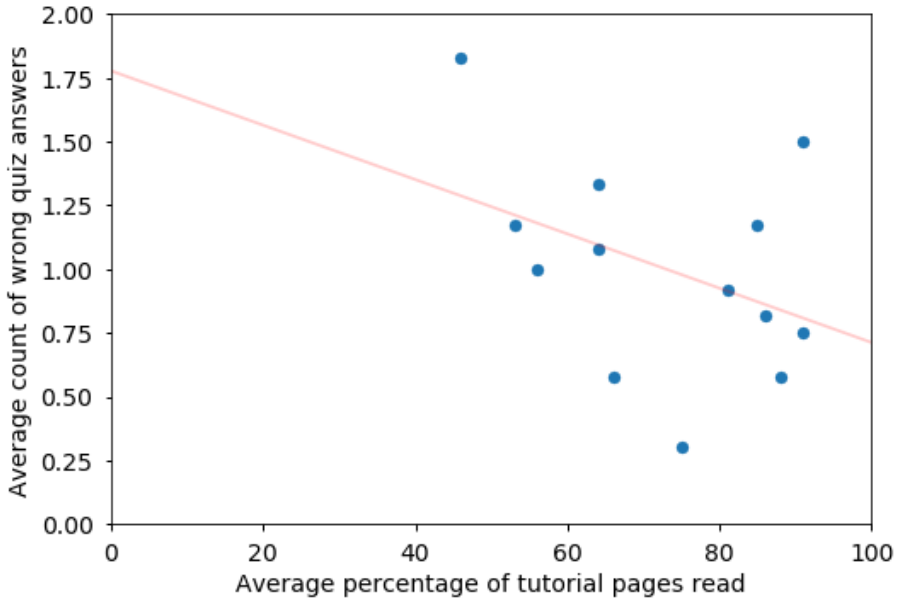


Figure 5. Count of wrong answers in quizzes versus percentage of tutorial pages read

expected results for the conditional concept. It is still unclear whether children at that age can't understand this concept well or if the way it was presented in our experiment was too complicated.

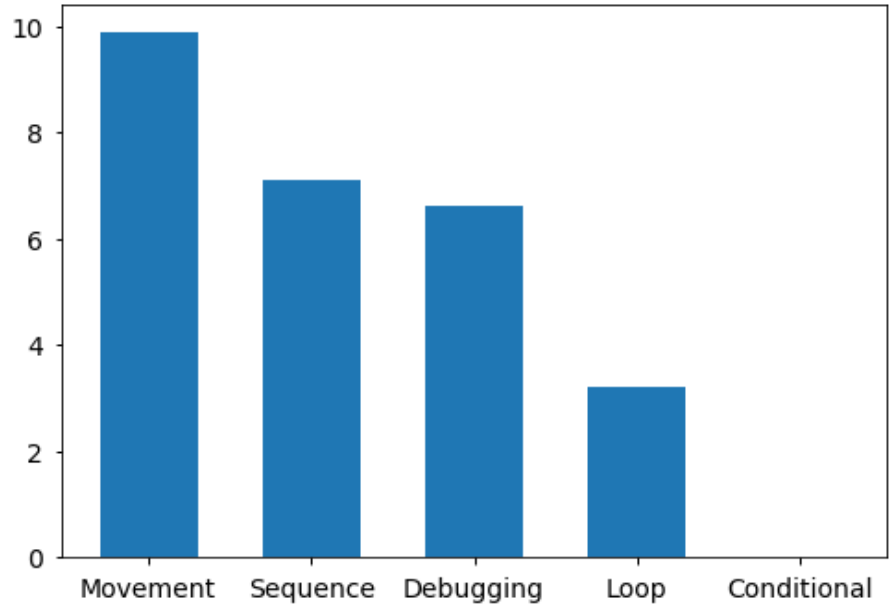


Figure 6. Average grade for all groups by concept

Table 8 shows the answers to the feedback form. Each question had three possible answers: yes, partially, and no. Overall, the feedback provided by the children was very positive - we only had 5.5% of negative answers across all questions, against 25.5% neutral and 69% positive. The only question that did not have a significant positive outcome was "Is the application easy to use?", indicating that we need to evaluate what's the best

Table 8. Questions and answers in the feedback form

Question	Positive answers	Neutral answers	Negative answers
Would you use the application and robot again?	16	5	1
Would you recommend the application and robot to a friend?	17	4	1
Is the application easy to use?	7	14	1
Is the application content fun and interesting?	19	2	1
Did you like working in pairs?	17	3	2
Average	**15.2 (69%)**	**5.6 (25.5%)**	**1.2 (5.5%)**

way to present the programming interface. The teachers also gave us very positive feedback, stating that the students were very excited to be participating and that they felt this was an excellent way to keep their attention.

The form also included space for comments and suggestions. Many of the children wrote that they would like to see other types of robots and other features: "*My suggestion is to create other types of robot and present it to other schools.*", "*I wish it [the robot] could talk and that it had a laser.*", "*My suggestions are that the robot should have arms and legs like other toys and be able to speak.*", "*A suggestion is to make the robot faster.*".

About working in pairs, the children were conflicted: some liked it ("*I liked working in pairs. It's fun that every level is different and has a tutorial.*") and other did not ("*I liked the idea of working in pairs but I didn't like my pair.*").

Finally, many of the feedback was related to having more activities and levels: "*I think it would be cool if you made more activities and different things to do.*", "*I wish there were more worlds.*", "*More levels. It would be fun and come back next year so more people can enjoy this project.*".

6. Conclusion

Over the past decade several authors discussed the importance of and methods to teach CT and CS concepts to children, nonetheless, there is still no consensus on the best teaching methodology. In this paper, we described our experiments teaching computational thinking concepts to children between 9 and 11 years old and share our insights in order to support the development of effective tools and methodologies to teach CT to children.

Our results indicate that:

- children were very excited to interact with the robot;
- by giving a name to the robot, children established a personal connection with it improving their engagement in the experiment;
- competition is a motivating factor and encourages teamwork;

- children did not have any difficulty with sequence concept, however, they had a hard time applying loop and conditional concepts.

Additionally, the interaction with the children helped us design the following **new hypothesis**:

- having a physical instrument to interact with is a motivating factor, whether it is a robot, a board, a set of command pieces, etc;
- adding tests, exams or other forms of formal evaluation can have a negative impact on the way children see the CT education project;
- leaving the children free to use the application and play with the robot make them more comfortable, but it does not necessarily mean a positive impact on their learning;
- having one robot per child or projects that require collaboration between the children to be completed improves learning experience and children's engagement in the activities;
- competition improves children's engagement in the activities;
- have teachers suggest activities that complement the regular disciplines may improve learning and engagement in the classroom.

Future experiments with a higher number of children and for a longer period are needed in order to test these hypothesis. We believe that such a study would greatly contribute to the development of methodologies to teach CT and CS to children.

References

Aono, A. H., Rody, H. V. S., Musa, D. L., Pereira, V. A., & Almeida, J. (2017) A Utilização do Scratch como Ferramenta no Ensino de Pensamento Computacional para Crianças. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2169).

Barcelos, T. S. & Silveira, I. F. (2012). Pensamento computacional e educação matemática: Relações para o ensino de computação na educação básica. In XX Workshop sobre Educação em Computação, Curitiba. Anais do XXXII CSBC (Vol. 2, p. 23).

Barr, V. & Stephenson, C. (2011) Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. Acm Inroads, 2(1), 48-54.

Brennan, K. & Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (pp. 1-25).

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. Review of Educational Research, 87(4), 834-860.

Code.org. (2017) Curriculum, https://code.org/educate/curriculum.

CSTA. (2017) CSTA K-12 Computer Science Standards, Revised 2017, https://sites.google.com/site/cstastandards/standards.

Eloy, A. A. D. S., Martins, A. R. Q., Pazinato, A. M., Lukjanenko, M. D. F. S. P., & Lopes, R. D. D. (2017, June). Programming Literacy: Computational Thinking in Brazilian Public Schools. In Proceedings of the 2017 Conference on Interaction Design and Children (pp. 439-444). ACM.

França, R. S. de, & Amaral, H. J. C. do. (2013) Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do Scratch. In Anais do Workshop de Informática na Escola (Vol. 1, No. 1, p. 179).

Godinho, J., Torres, K., Batista, G., Andrade, E., & Gomide, J. (2017) Projeto Aprenda a Programar Jogando: Divulgando a Programação de Computadores para Crianças e Jovens. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2140).

Google for Education. (2012), Blockly, https://developers.google.com/blockly/.

Grover, S. and Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

Logo Foundation. (1991) Logo, http://el.media.mit.edu/logo-foundation/.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. Computers in Human Behavior, 41, 51-61.

Papert, Seymour. (1972) Teaching Children Thinking, Programmed Learning and Educational Technology, 9(5), 245-255.

Papert, Seymour. (1980) Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

SBC. (2017) Referenciais de Formação em Computação: Educação Básica, http://www.sbc.org.br/noticias/10-slideshow-noticias/1996-referenciais-de-formacao-em-computacao-educacao-basica.

Silva Junior, S. M. da, & França, S. V. A. (2017) Programação para todos: Análise Comparativa de Ferramentas Utilizadas no Ensino de Programação. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2199).

Smith, Megan. (2016) Computer Science For All, The White House, https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.

UK Department for Education. (2013) National curriculum in England: computing programmes of study, https://www.gov.uk/government/publications/national- curriculum-in-england-computing-programmes-of-study.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, 366(1881), 3717-3725.

AS AÇÕES DO PET NO DESENVOLVIMENTO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Leonardo Bandeira de Lucena[1], **Giovana Lorena Costa de Andrade**[1], **Elisa de Fátima Andrade Soares**[1], **Wilton Silva dos Santos Júnior**[1], **Álvaro Gabriel Gomes de Oliveira**[1], **Lígia Maria de Souza Dantas Batista**[1], **Daniel Alves Gomes**[1], **Rommel Wladimir de Lima**[1,2]

[1]Programa de Educação Tutorial em Ciência da Computação

[2]Departamento de Informática – Universidade do Estado do Rio Grande do Norte (UERN)
Caixa Postal 70 – 59.610.210 – Mossoró – RN – Brasil
{leonardolucena.cc, giovana.lca, elisandradecc, wiltonjunior2010, alvarogab6, ligiamsdb, daniel.alvessg}@gmail.com, rommelwladimir@uern.br

Abstract. *Undergraduate in the country is going through a crisis that goes from the high dropout rate to the low level of schooling of the students, passing through the low income level of the graduates. In this context, the Tutorial Education Program (PET) brings in its guidelines the search for improvements in teaching-learning in graduate courses in the country. In this sense, the objective of this work is to present the initiatives of PET Computer Science in the fight against these adversities. As a result, it can be seen that the actions promoted, besides developing the PET members both academically and as citizens, it is observed that these activities have also played an important role in divulgation and promoting the course with the community.*

Resumo. *A educação superior no país passa por uma crise que vai do alto índice de evasão ao fraco nível de escolaridade dos ingressantes, passando pelo baixo nível de rendimento dos egressos. Nesse contexto, o Programa de Educação Tutorial (PET) traz em suas diretrizes a procura por melhorias do ensino-aprendizagem nos cursos de graduação do país. Nesse sentido, o objetivo deste trabalho é apresentar as iniciativas do PET Ciência da Computação no combate a essas adversidades. Como resultado, verifica-se que as ações promovidas, além de desenvolver os integrantes do PET de forma acadêmica e como cidadãos, observa-se que essas atividades também têm desempenhado um importante papel na divulgação e promoção do Curso junto a comunidade.*

1. Introdução

Na última década as políticas educacionais do governo brasileiro proporcionaram um grande aumento na oferta de vagas para o ensino superior. Contudo, o que se observa, empiricamente, é que a qualidade dos estudantes que entram no sistema é cada vez mais precária. Isso faz com que a educação superior no país passe por uma crise, que vai do alto índice de evasão ao fraco nível de escolaridade dos ingressantes, passando pelo baixo nível de rendimentos dos egressos.

Nesse sentido, algumas ações têm sido propostas para melhorar a qualidade do ensino superior e uma das mais antigas é o Programa de Educação Tutorial (PET). Fundado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), em 1979, o PET tinha como objetivo melhorar as condições de ensino-aprendizagem nos cursos de graduação das Instituições de Ensino Superior (IES) do país, trazendo atividades extracurriculares aos grupos tutoriais de alunos. Reformulado e gerido pela Secretaria de Educação Superior do Ministério da Educação (SESu/MEC), a partir do final de 1999, o programa passou a integrar em suas diretrizes atividades de ensino, pesquisa e extensão. Em 2004 o programa passou a ser chamado de Programa de Educação Tutorial, porém, ainda manteve o acrônimo PET como referência ao programa [BRASIL 2001, BRASIL 2006].

O PET é um programa de apoio e melhoria aos cursos de graduação das IES do país. Procura melhorar a qualidade do ensino, visando a integração de atividades extracurriculares que favoreçam a base curricular dos cursos vinculados a iniciativa, além de auxiliar na formação de profissionais no aspecto acadêmico e social preparando-os para mercado de trabalho. É composto por um grupo de alunos, tutorados por um professor vinculado ao curso. Ainda prevê uma bolsa, financiada pelo MEC, de estímulo aos discentes participantes do programa, e um custeio para subsidiar as atividades desenvolvidas pelo grupo [BRASIL 2006].

Nesse sentido, o objetivo deste trabalho é apresentar as atividades desenvolvidas pelo PET Ciência da Computação (PETCC) que têm como objetivo diminuir os problemas apresentados dentro do curso de Ciência da Computação. Para isso, este trabalho está organizado da seguinte maneira: Seção 2 apresenta às ações desenvolvidas pelo grupo; a Seção 3 apresenta os resultados e discussões das atividades executadas lado ao tema discutido; por fim a Seção 4 apresenta as conclusões a respeito das contribuições do grupo PET sobre o ensino da computação.

2. Ações

Esta seção apresenta as principais intervenções desenvolvidas pelo PETCC, da Universidade do Estado do Rio Grande do Norte (UERN), que promovem e fortalecem o ensino da computação e a democratização do curso através da integração entre os discentes e professores.

2.1. Monitorias

A monitoria é um objeto de ensino-aprendizagem que contribui para a formação integrada do aluno nas atividades de ensino, pesquisa e extensão dos cursos de graduação [Lins et al. 2009]. Dessa forma, foi criado pelo PETCC um núcleo de monitorias para realizar tarefas como resolução de listas de exercícios, elucidação de dúvidas e integralização do aprendizado do aluno através de dicas e conselhos para viabilizar o aprendizado dos participantes do projeto. Todavia, os novos ingressantes deparam-se com dificuldades em disciplinas do curso, gerando assim, um elevado índice de reprovação.

O projeto trata-se de um serviço de apoio pedagógico que visa oportunizar o desenvolvimento de habilidades técnicas e aprofundamento teórico, proporcionando o aperfeiçoamento acadêmico [Scarparo Haag et al. 2008]. Ainda proporciona competências importantes para o petiano, como o desenvolvimento da prática de estudo e

apresentação de um conteúdo ao público, compreensão de dúvidas alheias e aumento do conhecimento na disciplina ministrada.

A monitoria, além de ajudar a tirar dúvidas, tenta ao máximo mediar o conteúdo de forma enxuta através de uma linguagem informal para uma maior absorção de tópicos. Isso facilita o processo, pois não passa de uma interação de alunos para alunos. Em seu escopo, integram-se atividades que explanam e revisão os conteúdos dados em sala de aula. Também são desenvolvidos exercícios por meio de listas para uma maior fixação dos conteúdos discutidos durante a atividade.

Esta atividade contribuiu de forma decisiva para o auxílio dos alunos nas disciplinas e para uma plena compreensão e aprendizagem. As monitorias ministradas foram também essenciais para aqueles alunos que não conseguiam absorver por completo todo o conteúdo ministrado pelos professores ou que sentiam-se intimidados em realizar perguntas em meio a público, fato que é muito comum em alunos novatos. Ser monitor é ser um colaborador no processo de ensino e aprendizagem do graduando, além de ser também a função de um petiano.

Às monitorias conseguiram atingir uma média de 10 alunos cada, contabilizando-se desde o início do projeto até a presente data. Foram realizadas ao todo 11 monitorias, às quais constituíram-se de disciplinas tanto específicas do curso quanto de demais áreas associadas à computação, dentre elas: Álgebra Abstrata, Lógica Matemática, Dispositivos Semicondutores e Teoria dos Circuitos e demais disciplinas de programação. Os alunos com presença assídua nas monitorias obtiveram um maior rendimento em suas respectivas disciplinas, o que, consequentemente, os levou a aprovação nas mesmas.

2.2. Revista RedINFO

Um desafio que as universidades enfrentam é formar indivíduos capazes de sempre buscar conhecimentos em sua área, tornando-se um excelente profissional inserido no mercado de trabalho. A "informação é muito mais que um conjunto de dados. Transformar esses dados em informação é transformar algo com pouco significado em um recurso de valor para nossa vida pessoal ou profissional" [Fontes 2006]. Logo, em meio à profusão de dados, hoje, identificar fatos e conteúdos verídicos se torna uma tarefa confusa e trabalhosa, principalmente quando há um déficit de fontes sobre o assunto e falta de propriedade nas informações que circulam pelas redes.

Especificamente na área tecnológica, que sofre mudanças rapidamente, é imprescindível a atualização de saberes e contínuo desenvolvimento de habilidades para aumentar o potencial do profissional das Tecnologias da Informação (TIs), aperfeiçoando suas habilidades e competências. Levando este cenário para as IES, pode-se observar que frequentemente os discentes não tomam conhecimento de assuntos importantes para sua formação.

A REDinfo (Revista Eletrônica de Computação) é um mecanismo para a obtenção de informações sobre as TIs e soluções tecnológicas que têm surgido, através de manchetes, entrevistas, curiosidades e dicas, proporcionando um leque de conhecimentos para os graduandos, não apenas de Ciência da Computação, mas a toda comunidade acadêmica. A 1ª edição foi lançada em 2011, com o tema: Engenharia de software, este projeto foi estacionado na 4ª edição no ano de 2014. Porém, com a análise dos resultados satisfatórios, a revista foi retomada com a 5ª edição, Figura 1, no ano de 2017 com o tema: Os

desafios da computação, abordando sobre a Computação em Nuvem, Internet das Coisas, Computação Verde, Segurança de Redes e Cidades Inteligentes, a qual terá o evento de lançamento programado para o primeiro semestre de 2018.

Figura 1. Capa da 5ª edição da REDinfo

O procedimento metodológico se deu por meio de pesquisas sobre as TIs que estão tendo maior usabilidade e melhores soluções, em seguida o desenvolvimento das manchetes, correção, diagramação e o evento de lançamento para a propagação do projeto. Este é um projeto realizado pelos alunos de Ciência da Computação da UERN, associados ao PETCC.

2.3. Reaproveitamento do Lixo Tecnológico

A produção industrial, hoje em larga escala, estimula a sociedade a aumentar o seu consumo, criando assim uma nova diversidade de resíduos [Clezar Júnior 2006]. Nota-se que milhares de aparelhos e equipamentos eletrônicos são substituídos todos os dias por se tornarem obsoletos [Moi et al. 2014]. "O Brasil produz cerca de 2,6 kg por ano de resíduos eletrônicos por habitante" [Alves et al.]. Devido a suas características próprias, esses resíduos exigem soluções distintas as aplicadas no lixo convencional [LEMOS et al. 2015]. "A maioria das pessoas não sabe que, por trás de equipamentos eletroeletrônicos, existem materiais que podem ser reciclados, basta fazer a retirada e manejo adequado" [Tanaue et al. 2015].

O Reaproveitamento do Lixo Tecnológico (Reltec), é um projeto que propõe, a principio, ações que promovem a conscientização acerca dos resíduos sólidos tecnológicos. Ainda em seu escopo, visa alternativas viáveis ao descarte e reuso desse tipo de material. Para tal, traz em suas metodologias formas de beneficiar o meio ambiente

promovendo ações de coleta e disseminação de informações sobre esse tipo de problema ambiental.

Com o objetivo de mostrar os riscos desse tipo de resíduos, o projeto induz ações de divulgação de consciência ambiental dando enfoque nos sólidos tecnológicos. Em palestras dadas por participantes do projeto, mostravam-se alternativas ao descarte incorreto desses materiais, salientando que o reuso também é viável. Ainda nessas, mostrava-se que esses materiais podem ser produzidos tanto em âmbito doméstico como em grandes indústrias, hospitais e academias, instigando assim, os espectadores a tomar conhecimento dos recursos tecnológicos utilizados em seu cotidiano.

As ações de extensão realizadas também se aliaram a formas de divulgação digital, como páginas em redes sociais. Também foram dadas entrevistas em rádios e entrevistas para TV, como forma de enfatizar a importância do temas nas mídias locais. Ainda, no período de tempo entre 3 a 11 de junho de 2016, 11 pontos de coleta foram distribuídos em 4 cidades, fixados em parceiros e apoiadores do projeto favorecendo a coleta de resíduos tecnológicos. Em especial nos dias 3 e 11 de julho foi feito, em praça pública, um evento de divulgação e fomento ao tema.

Na conclusão do projeto, conseguiu-se arrecadar 2900 quilos de resíduos tecnológicos, Figura 2, nos pontos de coleta, esse material foi repassado a uma empresa de reciclagem. No tocante às ações de conscientização, pode-se ressaltar que com palestras, entrevistas para rádio e TV, páginas em redes sociais, utilizadas no decorrer do projeto, um grande alcance para divulgação do tema foi criado, estimulando-se assim a fomentação do tema em âmbito local e regional, levando em consideração as cidades vizinhas.

Figura 2. Parte do material coletado

2.4. DI nas Escolas

O projeto DI nas Escolas busca disseminar a utilização das ferramentas computacionais no ambiente escolar, promovendo uma maior inclusão digital no âmbito acadêmico. Esse processo acontece por meio da formulação e aplicação de cursos de capacitação, além de suporte técnico e intelectual, disponibilizados às escolas conveniadas ao projeto. O DI nas escolas propõe uma maior interação entre escolas e universidade, utilizando como intermédio, as apresentações sobre o curso para o ensino médio que são realizadas em

escolas públicas e particulares, para que assim, haja um maior interesse dos alunos pela área de computação.

A base para o desenvolvimento do projeto foi palestras, Figura 3, realizadas em escolas públicas e particulares da cidade de Mossoró-RN, e foram direcionadas aos alunos do ensino médio. O conteúdo das palestras era, basicamente, sobre o curso de Ciência da Computação da UERN, desmistificando, assim, estereótipos existentes sobre o mesmo. Logo após, foi mostrado as características dos profissionais da área e, também, o objetivo do curso, que é formar indivíduos qualificados para o exercício das atividades na área de informática. Também foram apresentadas algumas disciplinas da grade curricular, o corpo docente, e os grupos de pesquisa. Ainda houve um questionário que foi respondido pelos alunos, sobre a palestra, para saber se os estudantes já tinham planejado o futuro e para quais áreas pretendem seguir.

Figura 3. Palestra promovida pelo projeto

Por meio da análise do questionário, concluiu-se que os alunos não conheciam o curso devidamente, e que, possivelmente, a causa estava associada a falta de informações sobre a graduação, o que desenvolve uma visão equivocada sobre o curso e os profissionais desta área. Sendo assim, o projeto DI nas Escolas é uma das possíveis soluções para que os alunos do ensino médio conheçam o curso de Ciência da Computação adequadamente, e suas áreas de trabalho.

2.5. Semana Acadêmica de Ciência da Computação

A Semana Acadêmica de Ciência da Computação (SACC) visa propiciar aos participantes a realização de atividades que elevem a qualidade da formação acadêmica, consistindo em um espaço de debate, interação, conhecimento e apresentação do que está sendo desenvolvido no nosso curso.

O evento tem o objetivo de trazer à tona assuntos inerentes aos desafios e possibilidades da computação apresentando a importância do empreendedorismo, oportunizando que alunos e profissionais da área troquem ideias entre si acerca do que será discutido durante a semana. Promove também diálogos com profissionais dos seguintes segmentos: empreendedorismo e experiências no exterior, além disso, mostrar de forma resumida e dinâmica as principais áreas de abrangência que o curso de Ciência da computação oferta. O seu tema varia, sendo sempre assuntos que estejam ligados às áreas que o mercado de trabalho oferece.

A I Semana Acadêmica de Ciência da Computação obteve a presença de alguns profissionais e professores que apresentaram palestras, Figura 4, e mesas redondas variadas, indo desde o tema empreendedorismo, com palestras sobre empresas juniores e mercado de TI na região até programas impulsionadores de formação acadêmica, como o programa Ciências Sem Fronteiras que agrega atributos e conhecimentos que normalmente a graduação em si não oferece. Tendo um público com mais de 55 alunos a I Semana Acadêmica de Ciência Computação obteve uma ótima avaliação do público com base em uma pesquisa realizada no evento, em que 75,6% dos participantes avaliaram o evento como ”Muito Bom”.

Figura 4. Palestra promovida durante o evento sobre intercâmbio

2.6. Treinamento OBI

Incentivar pessoas a usar o computador não só como uma diversão, mas como uma ferramenta para desenvolver habilidades cognitivas é o que caracteriza o pensamento computacional [Andrade et al. 2013]. Várias iniciativas surgem ao redor do mundo com o objetivo de disseminar o pensamento computacional e no Brasil a principal iniciativa é a Olimpíada Brasileira de Informática (OBI). O projeto OBI desenvolvido pelo PETCC acontece em uma parceria com algumas escolas de ensino público e privado do município de Mossoró-RN, com o intuito de capacitar os alunos do ensino fundamental e médio a participarem da OBI.

A metodologia desenvolvida para o treinamento está baseada nas modalidades da OBI: iniciação e programação. Na modalidade iniciação, treinamos os alunos para praticar o raciocínio lógico e desenvolver essa habilidade para ter um pensamento mais exato. Com o uso de ferramentas como o software Scratch e o site code.org que utilizam a forma de programação de movimentação de blocos. Na modalidade programação, é ensinado os alunos à linguagem de programação C, por meio de slides, exercícios e resoluções de provas anteriores.

Nas escolas de ensino público o projeto de treinamento da OBI foi interrompido devido à ausência dos alunos das escolas participantes. Foram feitas duas tentativas de

retorno do projeto, porém o mesmo problema persistiu, o que levou à adoção de outra linguagem de programação na tentativa de resolver o problema.

Já na escola de ensino privado mais da metade dos alunos, participantes das duas modalidades, chegaram a fase final da prova. Inicialmente o cronograma era de 20 semanas, porém este foi estendido, em especial aos alunos que passaram para a fase final, focando na resolução de provas de anos anteriores da OBI em conjunto com os alunos. Na edição de 2016 o treinamento no colégio privado foi realizado somente para a modalidade programação, nesta edição 80% dos alunos chegaram à fase final da prova. Na edição de 2017 ocorreu o treinamento para as modalidades iniciação e programação. Na modalidade iniciação e na modalidade de programação 60% e 50% dos alunos chegaram à fase final, respectivamente.

2.7. Maratona

A maratona de programação é um evento da Sociedade Brasileira de Computação (SBC) que ocorre todos os anos, teve seu início em 1996 e foi criada das competições regionais classificatórias para as finais mundiais do concurso de programação da International Collegiate Programming Contest (ACM). A competição busca incentivar o trabalho em equipe, a criatividade e achar soluções de problemas sob pressão. É realizada com equipes de três alunos universitários e um único computador para solucionar de 8 a 12 problemas do mundo real no prazo de cinco horas. Cada problema vem acompanhado de uma descrição e um exemplo de como os dados deve ser inseridos e suas respectivas respostas. As soluções apresentadas pelas equipes devem ser feitas nas linguagens C, C++, Java ou Python [MARATONA 2017].

Como base para os treinos da maratona, foi utilizado o livro Programming Challenges, onde cada capítulo foi estudado em aproximadamente um mês. A metodologia feita para se estudar cada capítulo foi composta em quatro etapas, uma por semana, na primeira semana havia a apresentação, a segunda semana a resolução dos problemas relacionados ao capítulo apresentado, na terceira semana é realizado uma competição individual ou em equipes e na última semana do ciclo é mostrado as soluções dos problemas abordados na competição.

Com a finalização dos treinamentos, houve uma grande melhoria na capacidade de elaboração de códigos, na abstração de problemas e na criatividade para resolvê-los. Desde a sua criação o PETCC conseguiu enviar em média 3 equipes para participar da seletiva regional, e em 2015 uma dessas equipes conseguiu chegar na final nacional que aconteceu em São Paulo. Junto com as experiências vindas da participação na maratona, percebeu-se a oportunidade de criar o ComPet, uma competição interna para melhorar e aumentar a competitividade e conhecimento para se ter uma maior participação nas futuras edições da competição.

3. Resultados

As atividades desenvolvidas pelo PETCC têm desempenhando um importante papel dentro do curso de Ciência da Computação. Além dos resultados já apresentados na seção anterior, observa-se que o PETCC tem exercido uma função significativa na democratização do curso. As ações como Monitoria, DI nas Escolas, Reltec e Semana Acadêmica de Ciência da Computação, foram incorporadas ao Projeto Pedagógico do Curso e institucionalizadas pelo Departamento de Informática.

Além da institucionalização das atividades, as ações desenvolvidas sempre envolvem a participação de docentes do curso, que atuam no PETCC como colaboradores, mas são os petianos os membros ativos das intervenções. Essa abordagem rompe com o paradigma de que o professor é o responsável pelo curso, trazendo para o aluno a responsabilidade pela sua formação.

4. Conclusões

Entre as diversas dificuldades encontradas no ensino superior, tais como: baixo rendimento, evasão, entre outras. Verifica-se que existe a necessidade de tratar esses problemas e uma das abordagens que tem surtido significativamente é colocar o discente a frente do processo. Nesse sentido, o PETCC tem atuado de forma expressiva no curso de Ciência da Computação tornando o processo de ensino-aprendizagem mais democrático.

Referências

Alves, F. M., Santos, J. A., da Silva, W. G. G., and Pereira, W. G. Um estudo realizado sobre qual o destino dos equipamentos eletrônicos, baterias, pilhas, celulares e computadores na cidade de cacoal/ro.

Andrade, D., Carvalho, T., Silveira, J., Cavalheiro, S., Foss, L., Fleischmann, A. M., Aguiar, M., and Reiser, R. (2013). Proposta de atividades para o desenvolvimento do pensamento computacional no ensino fundamental. In *Anais do Workshop de Informática na Escola*, volume 1, page 169.

BRASIL (2001). Ministério da Educação: programa especial de treinamento – pet.

BRASIL (2006). Ministério da Educação: manual de orientações – pet.

Clezar Júnior, B. (2006). O perfil da infra-estrutura urbana das cidades do litoral norte do rio grande do sul.

Fontes, E. L. G. (2006). *Segurança da Informação: o usuário faz a diferença*. Saraiva.

LEMOS, L. R., dos SANTOS, D. F., and BUENO, P. V. (2015). Lixo tecnológico no brasil e no mundo. *ANAIS-ENCONTRO CIENTÍFICO DE ADMINISTRAÇÃO, ECONOMIA E CONTABILIDADE*, 1(1).

Lins, L. F., FERREIRA, L. M. C., Ferraz, L. V., and CARVALHO, S. d. (2009). A importância da monitoria na formação acadêmica do monitor. *Jornada de ensino, pesquisa e extensão, IX*.

MARATONA (2017). Maratona de programação.

Moi, P. C. P., de Souza, A. P. S., Oliveira, M. M., Faitta, A. C. J., de Rezende, W. B., Moi, G. P., and Freire, F. A. D. L. (2014). Lixo eletrônico: consequências e possíveis soluções. *Connection line*, (7).

Scarparo Haag, G., Kolling, V., Silva, E., Bastos Melo, S. C., and Pinheiro, M. (2008). Contribuições da monitoria no processo ensino-aprendizagem em enfermagem. *Revista Brasileira de Enfermagem*, 61(2).

Tanaue, A. C. B., Bezerra, D. M., Cavalheiro, L., and Pisano, L. C. (2015). Lixo eletrônico: Agravos a saúde e ao meio ambiente. *Ensaios e Ciência: C. Biológicas, Agrárias e da Saúde*, 19(3).

Método baseado nos Referenciais de Formação da SBC para reestruturação de descritivos de disciplinas de Ciência da Computação em conformidade com as DCN de 2016

Alcides Calsavara[1], Ana Paula Gonçalves Serra[2], Francisco de Assis Zampirolli[3], Leandro Silva Galvão de Carvalho[4], Miguel Jonathan[5], Ronaldo Celso Messias Correia[6]

[1]Escola Politécnica, Pontifícia Universidade Católica do Paraná (PUCPR) – Curitiba – PR; [2]Faculdade de Tecnologia e Ciências Exatas, Universidade São Judas Tadeu (USJT) – São Paulo – SP; [3]Centro de Matemática, Computação e Cognição, Universidade Federal do ABC (UFABC) – Santo André – SP; [4]Instituto de Computação, Universidade Federal do Amazonas (UFAM) – Manaus – AM; [5]Departamento de Sistemas e Computação, Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro – RJ; [6]Departamento de Matemática e Computação, Universidade Estadual Paulista (UNESP) – Presidente Prudente – SP

alcides.calsavara@pucpr.br, prof.anapaula@usjt.br, fzampirolli@ufabc.edu.br, galvao@icomp.ufam.edu.br, jonathan@dcc.ufrj.br, ronaldo@fct.unesp.br

Resumo. *Os Referenciais de Formação para o Bacharelado em Ciência da Computação (RF-CC-17) da SBC organizam as competências e habilidades descritas nas Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16) em eixos de formação e também indicam um conjunto de conteúdos associados. Este ensaio apresenta um método, baseado nos RF-CC-17, para elaborar um Mapeamento de Conformidade e Mobilização (MCM), como parte do descritivo de uma disciplina. Como exemplo, o método é aplicado na elaboração de dois descritivos distintos de introdução à programação, um baseado no paradigma imperativo e outro, orientado a objetos. Por fim, discute as vantagens de se usar o método para auxiliar na revisão de projetos pedagógicos de cursos vigentes tal que fiquem em conformidade com as DCN16.*

Abstract. *The SBC Computer Science Curricula 2017 (RF-CC-17) arranges the competences and skills described in the National Curricular Guidelines for Computing Undergraduate Courses (DCN16) into axial competences. Also, it associates content topics to each axial competence. This essay presents a method to elaborate a Mapping of Compliance and Mobilization (MCM) as a part of a discipline program, based on RF-CC-17. As an example, the method is applied in the elaboration of two distinct programs of introduction to programming, one based on the imperative paradigm and another, object-oriented. Finally, it discusses the advantages of using the method to review pedagogical projects of current courses in compliance to the DCN16.*

1. Introdução

Em novembro de 2016, o Ministério da Educação homologou as Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (MEC 2016), simplesmente chamadas de **DCN16**, que estabelecem as normas legais para a organização e o

funcionamento de centenas de cursos brasileiros de graduação da área de computação. As DCN16, em seu Artigo 10, estipulam um prazo de dois anos, a partir da sua publicação, para que cada Instituição de Educação Superior (IES) implante as normas nelas estabelecidas aos alunos ingressantes. Ou seja, todo curso da área de computação deve, obrigatoriamente, revisar o seu Projeto Pedagógico de Curso (PPC) tal que fique em conformidade com as DCN16 até, o mais tardar, o ano letivo de 2019.

As DCN16 elencam o *perfil do egresso* e as *competências* e *habilidades* que um estudante deve adquirir durante a graduação; porém, não especificam quaisquer conteúdos básicos ou tecnológicos. Assim, as DCN16 seguem a tendência pedagógica de definir o que os egressos de um curso devem saber fazer e como devem se comportar na sua vida profissional, ao mesmo tempo que deixa cada curso livre para definir os componentes curriculares (incluindo as disciplinas do curso) e os correspondentes conteúdos, contanto (de acordo com o Artigo 6° das DCN16) que haja consistência com o perfil, as competências e as habilidades especificadas para o egresso.

Atingir a conformidade do PPC com as DCN16 representa tanto uma oportunidade como um desafio para o corpo docente de um curso, em especial para a sua coordenação e o seu Núcleo Docente Estruturante (NDE). Enquanto a oportunidade consiste em poder construir uma matriz curricular sem as limitações e as possíveis distorções decorrentes da rigidez do modelo tradicional baseado em conteúdo, o desafio reside justamente na mudança de paradigma, pois o modelo baseado em competências para construção de matrizes curriculares é, ainda, pouco explorado pelas IES brasileiras.

Por outro lado, a Sociedade Brasileira de Computação (SBC) participou na homologação das DCN16, principalmente na elaboração da sua proposta anos antes (MEC 2012). Por isso, há anos vem promovendo estudos e discussões entre acadêmicos para o avanço do domínio do modelo baseado em competências para ensino e aprendizagem de computação. Em especial, tem organizado grupos de trabalho específicos para cada tipo de curso (Bacharelado em Ciência da Computação, Engenharia de Computação, Sistemas de Informação e Engenharia de Software, bem como Licenciatura em Computação) a fim de revisar os chamados *currículos de referência* segundo esse novo modelo. Como resultado desse longo trabalho, a SBC publicou, em outubro de 2017, os chamados *referenciais de formação* para cursos de graduação em computação (Zorzo et al. 2017). A partir de então, os referenciais de formação passaram a ser a recomendação oficial da SBC para a elaboração do PPC de cada IES, em substituição aos currículos de referência.

Coerentemente, os referenciais de formação baseiam-se fortemente nas DCN16 para fazer recomendações na elaboração de um PPC segundo o paradigma de competências. Incorpora, ainda, contribuições de outros documentos importantes, tais como os antigos currículos de referência da própria SBC (SBC 1999, SBC 2003 e SBC 2005) e o currículo de referência elaborado pela comunidade internacional da computação (ACM/IEEE 2013), além de contribuições advindas da experiência de muitas IES brasileiras. Assim, comparado com as DCN16, os referenciais de formação apresentam uma visão mais holística da área de computação, logo mais adequada para a elaboração do PPC, sem qualquer prejuízo à sua conformidade com as DCN16.

Muito embora os referenciais de formação representem um grande avanço promovido pela SBC e, de fato, facilitem a elaboração do PPC segundo o paradigma de competências, ainda pode-se considerar a criação de instrumentos adicionais para sua maior efetividade. Mais especificamente, os referenciais de formação do Bacharelado em Ciência da Computação (Calsavara et al., 2017), aqui referenciados como RF-CC-17,

associam conteúdos curriculares a cada competência (ou habilidade) presente nas DCN16 de acordo com o contexto em que é requerida, caracterizado como um *eixo de formação*. Estes são definidos por uma "macro-competência" do egresso. Cada competência das DCN16, chamada de *competência derivada*, pode ser requerida em mais de um eixo de formação. Além disso, os conteúdos associados estão, em sua maioria, presentes nos antigos currículos de referência sob o título de *matérias*, sendo, portanto, de fácil compreensão pela comunidade acadêmica. Entretanto, cabe a cada curso definir uma estratégia de como usar toda essa informação na elaboração do PPC, em especial, na descrição dos componentes curriculares do curso.

Neste trabalho, classificado como ensaio, propõe-se um método para auxiliar a elaboração da parte do PPC correspondente à descrição das disciplinas de um curso, a partir dos RF-CC-17. Adota-se aqui o termo *descritivo de disciplina* para designar o texto presente em um PPC que descreve cada disciplina do curso. Naturalmente, além de diferentes designações, o descritivo de disciplina pode assumir muitas formas, dependendo de cada IES, mas invariavelmente inclui uma *ementa*, que tipicamente constitui-se por uma lista de tópicos de estudo.

Devido à heterogeneidade de formas e exigências das IES para o descritivo de disciplina, o produto obtido com a aplicação do método proposto neste trabalho não constitui o descritivo da disciplina propriamente dito. Tampouco, constitui o plano de ensino (ou plano de estudos), que muitas IES desvinculam do descritivo de disciplina para dar mais perenidade ao PPC e, ao mesmo tempo, mais flexibilidade na operacionalização das disciplinas (pode haver um plano de ensino específico para cada turma da disciplina, por exemplo, com detalhamento de cronograma e instrumentos de avaliação). Constitui, sim, um instrumento para auxiliar a coordenação e o NDE de um curso a elaborarem estes dois documentos: descritivo de disciplina e plano de ensino. Tal instrumento, designado *Mapeamento de Conformidade e Mobilização* (MCM), visa garantir que a disciplina está em conformidade com os RF-CC-17 – logo, com as DCN16 – e, ainda, descreve o que o estudante deve realizar na disciplina para adquirir cada uma das competências da DCN vinculadas à disciplina, isto é, que recursos, incluindo conteúdos, o estudante deve mobilizar para adquirir cada competência.

Considerando que os RF-CC-17 são recentes, não foram encontrados artigos relacionados ao seu uso. No entanto, existem esforços em escrever os PPCs de Ciência da Computação seguindo o modelo de competências, dentre eles Rezende, et al. (2004). Por outro lado, o currículo ACM/IEEE (2013), em seu Apêndice C, ilustra o uso do currículo por meio de 83 exemplos de disciplinas ministradas em diversas universidades, a maioria dos EUA. Os exemplos são descritos seguindo um *template* apresentado no próprio Apêndice C. Não é apresentado um método próprio para preenchimento do *template*, mas apenas uma explicação do significado de cada campo.

Este trabalho está organizado como se segue. A Seção 2 descreve o método proposto e o instrumento gerado com a sua aplicação: o MCM. A Seção 3 ilustra a aplicação do método para a disciplina de introdução à programação de duas IES distintas, uma que usa o paradigma de programação imperativo e a outra que usa a orientação a objetos, mostrando que o método proposto contempla as especificidades de cada contexto de aplicação. A Seção 4 faz algumas considerações sobre os benefícios do método proposto. Finalmente, a Seção 5 apresenta conclusões sobre o trabalho.

2. Método

O método parte do princípio que muitas coordenações de curso enfrentam o desafio de reformular um PPC já em vigor em sua instituição, e não construir um PPC inteiramente novo. Portanto, o método adota um procedimento incremental, partindo de uma abordagem baseada em conteúdos para chegar a uma proposta baseada em competências, tal como preconizada pelas DCN16. Em geral, o descritivo de cada disciplina contempla os seguintes campos: Ementa, Objetivos Gerais, Objetivos Específicos, Referências Bibliográficas Básicas e Complementares. Esta seção apresenta um método para se elaborar uma estrutura adicional, o *Mapeamento de Conformidade e Mobilização* (MCM), com o objetivo de auxiliar na revisão do descritivo segundo o paradigma de competências.

2.1. Descrição do Método

O método para elaborar o MCM de uma disciplina consiste nos seguintes passos:

Passo 1: *Seleção dos conteúdos dos RF-CC-17 pertinentes à disciplina*. Essa seleção pode se basear na descrição original da disciplina, normalmente formulada na abordagem conteudista. Por exemplo, para uma disciplina de introdução à programação, os seguintes conteúdos dos RF-CC-17 podem ser selecionados: algoritmos, estruturas de dados, técnicas de programação e programação imperativa. Além deles, podem ser incluídos conteúdos relacionados a competências transversais, tais como língua inglesa e trabalho em equipe.

Passo 2: *Seleção das competências derivadas*. Devem ser selecionadas as competências derivadas entre aquelas dos RF-CC-17 que possuem vínculo com os conteúdos selecionados no Passo 1. Essa seleção deve considerar a disciplina no contexto do curso e pode se basear nos objetivos da disciplina constantes na sua descrição original. Como podem existir competências derivadas com o mesmo nome em diferentes eixos de formação, mas com semânticas específicas para cada eixo, a seleção deve observar se a semântica no eixo é pertinente à disciplina.

Passo 3: *Contribuição da disciplina*. Deve-se explicar como a disciplina contribui para construir no estudante cada competência derivada selecionada no Passo 2. Essa explicação deve focar nas atividades que o estudante desenvolve na disciplina, incluindo os instrumentos e métodos utilizados, e quais são os objetivos específicos dessas atividades que sejam relacionados com a competência derivada. Também deve deixar claro qual o nível cognitivo – criar, aplicar, etc. (Ferraz e Belhot, 2010) – que se pretende desenvolver no estudante. Os RF-CC-17 já recomendam um nível cognitivo para cada competência derivada, mas cada curso pode redefinir esse nível de acordo com os objetivos da disciplina. A contribuição da disciplina para desenvolver uma competência derivada depende de muitos fatores de contexto, tais como perfil do corpo docente, perfil de estudantes, recursos disponíveis na IES, carga horária da disciplina e metodologia de ensino-aprendizagem. O mapeamento dessas relações está fora do escopo deste trabalho.

2.2. Estrutura do MCM

O MCM obtido com a aplicação do método proposto na seção anterior pode ser estruturado conforme mostra a Tabela 1.

Tabela 1. Estrutura do MCM.

Eixo	Competência derivada	Conteúdos	Contribuição da disciplina

Na coluna **Eixo**, são relacionados, em diferentes linhas, alguns dos sete Eixos de Formação contidos nos RF-CC-17, por meio dos quais a disciplina terá o papel de estimular algumas das 25 competências derivadas oriundas das DCN16. Por exemplo, uma primeira disciplina de programação em um curso de Ciência da Computação deve estimular o estudante a *Resolver Problemas* (Eixo 1 dos RF-CC-17). Como **Competência derivada**, podemos ter *Resolver problemas usando ambientes de programação* (terceira competência geral das DCN16). Para cada competência derivada, está associado um nível da Taxonomia de Bloom Revisada (Ferraz e Belhot, 2010). Neste caso, podemos ter *Criar*, indicando que o estudante deverá criar programas simples usando ambientes de programação, como o *Netbeans* ou *Eclipse*, e uma linguagem de programação, como C, Java ou Python. Na coluna **Conteúdos**, serão incluídos alguns conteúdos, como Algoritmos, Técnicas de Programação ou Estruturas de Dados. A maior parte deles foi descrita nos antigos Currículos de Referência (SBC, 1999, 2003 e 2005). Finalmente, a coluna **Contribuição da disciplina** descreve como a disciplina contribui para construir no estudante a competência derivada.

3. Aplicação do método proposto

Para ilustrar a aplicação do método proposto, realizou-se um estudo de caso a partir do descritivo tradicional da disciplina de introdução à programação oferecida a estudantes de primeiro período por duas IES brasileiras, neste trabalho denominadas de α e β , para manter o seu anonimato. As instituições adotam paradigmas de programação distintos nessa disciplina: imperativo e orientado a objetos. Não se pretende aqui definir um descritivo padrão para disciplinas de programação, mas somente ilustrar a aplicação do método especificamente para as duas IES consideradas.

3.1. Abordagem imperativa

Esta seção ilustra a aplicação do método para uma disciplina denominada *Introdução à Programação Imperativa* de uma IES α cujos cursos de computação e de engenharia compartilham essa disciplina em suas matrizes curriculares.

A IES α oferta dois tipos de turmas para a disciplina: *mista* e *exclusiva*. Em uma turma mista, pode haver estudantes de quaisquer cursos, enquanto que em uma turma exclusiva há estudantes de um único curso. Assim, por exemplo, pode haver uma turma composta exclusivamente de estudantes de Ciência da Computação, outra composta exclusivamente por estudantes de Engenharia Mecânica e outra mista, composta por estudantes de Ciência da Computação, de Engenharia Mecânica e de Engenharia Civil.

Independentemente do tipo de turma, o descritivo da disciplina é idêntico em todos os PPCs dos cursos, mas deve haver um plano de ensino específico para cada turma. Por isso, o descritivo da disciplina deve ser genérico o suficiente para se aplicar a todos os estudantes, independentemente do curso, mas também deve ser suficientemente preciso e detalhado a fim de permitir a derivação de um plano de ensino que se ajuste ao perfil dos estudantes de cada turma. Por exemplo, o plano de ensino para a turma exclusiva da Ciência da Computação pode estabelecer que a linguagem de programação C deve ser usada na disciplina, enquanto que o plano de ensino de uma turma mista pode estabelecer que a linguagem Python deve ser usada.

Além disso, por uma orientação geral da IES α, as disciplinas devem, sempre que aplicável, promover a multidisciplinaridade. Assim, embora a disciplina seja da área de computação, o seu descritivo estabelece uma forte interação com outras áreas, mais

especificamente por meio da proposição de atividades para os estudantes que envolvam a resolução de problemas de outras áreas, em especial dos diversos tipos de engenharia. Essa multidisciplinaridade é exigida não apenas para as turmas mistas, onde naturalmente já ocorre, mas também para as turmas exclusivas. Por isso, o plano de ensino de uma turma exclusiva deve prever atividades que exijam interação com estudantes (e, possivelmente, professores) de outros cursos.

A Tabela 2 mostra a parte do descritivo original da disciplina que contém informações úteis para a aplicação do método. O MCM obtido para a disciplina é composto pelas informações das Tabelas 3 e 4. A partir das informações da Tabela 1, no Passo 1, selecionam-se os seguintes conteúdos dentre os listados no RF-CC-17: Algoritmos, Estruturas de Dados, Técnicas de Programação e Programação Imperativa. Com essa lista de conteúdos, no Passo 2, são selecionadas seis competências derivadas (e correspondentes eixos) que aparecem vinculadas a esses conteúdos no RF-CC-17, conforme mostra a Tabela 3. Algumas competências derivadas não foram selecionadas, mesmo estando vinculadas aos conteúdos, pois não se adequam ao contexto da disciplina. Por fim, a Tabela 4 contém a contribuição da disciplina no desenvolvimento de cada competência derivada, o que, inclusive, justifica a própria seleção feita. Observa-se que o texto se concentra em explicar o que o estudante realiza na disciplina, isto é, que recursos, incluindo tópicos de estudo, mobiliza para adquirir a competência.

Tabela 2. Parte do descritivo original da disciplina Introdução à Programação Imperativa da IES α (auxilia em todos os passos do método).

NOME DISCIPLINA: Introdução à Programação Imperativa
Ementa: Conceitos de algoritmos e programação estruturada. Tipos de dados, constantes, variáveis e atribuição. Pseudolinguagem e fluxogramas. Estruturas de seleção. Estruturas de repetição. Vetores. Matrizes. Funções. Entrada e saída de dados. Leitura e escrita em arquivos textos. Depuração de programas. Melhores práticas de programação.
Objetivo Geral: Criar algoritmos básicos para solucionar problemas de natureza técnico-científica e os implemente em uma linguagem de programação.
Metodologia: Apresentação fundamentos sobre manipulação e tratamento de dados e informações, utilizando explicações e experimentações dos conceitos de lógica de programação.

Tabela 3. Eixos e competências derivadas selecionadas para a disciplina Introdução à Programação Imperativa da IES α e conteúdos associados (Passo 2 do método).

Eixo	**Competência derivada**	**Conteúdos**
Resolução de Problemas	C.1.1. *Identificar problemas que tenham solução algorítmica* [Avaliar]	Algoritmos
	C.1.3. *Resolver problemas usando ambientes de programação* [Criar]	Algoritmos; Técnicas de Programação; Estruturas de Dados
	C.1.5. *Reconhecer a importância do pensamento computacional no cotidiano e sua aplicação em circunstâncias apropriadas e em domínios diversos* [Aplicar]	Algoritmos; Estruturas de Dados
Desenvolvimento de Sistemas	C.2.1. *Resolver problemas usando ambientes de programação* [Criar]	Algoritmos; Programação Imperativa
Desenvolvimento de Projetos	C.3.7. *Reconhecer a importância do pensamento computacional no cotidiano e sua aplicação em circunstâncias apropriadas e em domínios diversos* [Aplicar]	Algoritmos; Estruturas de Dados
Aprendizado Contínuo e Autônomo	C.6.6. *Compreender os fatos essenciais, os conceitos, os princípios e as teorias relacionadas à Ciência da Computação para o desenvolvimento de software e hardware e suas aplicações* [Avaliar]	Algoritmos

Tabela 4. Contribuição da disciplina Introdução à Programação Imperativa da IES α para cada Competência Derivada (CD) selecionada (Passo 3 do método).

CD	Contribuição da disciplina
C.1.1	O estudante experimenta a aplicação de alguns algoritmos simples em problemas de domínios diversos, incluindo algoritmos de ordenação e busca em conjuntos de dados armazenados em memória e em arquivo. Com essa experiência, o estudante passa a compreender o potencial da computação na resolução de problemas que envolvam o tratamento de grandes volumes de dados e pode avaliar a possibilidade de aplicação desses algoritmos em outros contextos. Por exemplo, dado um algoritmo simples de ordenação, o estudante deve ser capaz de simular (em papel) a sua execução. Além disso, o estudante deve ser capaz de escrever (em pseudocódigo e fluxograma) a solução de problemas simples, como ordenar uma lista de estudantes pelo conceito final de uma disciplina, ou exibir os nomes dos estudantes em ordem alfabética.
C.1.3	O estudante resolve problemas simples que envolvam o tratamento de dados numéricos e textuais (*strings*) por meio da implementação de programas em linguagem imperativa, com base em algoritmos que empregam comandos de atribuição, desvio, seleção e repetição, variáveis e constantes de tipos primitivos (numéricos, textuais e booleanos) e estruturas de dados de baixa complexidade (vetores e matrizes). Os algoritmos podem ser selecionados da literatura ou desenhados especificamente para os problemas propostos. Com isso, o estudante passa a ter o domínio básico de uma linguagem e de um ambiente de programação, que constituem a ferramenta prática mais fundamental da computação para a resolução de problemas do mundo real.
C1.5	Idem a C.1.1, com ênfase na aplicação da computação em diversas áreas do conhecimento, ou seja, o estudante é apresentado a problemas típicos de outras áreas, tal que compreenda o essencial do processo de transposição de conhecimentos da computação para outras áreas. Por exemplo, o estudante deve ser capaz de transformar uma imagem colorida na correspondente imagem em preto e branco, ambas representadas por matrizes. Este é um primeiro passo para análise de imagens mais complexas, como imagens de satélite, microscópicas do tecido humano, do espaço, etc. Esta análise também pode ser útil em robótica, para veículos autônomos, por exemplo. O estudante deve ser capaz de desenvolver algoritmos simples para automação de casas, fábricas, escritórios, etc, quando os problemas possam ser resolvidos usando apenas a lógica de programação.
C.2.1	O estudante constrói programas em linguagem imperativa com base em algoritmos especificados por meio de pseudocódigo, fluxograma ou formulação matemática. A complexidade dos programas propostos aumenta gradativamente ao longo do período letivo, tal que, a partir de certo ponto, os programas sejam, necessariamente, estruturados em funções e procedimentos. Finalmente, os programas são validados pelo estudante, inicialmente seguindo um plano de testes proposto e, posteriormente, a partir de um plano de testes elaborado pelo próprio estudante. Dessa forma, o estudante experimenta três etapas fundamentais do ciclo de desenvolvimento de sistemas de software: a especificação, a implementação e a validação.
C.3.7	Idem a C.1.5, incluindo o desenvolvimento de um projeto em outra área do conhecimento, a fim de permitir que estudante vivencie a prática de transposição de conhecimentos da computação para outra área.
C.6.6	O estudante realiza estudos sobre algoritmos simples de ordenação e busca por meio de consultas a livros e autores clássicos dessa área de estudo da computação, além de analisar artigos científicos sobre o assunto, em especial com avaliações e propostas de melhorias de algoritmos. Assim, o estudante toma conhecimento dos meios de publicação de conhecimentos científicos que o auxiliarão no restante do curso e, futuramente, na sua carreira profissional.

3.2. Abordagem orientada a objetos

Esta seção ilustra a aplicação do método para uma disciplina denominada *Programação Orientada a Objetos* de uma IES β na qual essa disciplina é oferecida aos estudantes dos cursos de Ciência da Computação e Sistemas de Informação, sendo adotado um único descritivo e um único plano de ensino para ambos os cursos. Uma mesma turma pode ser composta por somente estudantes de Ciência da Computação, ou por estudantes de Ciência da Computação e Sistemas de Informação, em uma configuração mista. Para ambos os cursos, essa é uma disciplina de primeiro semestre e a primeira disciplina de programação. A IES β usa a orientação objetos como primeiro paradigma de programação, não havendo nenhuma outra disciplina anterior de abordagem estruturada.

Parte do descritivo original da disciplina é mostrada na Tabela 5. Aplicando-se o Passo 1, são selecionados os seguintes conteúdos: Algoritmos; Técnicas de Programação; e Programação Orientada a Objetos. A Tabela 6 apresenta o resultado do Passo 2,

selecionando as competências derivadas. A Tabela 7 completa o MCM, descrevendo como cada competência derivada contribui para a formação do estudante (aplicação do Passo 3).

Tabela 5. Parte do descritivo original da disciplina Programação Orientada a Objetos da IES β (auxilia em todos os passos do método).

NOME DISCIPLINA: Programação Orientada a Objetos
Ementa: Conceitos básicos de orientação a objetos (classe, objeto, atributos, métodos, encapsulamento). Estruturas básicas de programação orientada a objetos. Tipos de dados, constantes, variáveis e atribuição. Estrutura de seleção. Estruturas de repetição. Vetores. Matrizes. Prática de desenvolvimento de algoritmos e programação orientada a objetos. Interação entre classes por relacionamento de associação. Classe de interface gráfica. Classe de negócio. Classes persistentes.
Objetivo Geral: Criar algoritmos orientados a objetos para exercitar a abstração e a capacidade de resolução de problemas computacionais com soluções de programas orientados a objetos.
Metodologia: Apresentação dos conceitos de orientação a objetos, por meio de explicação, experimentação dos conceitos, utilizando interface gráfica e interação com várias classes, por meio de técnicas colaborativas de aprendizado.

Tabela 6. Eixos e competências derivadas selecionadas para a disciplina Programação Orientada a Objetos da IES β e conteúdos associados (Passo 2 do método).

Eixo	**Competência derivada**	**Conteúdos**
Resolução de Problemas	C.1.1. Identificar problemas que tenham solução algorítmica [Avaliar]	Algoritmos
	C.1.3. Resolver problemas usando ambientes de programação [Criar]	Algoritmos; Técnicas de Programação
Desenvolvimento de Sistemas	C.2.1. Resolver problemas usando ambientes de programação [Criar]	Algoritmos; Programação Orientada a Objetos
Aprendizado Contínuo e Autônomo	C.6.6. Compreender os fatos essenciais, os conceitos, os princípios e as teorias relacionadas à Ciência da Computação para o desenvolvimento de software e hardware e suas aplicações [Avaliar]	Algoritmos

4. Discussão sobre o método proposto

De um lado, as DCN16 relacionam 25 competências e habilidades mínimas que os cursos de Ciência da Computação devem desenvolver em seus egressos. De outro, o parecer CNE/CES Nº 136/2012 (MEC, 2012), que acompanha as diretrizes, elenca um vasto conjunto de conteúdos curriculares a serem ministrados pelos cursos. Essas duas listagens são apresentadas de forma desconexa, em documentos diferentes. Assim, para as coordenações de curso e NDEs que desejam atualizar seus PPCs em conformidade com as DCN16, a principal vantagem de utilizar os RF-CC-17 e o método aqui proposto é explicitar a ligação entre competências e conteúdos curriculares.

Essa ligação pode ser realizada de forma diferente em cada IES. Por exemplo, a IES α decidiu desenvolver a competência C.1.5, diferentemente da IES β. Isso não implica que a IES β esteja em falta com as DCN, desde que ela desenvolva a competência derivada em outra disciplina da matriz curricular.

Além disso, a listagem de competências e habilidades nas DCN16, divididas entre gerais e específicas, apresenta sobreposições. Por outro lado, os RF-CC-17 agrupam essas competências e habilidades em sete eixos de formação. Dessa forma, as coordenações de curso e NDEs, ao usarem o método de geração do MCM, contam com um roteiro e um estudo de caso para facilitar a compreensão das diretrizes, a fim de elaborar um descritivo de disciplina orientado a competências. De qualquer maneira, o método proposto não dispensa a leitura e o conhecimento das DCN16 e dos RF-CC-17.

Tabela 7. Contribuição da disciplina Programação Orientada a Objetos da IES β para cada Competência Derivada (CD) selecionada (Passo 3 do método).

CD	Contribuição da disciplina
C.1.1	O estudante modela classes que simulam objetos do mundo real, e identifica métodos que podem ser implementados por meio de algoritmos simples. O estudante desenvolve a capacidade de abstração que o habilita a compreender como mapear problemas do mundo real no computador e a construir soluções orientadas a objetos computacionais eficazes.
C.1.3	O estudante implementa programas usando uma linguagem orientada a objetos em um ambiente de programação simples, sem muitos recursos de *plugin* e/ou interface gráfica. Na implementação de cada classe, o estudante emprega conceito de encapsulamento, técnicas de representação de dados (variáveis, constantes, tipos primitivos e estruturas simples, como vetores e matrizes) e comandos de atribuição, desvio, seleção e repetição. Com isso, o estudante passa a ter domínio básico de técnicas de programação e de um ambiente de programação, que é a ferramenta prática mais fundamental da computação para a resolução de problemas do mundo real.
C.2.1	O estudante constrói programas em linguagem orientada a objetos com base em um problema computacional, implementando classes de interface gráfica, classes de negócio e acesso a banco de dados, além de realizar testes unitários e testes de validação e verificação. Dessa forma, o estudante experimenta quatro etapas fundamentais do ciclo de desenvolvimento de sistemas de software, a saber: a especificação (problema a ser resolvido), projeto (identificação das classes e suas responsabilidades utilizando o princípio de arquitetura em camadas), implementação, e testes unitários e de verificação e validação.
C.6.6	O estudante realiza estudos sobre programação orientada a objetos por meio de consultas a livros e autores clássicos dessa área de estudo da computação, além de analisar artigos científicos sobre o assunto, em especial sobre boas práticas de programação orientação a objetos e plataformas que podem ser aplicadas. Assim, o estudante toma conhecimento dos meios de publicação de conhecimentos científicos que o auxiliarão no restante do curso e, futuramente, na sua carreira profissional. A metodologia de ensino apoia o aprendizado contínuo, por meio de aulas práticas com técnicas colaborativas de aprendizado professor-estudante e estudantes-estudantes. O objetivo é ensinar os estudantes a escreverem código de programação orientado a objetos de uma forma mais prática, dinâmica e colaborativa, com maior facilidade e qualidade, incentivando o entendimento de conceitos e técnicas de orientação a objetos, o gosto pela programação e exercitando o trabalho em equipe. O estudante participa ativamente da escrita do código, todos os alunos interagem e compartilham suas dúvidas, para isso, é utilizada uma técnica conhecida como *Coding Dojo*.

5. Conclusão

O Mapeamento de Conformidade e Mobilização (MCM), obtido com a aplicação do método proposto, constitui uma base confiável para elaborar os descritivos de disciplinas de um PPC em conformidade com as DCN16. Isso foi verificado por meio da aplicação do método para a disciplina de introdução à programação em duas IES brasileiras. O MCM obtido para cada IES seleciona algumas das competências especificadas nos RF-CC-17 pertinentes à disciplina e descreve como os recursos vinculados, incluindo os conteúdos ministrados, são mobilizados pelo estudante para adquirir cada competência.

Apesar de o estudo de caso contemplar somente a disciplina de introdução à programação, o método proposto é genérico e pode ser utilizado para reestruturar as demais disciplinas de PPCs vigentes. Além disso, o MCM auxilia o docente responsável pela disciplina a planejar, organizar e definir as atividades que serão desenvolvidas pelo discente, isto é, a elaborar o plano de ensino.

O método proposto agiliza a reestruturação dos PPCs, principalmente para coordenações e NDEs pouco experientes com a abordagem baseada em competências. Dessa forma, contribui para que as IES cumpram o prazo legal estabelecido pelas DCN16, ou seja, para os estudantes ingressantes no ano letivo de 2019.

Os MCMs produzidos neste ensaio estão disponíveis em goo.gl/YgHi7h. Como trabalho futuro, convidamos a comunidade a utilizar o método proposto para expandir esse repositório, produzindo o MCM para outras disciplinas de Computação ministradas aos

cursos de graduação. Esse trabalho colaborativo pode proporcionar ainda mais agilidade no processo de reestruturação dos PPCs, bem como pode contribuir para a sua melhor qualidade e maior conformidade com as DCN16.

Referências

ACM/IEEE (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Final Report. ACM, New York, NY, USA. 2013. Disponível em: http://dx.doi.org/10.1145/2534860. Último acesso em: 16/03/2018.

Calsavara, A., Serra, A. P. G., Zampirolli, F. A., Carvalho, L. S. G., Jonathan, M., Correia, R. C. M. (2017). Referenciais de Formação: Bacharelado em Ciência da Computação. In: Zorzo, A. F., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R. M., Correia, R., Martins, S. (Org.). Referenciais de Formação para os Cursos de Graduação em Computação, 2017, p. 9-39.

Ferraz, A. P. C. M., Belhot, R. V. (2010). Taxonomia de Bloom: revisão teórica e apresentação das adequações do instrumento para definição de objetivos instrucionais. Gest. Prod., São Carlos, 17(2), 421-431.

MEC (2012). Proposta de Diretrizes Curriculares Nacionais para os cursos de graduação em Computação. Disponível em: https://goo.gl/esgE8f. Parecer CNE/CES nº 136/2012, aprovado em 8 de março de 2012. Último acesso em: 16/03/2018.

MEC (2016). Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16). Disponível em: https://goo.gl/35CmzT. Resolução CNE/CES nº 5, de 16 de novembro de 2016. Último acesso em: 16/03/2018.

Rezende, L., Segre, L. M., Campos, G. H. (2004). O modelo das competências e as implicações para o currículo do curso de ciência da computação. In Anais do XXIV Congresso da Sociedade Brasileira de Computação (WEI). Salvador (Vol. 2).

SBC (1999). Currículo de Referência da SBC para cursos de Graduação em Computação (CR99). http://lad.dsc.ufcg.edu.br/ec/cr99.pdf. Último acesso em: 16/03/2018.

SBC (2003). Currículo de Referência da SBC para cursos de Graduação em Computação e Informática (CR03). https://goo.gl/FXncde. Último acesso em: 16/03/2018.

SBC (2005). Currículo de Referência da SBC para cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia da Computação (CR05). https://goo.gl/VL7dcD. Último acesso em: 16/03/2018.

Zorzo, A. F., Nunes, D., Matos, E., Steinmacher, I., Leite, J., Araujo, R. M., Correia, R., Martins, S. (2017). Referenciais de Formação para os Cursos de Graduação em Computação. Sociedade Brasileira de Computação (SBC). 153p, 2017. ISBN 978-85-7669-424-3.

Criando aplicativos sobre o descarte adequado de lixo: experiências utilizando uma abordagem temática em um clube de programação

Andrea S. Charão[1], **Ana Luísa V. Solórzano**[2], **Rafael G. Trindade**[1]

[1] Departamento de Linguagens e Sistemas de Computação

[2]Curso de Ciência da Computação
Universidade Federal de Santa Maria
Santa Maria, RS, Brasil

{andrea, alsolorzano, rtrindade}@inf.ufsm.br

Abstract. *Programming clubs are an effective approach to bring computing fundamentals to school-age students. As non-formal, after-school activities, clubs face the challenge of creating stimulating learning environments with meaning to the participants. This paper presents an experience that embeded a social responsibility theme across multiple projects in a programming club, as a meaningful way to engage students while strengthening their environmental awareness. During 9 workshops mentored by undergraduate students, the club participants created mobile applications using MIT App Inventor, around the theme of proper waste disposal. The article discusses some key decisions and lessons learned that may be helpful in replicating the experience.*

Resumo. *Clubes de programação são abordagens efetivas para disseminar fundamentos de computação ao público da educação básica. Por se tratarem de atividades extra-classe, de ensino não-formal, clubes enfrentam o desafio de criar ambientes de aprendizado estimulantes e com significado aos participantes. Neste trabalho, apresenta-se uma experiência que inseriu um tema de responsabilidade social em atividades de um clube de programação, envolvendo os participantes em uma temática de conscientização ambiental. Ao longo de 9 oficinas tutoradas por universitários, os participantes do clube criaram aplicativos usando MIT App Inventor, em torno da temática do descarte adequado de lixo. O artigo discute decisões-chave e lições aprendidas que favorecem a replicação da experiência.*

1. Introdução

Clubes de programação são iniciativas notáveis, que fazem parte de uma tendência mundial de disseminação de noções de Computação a estudantes em idade escolar. Em contraste com ambientes de ensino tradicionais, os clubes desenvolvem o ensino não-formal, que é não-obrigatório, mais adaptável e centrado nos alunos [Hamadache 1991, Eshach 2007]. Clubes têm sido empregados não apenas para envolver os estudantes em uma intersecção entre educação e recreação, mas também para complementar a educação formal [Sahin 2013, Gottfried and Williams 2013].

Relatos de experiência de clubes de programação e iniciativas similares são bastante diversificados, variando conforme a idade do público-alvo, objetivos de aprendizado, atividades e ferramentas. Existem clubes voltados ao ensino fundamental [Smith et al. 2014] e ao ensino médio [Weintrop and Wilensky 2015]. As atividades variam desde exercícios de computação desplugada e quebra-cabeças lógicos [Bellettini et al. 2014] até o desenvolvimento de jogos e animações [Lakanen et al. 2014]. Em alguns casos, as atividades são feitas para serem reutilizadas [Dee et al. 2017].

Mesmo com recursos reutilizáveis, o planejamento de atividades extra-classe pode ser desafiador [Giaya et al. 2015], visto que as experiências são únicas, devido às expectativas individuais e heterogêneas dos estudantes e tutores. Mesmo quando os tutores propõem atividades que estimulam a criatividade e levam em conta as preferências pessoais, permanece o desafio de agregar significado às criações feitas utilizando programação [Lye and Koh 2014]. Os estudantes tendem a se engajar e aprender mais construindo programas que fazem sentido em suas vidas diárias [Kafai and Resnick 1996].

Em muitos casos, os autores exploram abordagens temáticas para atingir o objetivo de dar significância ao processo de aprendizagem [Tessier and Tessier 2015]. Nessa abordagem, um tema específico conecta várias atividades, unidades de ensino ou disciplinas em um currículo completo. Em sua maioria, as revisões de literatura sobre abordagens temáticas se concentram na educação formal, e discutem as vantagens e dificuldades de implementar a aprendizagem temática em sala de aula [Tessier and Tessier 2015, Retnawati et al. 2017].

No presente trabalho, apresenta-se uma experiência na qual um clube de programação incorporou um tema de responsabilidade social em vários projetos extra-classe, de modo a envolver os participantes em uma temática de conscientização ambiental, incentivando o descarte adequado de lixo. Esta iniciativa integra um programa de extensão universitária iniciado em 2014, envolvendo professores, funcionários e alunos de graduação em Ciência da Computação que participam como tutores voluntários. A experiência de um semestre envolveu alunos de 11 a 17 anos de 6º ao 9º ano de uma escola localizada em um bairro socialmente desfavorecido, na cidade de Santa Maria, RS.

2. Trabalhos relacionados

Dentre as iniciativas atuais para ensinar programação a iniciantes, em larga escala, destacam-se Code Club[1] e Code.org[2]. O Code Club propõe atividades e guias para que jovens de 9 a 13 anos aprendam programação enquanto constroem jogos, animações e sites, utilizando Scratch, HTML, CSS e Python, mas sem opções para criar aplicativos móveis. O Code.org oferece diversos recursos, com diferentes níveis de dificuldade e para diferentes grupos de idade. Essa iniciativa disponibiliza atividades e tutoriais, popularizados no evento Hora do Código[3], e planos de aula que podem ser usados em intervenções mais longas. Alguns tutoriais de Code.org focam na criação de aplicativos móveis com MIT App Inventor, Thunkable, App Lab ou CodeHS.

[1] http://codeclub.org
[2] http://code.org
[3] http://hourofcode.com

Lakanen et al. [Lakanen et al. 2014] descrevem experiências de cinco anos oferecendo um curso de verão (25 horas, 5 dias), inserido em um programa de extensão universitária na Finlândia. Ele foi direcionado a estudantes do ensino fundamental e médio (12-18 anos) e focou no desenvolvimento de jogos usando C# e Microsoft XNA. Bellettini et al. [Bellettini et al. 2014] apresentam 4 oficinas interativas (2 horas cada) para estudantes (10-17 anos) na Itália. As oficinas incluíram atividades desplugadas, seguidas de um desafio de programação usando o Scratch.

Sullivan et al. [Sullivan et al. 2015] relatam sobre um clube de programação voltado a meninas do ensino médio (12-17 anos) na Irlanda. Em intervenções de 20 horas, as alunas fizeram atividades de programação e de desenvolvimento do pensamento computacional usando os jogos *on-line* do Blockly, CS-Unplugged, Scratch, Kinect2Scratch e MIT App Inventor. Meyer e Batzner [Meyer and Batzner 2016] descrevem um programa para jovens (9-10 anos) na Alemanha criarem jogos com Scratch, durante uma intervenção de 18 horas (2 horas/semana). Todas essas iniciativas enfatizam a programação como uma atividade divertida e criativa, mas nenhuma integra alguma temática às criações.

O relato mais próximo ao presente trabalho foi o de Ni et al. [Ni et al. 2016], que descreve um acampamento de verão em que alunos do ensino médio nos EUA criaram aplicativos móveis para fins socialmente benéficos. Seus resultados reforçam a ideia de que o foco em temas socialmente relevantes é uma alternativa para motivar os alunos ao mesmo tempo em que aprendem programação. Eles mencionam alguns aplicativos informativos desenvolvidos para abordar questões de comunidades (por exemplo, mercados de agricultores locais ou informações sobre reciclagem), mas não parecem explorar um mesmo tema em vários projetos, como proposto no presente trabalho.

3. Clube de programação

O clube de programação integra um programa de extensão universitária iniciado em 2014, envolvendo professores, funcionários e alunos de graduação em Ciência da Computação, que participam como tutores voluntários. Nesse programa, explora-se a educação não-formal, visando proporcionar oportunidades para estudantes em idade escolar se engajarem em atividades de Computação, com ênfase em utilizar a programação como uma ferramenta criativa. Nos últimos anos, foram beneficiados estudantes de instituições públicas e privadas, de ensino fundamental e médio, localizadas em Santa Maria, RS.

Em experiências anteriores, foram utilizadas diferentes atividades e ferramentas com estudantes de variadas idades, de escolas distintas ou de uma mesma escola. Geralmente, as atividades são realizadas no formato de oficinas que duram 20 horas, distribuídas em encontros semanais de 2 a 4 horas, em laboratórios de informática da universidade ou nas escolas.

As atividades escolhidas envolvem rogramação baseada em blocos (usando Scratch, Blockly, tutoriais da Hora do Código ou MIT App Inventor) ou programação com linguagens textuais (Python ou Javascript), dependendo do público participante. Em menor escala, já foram utilizados jogos como Lightbot, Code Combat e Code Hunt, e atividades de computação desplugada envolvendo problemas lógicos para treinamento de participantes na Olimpíada Brasileira de Informática (OBI). Nos parágrafos seguintes, apresenta-se nossa experiência mais notável, pois envolveu uma abordagem diferente, temática, que contrastou com as anteriores.

3.1. Escola e estudantes

Em 2016, buscando expandir as atividades do programa para mais escolas, alcançou-se uma escola particular localizada em um bairro socialmente desfavorecido, focada em programas filantrópicos e comunitários no ensino primário e secundário. A maioria de seus estudantes recebe bolsas que cobrem taxas escolares, materiais didáticos e alimentação/uniforme. A escola preocupa-se com a inclusão digital e com questões ambientais, tendo um histórico de atividades extra-curriculares orientadas a *hardware*, por exemplo, desmontando e remontando computadores descartados, e introduzindo robótica com kits de baixo custo. Para orientar essas atividades, a escola emprega alguns professores com formação em informática e eletrônica.

Após apresentar o programa à escola no final de 2016, decidiu-se propor a eles duas atividades complementares durante 2017: treinamentos para a OBI (no primeiro semestre) e oficinas de criação de aplicativos móveis (segundo semestre). Esses assuntos atraíram a atenção dos professores da escola, que ajudaram a divulgar informações sobre o programa e a reunir estudantes interessados em atividades extra-classe.

Assim, formou-se um time de 21 alunos com idades entre 11 e 17 anos, de 6º à 9º ano, sendo que alguns já haviam participado de oficinas de robótica. A maioria das atividades ocorreram em um laboratório da universidade, com custos de transporte compartilhados entre a escola e a universidade, sempre com pelo menos um professor da escola junto aos estudantes.

3.2. Abordagem temática

Antes das oficinas, foi proposto aos professores da escola que trocassem ideias com seus alunos, em busca de um tema para os projetos de aplicativos. Obteve-se retorno de professores envolvidos em um curso chamado “Protagonismo Social”, oferecido a todas as séries, que engloba assuntos relacionados a transformações sociais e comunitárias. No bairro da escola, há problemas ambientais relacionados ao descarte inadequado de lixo, assim, esse tema de grande relevância foi o selecionado para os projetos.

Em seguida, os tutores de graduação foram desafiados a reunir ideias viáveis e envolventes de aplicativos móveis sobre o tema. Essa abordagem contrasta com outras experiências [Ni et al. 2016] em que os estudantes decidiram o que criar. Essa escolha foi motivada por duas razões observadas em experiências anteriores: (i) Os estudantes geralmente não se sentem à vontade para sugerir suas ideias verbalmente, mesmo quando incentivados nessa direção. Acredita-se que é melhor guiá-los no início das atividades, permitindo que se expressem através de suas customizações. (ii) Ter tutores envolvidos em atividades criativas pode impulsionar sua motivação, gerando impacto positivo em suas atitudes de tutoria.

3.3. Atividades desenvolvidas

As oficinas ocorreram em 9 encontros, de duas horas cada, durante quatro meses no segundo semestre. As atividades foram divididas em duas fases: tutoriais introdutórios e projetos temáticos. Em todos os encontros, contou-se com pelo menos 4 tutores de um total de 8 graduandos voluntários em 2017, que auxiliaram os estudantes nas atividades. Os estudantes trabalharam individualmente ou em duplas, sendo que o laboratório da universidade tinha cerca de 30 computadores disponíveis.

3.3.1. Tutoriais introdutórios

No primeiro encontro foi proposta uma introdução aos fundamentos de Computação usando o tutorial da Hora do Código chamado Labirinto Clássico [4]. Experiências anteriores mostraram que esse primeiro contato com um ambiente de programação baseado em blocos é uma ponte eficaz para o MIT App Inventor. Os estudantes que completaram rapidamente o primeiro tutorial foram incentivados a escolher outros da Hora do Código que utilizassem blocos.

No segundo encontro, apresentou-se o MIT App Inventor explicando suas áreas de edição e propondo a criação de um aplicativo simples, a partir do zero. Esse aplicativo consistiu em uma imagem que, quando clicada, mudava para outra e tocava um áudio.

No terceiro encontro, foi proposta a criação de um aplicativo "Quiz" (jogo de perguntas e respostas), usando variáveis, estruturas de dados (listas), declarações condicionais e funções. Todos esses assuntos foram revisados e detalhados posteriormente nos outros projetos. Como parte dos tutoriais introdutórios, esse aplicativo foi atrativo para os estudantes, que puderam escolher o tema e as questões do seu jogo.

A partir desse encontro, foram propostos projetos base, como mostrado na Figura 1). Com a ajuda dos tutores, esperou-se que os alunos entendessem os blocos já existentes e inferissem quais estavam faltando para terminar o aplicativo.

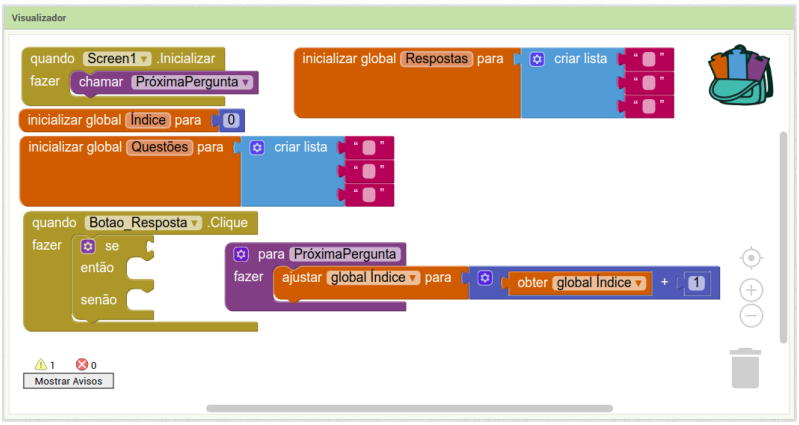


Figura 1. Exemplo da área de desenvolvimento de um projeto base usado

3.3.2. Projetos temáticos

As ideias dos projetos foram desenvolvidas gradualmente. Os tutores utilizaram o tempo entre encontros subsequentes para desenvolver suas ideias no MIT App Inventor antes de as propor nas oficinas, e para discutir sobre observações feitas quanto ao progresso dos estudantes.

Nos encontros temáticos, os tutores primeiramente apresentavam o cronograma do dia. Após, explicavam os conceitos de Ciência da Computação que seriam utilizados e mostravam uma prévia do app proposto, sendo trabalhado um app temático por encontro.

[4]https://studio.code.org/hoc/1

(a) Derrote os Monstros do Lixo (b) Locais de Reciclagem (c) Acerte a Lixeira! (d) Guia de Descarte de Lixo

Figura 2. Tela inicial dos quatro projetos sobre descarte de lixo, executando no emulador do AI2

Em seguida, era fornecido um *link* para *download* do projeto base e de seus arquivos de mídia (imagens e sons).

3.3.3. Recursos de apoio

Além do apoio dos tutores e dos projetos base, foram oferecidos recursos adicionais aos estudantes, para incentivá-los a completarem seus projetos: (i) *Guia do AI2:* mesmo havendo livros abrangentes sobre o MIT App Inventor [5], não encontrou-se um que fosse adequado à nossa experiência, principalmente por serem em outras línguas. Assim, decidiu-se criar um breve Guia do AI2 em português brasileiro, abordando uma introdução ao ambiente e aos nossos aplicativos introdutórios. As versões impressa e digital do guia estavam disponíveis durante as atividades. (ii) *Cheat Sheets:* foram criados *cheat sheets*, pequenas folhas com dicas para os ajudar com as perguntas mais frequentes: como fazer *login* na plataforma, conceitos básicos de computação e instruções para a conexão com o emulador. (iii) *Posts Motivacionais:* entre encontros, usou-se a página do programa no Facebook para postar fotos de atividades e desafios, premiados com camisetas. Ela também serviu como meio de comunicação com os professores, estudantes e seus pais.

4. Projetos temáticos

Foram desenvolvidos quatro aplicativos (Figura 2) que abordam a consciência ambiental, incentivando o descarte adequado do lixo.

4.1. Derrote os Monstros do Lixo

Esse app é um jogo (Fig. 2a) cujo objetivo é derrotar três monstros do lixo, clicando neles até que seus pontos de vida acabem e eles desapareçam. Ele usa uma tela, que mostra a imagem de um monstro e sua respectiva pontuação, e no final mostra uma imagem de parabéns.

[5]http://appinventor.mit.edu/explore/books.html

Neste app são usadas declarações condicionais e funções, porém esses conceitos foram abstraídos ao serem fornecidos prontos no projeto base, para nesta oficina se focar apenas em listas e variáveis. As listas são usadas para armazenar as imagens dos monstros e seus pontos de vida iniciais, e as variáveis para indexar as listas.

4.2. Locais de Reciclagem

Este é um app informativo, que exibe locais de reciclagem em um mapa (Fig. 2b). Para isso, os tutores estudaram alternativas para criar e mostrar o mapa dentro do app. A solução encontrada foi utilizar o My Maps da Google para marcar os locais de reciclagem, e então criar o aplicativo no AI2 para iniciar a aplicação Google Maps, que precisava estar instalada no dispositivo.

Este projeto aborda a composição como um recurso poderoso em programação e desenvolvimento de *software*, conforme proposto por dois tutoriais inspiradores disponíveis no site do MIT App Inventor[6][7]. Os dois utilizam GPS e o componente "Iniciador de Atividades", que pode iniciar uma atividade do Android, como um navegador para uma página da *web* ou o aplicativo da câmera. Usou-se a possibilidade de abrir o Google Maps para o mapa desenvolvido no My Maps, definido por um URL, ao clique de um botão na tela inicial.

4.3. Acerte a Lixeira!

Esse app foi inspirado em um tutorial do AI2 do jogo Space Invaders [8]. Nesta versão (Fig. 2c), o jogador lança uma bola de papel em direção a uma lixeira, que se movimenta aleatoriamente no topo da tela. A bola de papel sai de um saco de lixo, arrastado pelo jogador na extremidade inferior da tela. Para cada acerto é marcado um ponto e o jogo pode ser reiniciado a qualquer momento.

Este jogo explora o conceito de procedimentos/funções, chamados para mover a bola na tela. Ele também contém condicionais associados a itens na tela, como marcar um ponto quando a bola acerta a lixeira ou definir a nova posição do saco de lixo quando arrastado, servindo como uma prévia para o próximo projeto.

4.4. Guia de Descarte de Lixo

O quarto aplicativo (Fig. 2d) também é um jogo, em que o jogador deve determinar o descarte adequado de lixo para onze imagens de lixo. A tela mostra a imagem atual, a pontuação, um botão de reiniciar e quatro botões representando as opções de lixeiras com suas respectivas cores, como branco para lixo hospitalar e marrom para lixo orgânico.

Este projeto enfatiza o conceito de condicionais, além de rever outros anteriormente abordados. A implementação no AI2 consiste em inserir as imagens de lixo, conhecendo suas respectivas lixeiras, e conferir com uma declaração "quando verdadeiro"se a imagem selecionada para aquela lixeira pertence ou não a sua coleção de imagens.

[6]http://explore.appinventor.mit.edu/ai2/android-wheres-my-car
[7]http://explore.appinventor.mit.edu/displaying-maps
[8]http://explore.appinventor.mit.edu/ai2/space-invaders

5. Discussão

A avaliação da experiência ocorreu principalmente por meio de observações e *feedback* informal dos estudantes e tutores. Nos parágrafos seguintes, resume-se as lições aprendidas, enfatizando aspectos positivos e desafiadores enfrentados.

Aplicativos móveis. Acredita-se que focar em aplicativos para dispositivos móveis foi uma estratégia motivacional bem sucedida. O teste dos apps em dispositivos reais claramente motivou os estudantes, que ficaram animados ao ver suas criações no *smartphone* e ao mostrá-las aos colegas. Essas observações ratificam outras experiências de educação não-formal usando o App Inventor [Roy 2012, Wagner et al. 2013]. Como mencionado antes, os estudantes participaram de uma experiência no primeiro semestre do ano, sobre um assunto diferente mas relacionado (problemas lógicos da OBI). Nessa experiência, teve-se uma taxa de frequência menor. O foco em apps atraiu os estudantes que haviam abandonado o programa e, o mais importante, a maioria deles participou de todos os encontros.

Programação baseada em blocos. A abordagem baseada em blocos confirmou ser uma alternativa acessível para introduzir programação a iniciantes. Os estudantes rapidamente se familiarizaram com a movimentação e montagem dos blocos para alcançar um objetivo. Como era esperado, também tiveram algumas dificuldades, não relacionadas a problemas de lógica. Os estudantes relutaram principalmente com os blocos condicionais, pois eles exigem que o usuário selecione e monte mais ”peças”em comparação a outros blocos. Os tutores foram de grande auxílio aos estudantes, evitando que eles passassem muito tempo montando os condicionais corretamente.

Atividades temáticas. A abordagem temática foi eficaz em trazer valor e significado aos apps criados. Após uma experiência, um aluno afirmou que *“Criar aplicativos é um jeito legal de nos lembrar que temos que cuidar do ambiente”*. De fato, criar aplicativos em torno de um tema integrativo não requer um esforço adicional dos estudantes e ainda tem a vantagem de fortalecer assuntos que já são recorrentemente trabalhados em sala de aula. Mesmo para uma atividade extra-classe, é benéfico promover alguma colaboração com a escola, o que conseguiu-se quando os professores se envolveram sugerindo um tema. Tutores têm mais trabalho com essa abordagem, mas observou-se que isso foi motivador e os desafiou de maneira positiva, indo além de apenas replicar o que aprenderam em tutoriais do AI2.

Ambiente de programação. Mesmo o MIT App Inventor servindo bem para a proposta, observaram-se algumas desvantagens. (i) Ele requer que os estudantes façam *login* com um e-mail da Google, e a maioria não tinha o costume de usar e-mail. Assim, gastou-se um tempo criando contas no primeiro encontro, sendo que nos encontros seguintes alguns haviam esquecido sua senha. Descobriu-se que é possível hospedar AI2 em um servidor local, para que os usuários não precisem se registrar, porém, não houve tempo e recursos para isso ser efetivado nesta experiência. (ii) O AI2 é baseado em um navegador, mas o emulador de dispositivo é uma aplicação *desktop* e, portanto, requer um tempo para

ser instalado nos computadores. O emulador é um recurso valioso, pois alguns estudantes não têm dispositivos reais ou conexão para testar seus apps, mas pode ser uma fonte de problemas quando eles tentam instalá-lo e iniciá-lo sozinhos. Recomenda-se que esse processo seja orientado pelos tutores no início, assim os estudantes podem se concentrar apenas na criação dos apps.

6. Considerações finais

Neste trabalho, explorou-se uma abordagem temática inserida em um clube de programação extra-classe, oferecido a estudantes de uma escola localizada em um bairro socialmente desfavorecido. O tema foi escolhido colaborativamente, envolvendo estudantes, professores da escola e tutores de graduação.

Alcançou-se o objetivo de envolver os estudantes em atividades de programação ao mesmo tempo em que se enfatizava o assunto do descarte adequado de lixo. Acredita-se que envolver um único tema em vários projetos foi uma abordagem vantajosa, pois valorizou um assunto visto pelos estudantes em sala de aula, chamando sua atenção e dando sentido aos aplicativos criados com blocos de programação.

Essa experiência teve um escopo limitado, e pode não ser facilmente generalizada a outros temas, mas acredita-se que o tema de descarte de lixo poderia ser reutilizado em contextos similares, visto que problemas ambientais são uma preocupação global. Visando incentivar a replicação desta experiência, discutiram-se lições aprendidas que podem ser úteis a outros programas. Também, disponibilizam-se os projetos base e aplicativos completos para *download* em: https://goo.gl/RG5XLz.

Agradecimentos

Os autores agradecem à direção, professores e alunos da Escola Marista Santa Marta, pela receptividade e apoio incondicional ao projeto, e a todos os universitários tutores que colaboraram nesta experiência.

Referências

Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M., and Zecca, L. (2014). Extracurricular activities for improving the perception of informatics in secondary schools. In *Proceedings of the 7th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, (ISSEP)*, pages 161–172. Springer.

Dee, H., Cufi, X., Milani, A., Marian, M., Poggioni, V., Aubreton, O., Rabionet, A. R., and Rowlands, T. (2017). Playfully coding: Embedding computer science outreach in schools. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, pages 176–181. ACM.

Eshach, H. (2007). Bridging in school and out-of-school learning: formal, non-formal, and informal education. *Journal of Science Education and Technology*, 16(2):171–190.

Giaya, D., Macmillan, C., Price, A., and Roche, M. (2015). Creating engaging computer programs for children in an after-school computer club. Interactive Qualifying Project E-project-121414-190803, Worcester Polytechnic Institute, Melbourne, Australia.

Gottfried, M. A. and Williams, D. (2013). STEM club participation and STEM schooling outcomes. *Education Policy Analysis Archives*, 21(79).

Hamadache, A. (1991). Non-formal education: A definition of the concept and some examples. *Prospects*, 21(1).

Kafai, Y. B. and Resnick, M., editors (1996). *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*. Lawrence Erlbaum Associates.

Lakanen, A.-J., Isomöttönen, V., and Lappalainen, V. (2014). Five years of game programming outreach: Understanding student differences. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 647–652. ACM.

Lye, S. Y. and Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming. *Comput. Hum. Behav.*, 41(C):51–61.

Meyer, D. and Batzner, A. (2016). Engaging computer science non-majors by teaching k-12 pupils programming: First experiences with a large-scale voluntary program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli Calling '16, pages 174–175. ACM.

Ni, L., Schilder, D., Sherman, M., and Martin, F. (2016). Computing with a community focus: Outcomes from an app inventor summer camp for middle school students. *J. Comput. Sci. Coll.*, 31(6):82–89.

Retnawati, H., Munadi, S., Arlinwibowo, J., Wulandari, N. F., and Sulistyaningsih, E. (2017). Teachers' difficulties in implementing thematic teaching and learning in elementary schools. *The New Educational Review*, 48:201–212.

Roy, K. (2012). App inventor for android: Report from a summer camp. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 283–288. ACM.

Sahin, A. (2013). STEM clubs and science fair competitions: Effects on post-secondary matriculation. *Journal of STEM Education: Innovations and Research*, 14(1):5–11.

Smith, N., Sutcliffe, C., and Sandvik, L. (2014). Code club: Bringing programming to uk primary schools through scratch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 517–522. ACM.

Sullivan, K., Byrne, J. R., Bresnihan, N., O'Sullivan, K., and Tangney, B. (2015). Codeplus — designing an after school computing programme for girls. In *Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE)*, FIE '15, pages 1–5. IEEE Computer Society.

Tessier, L. and Tessier, J. (2015). Theme-based courses foster student learning and promote comfort with learning new material. *Journal for Learning through the Arts*, 12(1).

Wagner, A., Gray, J., Corley, J., and Wolber, D. (2013). Using app inventor in a k-12 summer camp. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 621–626. ACM.

Weintrop, D. and Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th Int. Conf. on Interaction Design and Children*, IDC '15, pages 199–208. ACM.

Cenários Prospectivos: Uma Visão do Futuro da Presença Feminina em Cursos de Ciência da Computação de uma Instituição de Ensino Superior

Josilene A. Moreira[1], Ricardo M. Silva[2], Maria Eulina P. Carvalho[3]

Centro de Informática[1], Centro de Tecnologia[2], Centro de Educação[3]
Universidade Federal da Paraíba - UFPB
Av. dos Escoteiros - s/n - Distrito Industrial de Mangabeira
58055-000 - João Pessoa-PB

josilene@ci.ufpb.br[1], ricardomoreira0203@hotmail.com[2],
mepcarv@terra.com.br[3]

Abstract. *This article projects scenarios and presents gender equity trend curves using regression analysis, based on historical quantitative series of students in three courses in the Computer Science area of a Brazilian University. In the Bachelor of Science degree in Computer Science it is observed that the number of women is reduced to a rate of 0.4% per year (aa); without interventions to reverse this trend, there will be no more women in 2050. In Computer Engineering there is a decrease in the entry of women to 0.7% per year; if the fall continues, there will be no women in 2031. In Computational Mathematics, the scenario is indicative of stagnation; there is a tendency to reduce women by 0.01% per year.*

Resumo. *Este artigo projeta cenários e apresenta curvas de tendência de equidade de gênero utilizando análise de regressão, baseado em séries históricas de quantitativos de estudantes em três cursos da área de Computação de uma IES do Brasil. No curso de Bacharelado em Ciência da Computação constata-se que o número de mulheres reduz-se a uma taxa de 0,4 % ao ano (aa); sem intervenções para reverter essa tendência, não haverá mais mulheres em 2050. Na Engenharia da Computação há diminuição na entrada de mulheres a 0,7% aa; caso a queda se mantenha, não haverá mulheres em 2031. Já na Matemática Computacional, o cenário é indicativo de estagnação; há uma tendência de redução das mulheres em 0,01% ao ano.*

1. Introdução

De acordo com o Censo da Educação Superior, as mulheres constituíram a maioria dos estudantes matriculados em cursos superiores no Brasil em 2015: do total de 8,3 milhões elas correspondem a 55,6%. Elas também superam os homens número de ingressantes (53,9%) e de concluintes (59,9%). Entretanto, quando analisamos os cursos e áreas de ingresso, verificamos uma polarização entre as escolhas de homens e mulheres: elas predominam nas áreas compreendidas como femininas, tipicamente ligadas ao cuidado, enquanto eles predominam na área tecnológica. Entre os dez maiores cursos de graduação em número de matrículas por sexo as mulheres são maioria em Pedagogia, Enfermagem, Psicologia, Serviço Social, Gestão de Pessoas, Fisioterapia e Arquitetura, enquanto os homens são maior número em Engenharia Civil,

Engenharia Mecânica, Engenharia de Produção, Formação de Professor em Educação Física, Análise e Desenvolvimento de Sistemas e Engenharia Elétrica[1] (INEP, 2016).

A Universidade Federal da Paraíba (UFPB) foi criada pela Lei Estadual 1.366, de 02/12/1955, sob o nome de Universidade da Paraíba e posteriormente, com a sua federalização, aprovada e promulgada pela Lei nº. 3.835 de 13/12/1960, foi transformada em Universidade Federal da Paraíba. Nos últimos cinco anos, com a adesão ao novo Plano de Reestruturação e Expansão das Universidades (REUNI), do Governo Federal, a UFPB conseguiu dobrar de tamanho e, hoje, já é a instituição de ensino superior do Norte e Nordeste do país a oferecer o maior número de vagas no seu processo seletivo (UFPB, 2016). Possui quatro campi, sendo o Campus I, na cidade de João Pessoa o maior, contando com treze centros. O curso de Bacharelado em Ciência da Computação, criado em 1985, vinculado originalmente ao Departamento de Informática do Centro de Ciências Exatas e da Natureza, encontra-se hoje vinculado ao Centro de Informática (CI), criado em 2012. O CI conta com mais dois cursos na área de Computação: Engenharia da Computação, criado em 2011, e Matemática Computacional, criado em 2012.

Embora recomendada pelo Conselho Econômico e Social das Nações Unidas em 1997 (ONU, 2002), a transversalidade de gênero (*gender mainstreaming*) na Educação anda em passos lentos Brasil. Admitindo que "a educação brasileira ainda não incorporou totalmente o princípio da igualdade de gênero" e que permanece "o sexismo nas escolhas das carreiras acadêmicas" (Brasil, 2013, p.22), o Plano Nacional de Políticas para Mulheres (PNPM) 2013-2015 propõe "promover políticas para a ampliação do acesso e permanência das mulheres no ensino profissional, tecnológico e no ensino superior, com destaque para as áreas científicas e tecnológicas" (p.23). Entre as ações propostas, indicou políticas de ação afirmativa e campanhas para ampliar o número de mulheres em cursos tradicionalmente masculinos (p.26).

Embora nosso Plano Nacional de Educação - PNE 2014-2024 seja omisso no tocante a essas questões, nosso Ministério da Ciência, Tecnologia e Inovação (MCTI), na Estratégia Nacional de Ciência, Tecnologia e Inovação (ENCTI) 2016-2019, destaca a adoção de programas (como outros países já fizeram) para "implantação de uma política de gênero nas instituições científicas" e a "promoção da paridade (...) a fim de reduzir as desigualdades e combater a discriminação" (BRASIL MCTI, 2016, p.55). Focaliza ainda a redução de disparidades no desenvolvimento das carreiras de CT&I e a transversalidade da abordagem de gênero nas pesquisas, argumentando que "garantir e incentivar a participação plena e efetiva das mulheres nas Ciências e assegurar a igualdade de oportunidades na área de CT&I" traz "benefícios diretos para a sociedade como um todo" (p.57).

A realidade atual quanto à igualdade de gênero nos cursos do Centro de Informática segue a tendência da predominância masculina nas áreas de tecnologia, conforme o panorama de todo o Brasil. No curso de Bacharelado em Ciência da Computação as mulheres são 10,2% (39) dos estudantes matriculados; na Engenharia da Computação elas representam 13,8% (47) e na Matemática Computacional são 21% (38). Esta pesquisa, com base em dados quantitativos históricos de alunos ingressantes, matriculados e concluintes nestes cursos desde a sua criação, constrói e apresenta as

[1] Administração, Direito e Ciências Contábeis estão entre os dez maiores cursos em ambos os gêneros.

curvas de tendências que indicam as possibilidades de cenários futuros quanto à perspectiva de equidade de gênero. Conclui que as desigualdades atuais tendem a permanecer e piorar, na ausência de uma política de inclusão de gênero direcionada à transversalidade e ao alcance da paridade entre discentes.

A técnica de construção de cenários, segundo Schwartz (2000) "é uma ferramenta para ajudar a perceber uma visão em longo prazo (com arte e criatividade) combinando com a prática da conversação estratégica, num mundo de grande incerteza política, social, econômica e tecnológica". Nesse sentido o método de construção de cenários pode ser alicerçado em diferentes paradigmas, ora usando modelos puramente matemáticos, colocando o ser humano em um segundo plano, ora com um foco maior neste último. Em contrapartida, compreendendo-se que os cenários podem ser afetados e modificados por fatores externos, apresenta-se aqui o projeto Meninas na Computação e suas estratégias para alterar a realidade atual e vindoura, promovendo assim uma maior inserção feminina na Ciência da Computação e no eixo da tecnologia.

Espera-se contribuir para o contexto do Ensino Superior em Computação apresentando a realidade da (des)igualdade de gênero e a necessidade de novas abordagens e políticas públicas que possibilitem às mulheres ingressarem em áreas tradicionalmente consideradas masculinas na Ciência e Tecnologia, especificamente na Computação, a partir de um estudo de caso realizado na IFES objeto do nosso estudo.

2. Inclusão da Perspectiva de Gênero na Educação Superior e na Tecnologia

Conforme Carvalho, Moreira e Silva (2018), as políticas supranacionais têm enfatizado a promoção do acesso das mulheres às TIC (Tecnologias de Informação e Comunicação), o estímulo para as meninas estudarem STEM (do inglês *Science, Technology, Engineering, and Mathematics* - Ciências, Tecnologia, Engenharia e Matemática), a necessidade de promover a participação e o avanço das mulheres nos setores tecnológicos, e a mudança dos estereótipos de gênero para que homens e mulheres compartilhem o trabalho doméstico. Dentre os Objetivos de Desenvolvimento Sustentável – ODS Agenda 2030, proclamados pela ONU em 2015, o ODS 5 - "Alcançar a igualdade de gênero e empoderar todas as mulheres e meninas" invoca o fim de todas formas de discriminação e a garantia de participação plena e efetiva das mulheres na vida política, econômica e pública (ONU, 2002; ONU, 2015).

A inclusão da perspectiva de gênero na educação superior pode assumir distintos enfoques: institucional, com garantia de apoio à necessária capacitação docente; curricular, envolvendo a transformação da prática docente em conteúdos e metodologia; e comportamental. Sua implementação tem se dado segundo diversas modalidades: transversalidade (a mais ampla e profunda), disciplinas, cursos específicos e eventos. No Brasil tem se dado limitadamente a partir de iniciativas individuais de docentes para a inclusão de conteúdos em disciplinas e oferta de disciplinas (em geral optativas) em alguns cursos, sobretudo nas áreas de ciências sociais e humanas, e mais na pós-graduação do que na graduação. Portanto, a inclusão de gênero na educação superior brasileira padece, em geral, de fraca institucionalização e não alcança a área de STEM (Lima e Costa, 2016).

De fato, as mulheres conquistaram alguns direitos civis, sociais e políticos antes nominalmente próprios dos homens. Entretanto existem algumas permanências, repaginando-se na antiga divisão sexual que estabeleceu espaços, atividades e valores

distintos e assimétricos para homens e mulheres. Um dos aspectos que demonstram a perpetuação dessa divisão são as escolhas de cursos das áreas em Instituições de Ensino Superior. Nesses cursos há uma relação desigual, e assim, uma aceitação e adesão inconsciente das regras pelos estudantes graças ao "habitus", conforme reforça Bourdieu (1999).

A questão "por que tão poucas?" tem sido feita por estudiosos e educadores, interessados em STEM, em vários países. As respostas envolvem desde a socialização primária e secundária, que reproduz estereótipos de gênero negativos sobre o desempenho intelectual das mulheres, rebaixando sua autoestima, autoconfiança e autoeficácia, até o clima frio ou hostil nos ambientes acadêmicos (Cooper & Eddy et al, 2010), além da persistência de crenças sexistas sobre a preponderância do talento inato sobre a aquisição de competências matemáticas e espaciais (Hill, Corbett, St. Rose, 2010). Esses fatores não apenas desviam as meninas da escolha de carreiras em STEM, ao final do ensino secundário ou médio, mas condicionam sua evasão quando ingressam em cursos superiores da área (Blickenstaff, 2005).

Desde a década de 1990, há países que desenvolvem iniciativas para incentivar a inclusão de meninas e mulheres em cursos e carreiras de STEM. Na União Europeia (UE), a política de promoção da equidade de gênero em universidades e institutos de pesquisa tem assumido três abordagens: tratamento igual, ação afirmativa e transversalidade de gênero. Todavia, o avanço das mulheres nos campos científicos masculinos tem sido lento; constata-se que, independentemente de disciplina e da proporção de alunas e país, elas abandonam as carreiras científicas em número muito superior aos homens em todas as etapas e especialmente após o doutorado (Rees, 2001).

De acordo com Jung & Apedoe (2013), nos Estados Unidos a média de mulheres que cursam o Bacharelado em Ciência da Computação é menor do que 20%, e a presença das mulheres nas carreiras relacionadas à Ciência da Computação só chega a 25%. Estes autores também citam que uma das principais razões das meninas não escolherem carreiras relacionadas à Ciência da Computação é a falta de conhecimento sobre os diversos tópicos cobertos nos cursos superiores e sobre as diversas oportunidades oferecidas por estas carreiras. No século XXI ainda não se tem uma academia amistosa para as mulheres e persistem o clima frio e o teto de vidro (Cooper e Eddy et al, 2010), mesmo em países onde há legislação e políticas de combate à desigualdade (Kjeldala, Rindfleisha e Sheridana, 2005), sobretudo nos campos do conhecimento dominados por homens.

Nesse contexto, surgiu no Brasil o Programa Mulher e Ciência, em 2005, através de parceria entre vários ministérios (MCTI, CNPq, SPM, MEC, MDA) e a ONU-Mulheres, com três estratégias de intervenção: financiamento de projetos de pesquisa em gênero, mulheres e feminismos, instituição do Prêmio Construindo a Igualdade de Gênero para estudantes de ensino médio, graduação e pós-graduação, e realização do Encontro Nacional de Núcleos e Grupos de Pesquisa "Pensando Gênero e Ciências". Em 2013, foi lançada a chamada pública "Meninas e Jovens Fazendo Ciências Exatas, Engenharias e Computação" (CNPq/MCTI, SPM, Petrobrás), visando estimular a formação de mulheres para as carreiras de CTEM, articulando ensino médio (para atrair novas alunas) e ensino superior (para minimizar a evasão).

Entretanto, as políticas públicas para a inclusão feminina na Ciência e Tecnologia não têm tido a continuidade e a efetividade necessárias para garantir que as

mulheres realmente ganhem novos espaços nesta grande área. Apresentamos a seguir dados e tendências de três cursos da área de Ciência da Computação construídos através da técnica de cenários que mostram a reduzida participação feminina neste espaço de atuação.

3. Metodologia

As tendências são baseadas em dados históricos sobre os quais são calculados os cenários futuros; relembra-se que toda a projeção é calculada isolando-se os fatores externos (que, caso ocorram, podem interferir nas projeções). Quando a análise é feita com dados históricos, o problema da desigualdade torna-se mais visível. Foram analisados os dados do alunado dos três cursos, por sexo, utilizando-se as seguintes variáveis preditivas: "entrada de graduandos" e "saída de graduados", em uma perspectiva longitudinal desde a fundação de cada curso, percorrendo-se o seguinte caminho metodológico:

(i) Transformação de números quantitativos em percentagem por sexo;

(ii) Cálculo do coeficiente de Pearson ($R>0,8$ em todos os conjuntos de dados, mostrando forte correlação) e da curva de tendência através da análise de regressão, técnica que permite explorar e inferir a relação de uma variável dependente com variáveis independentes, utilizando-se o método dos mínimos quadrados ordinários, adotando-se o modelo $Y= a+bX$ (Triola, 2013);

(iii) Construção e interpretação dos gráficos contendo os dados em percentual e reta de tendência, possibilitando prospecção do cenário futuro provável.

Silva e Santos (2009) afirmam que apesar das inúmeras limitações dos modelos matemáticos da criação de cenários, eles possuem o mérito de, serem baseados no passado, apontarem o futuro, mesmo com a constatação de que o ambiente é mais complexo do que esses modelos podem prever. Esse é o caso desse artigo, onde todos os cálculos das curvas de tendência foram feitos através da análise de regressão utilizando dados históricos de cada curso da UFPB.

4. Cenários e Tendências

Vejamos o que se passou na IFES estudada nas últimas décadas, onde a pequena vantagem numérica das mulheres em matrículas gerais em 2017 nos cursos de graduação (51,5% de 33.228) (STI/UFPB, 2018) contrasta com a sua reduzida presença em cursos masculinos da área de Computação, Centro de Informática, foco do nosso estudo (Tabela 1).

Tabela 1: Discentes mulheres por curso do Centro de Informática, UFPB, 2017.

Curso	Mulheres
Ciência da Computação	39 (10,2%)
Engenharia de Computação	47 (13,8%)
Matemática Computacional	38 (21%)

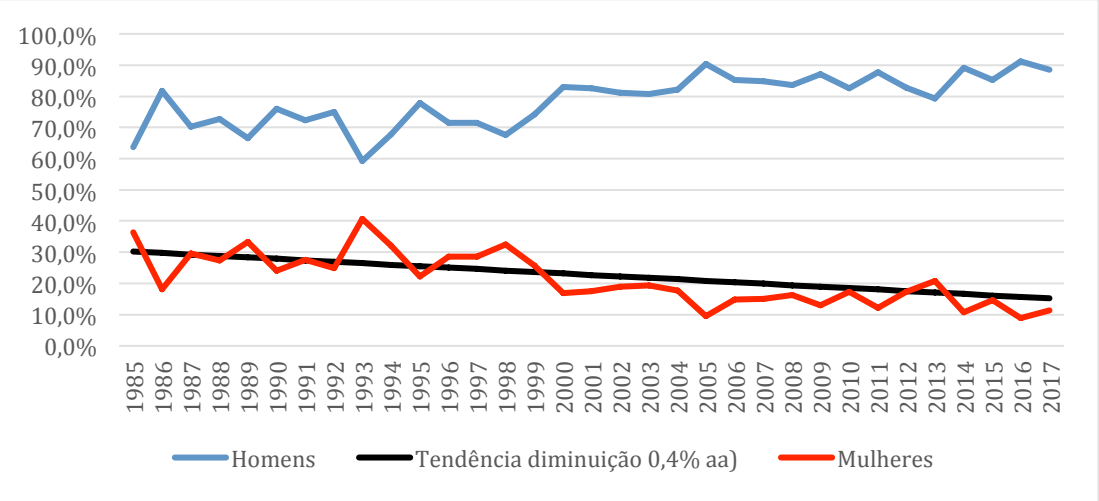
Fonte: SIGAA (matrículas ativas em no período 2017.2)

As projeções foram calculadas a partir do número de estudantes ingressantes a cada ano. Os gráficos mostram o percentual de homens e mulheres ingressantes e a curva de tendência calculada a partir destes percentuais.

4.1 Ciência da Computação

Quando este curso foi fundado, em 1985, atraiu um maior número de mulheres. Na primeira turma elas eram 36,4% e chegaram a 40,7% em 1993; entretanto, uma queda acentuada no ingresso feminino se deu a partir do ano 2000. A média de mulheres no período analisado é 21,3% e a de homens 78,7% (Figura 1). Há uma tendência de diminuição das mulheres a uma taxa de 0,4 % aa, ou seja, a desigualdade aumenta. Sem ações afirmativas para reverter essa tendência, não haverá mais mulheres em 2050.

Figura 1: Desigualdade de sexo/gênero na Ciência da Computação

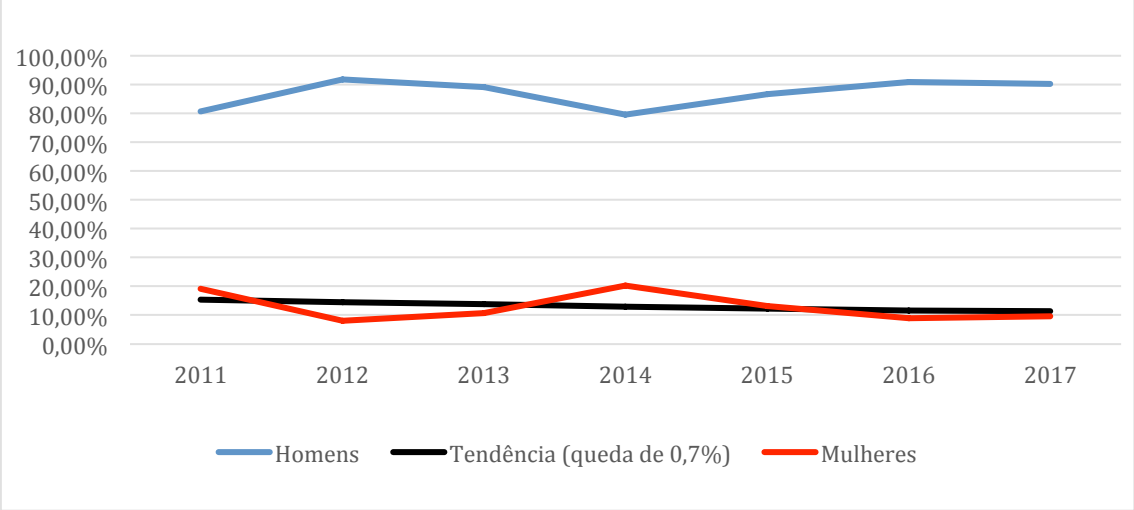


Fonte: Elaboração própria a partir de dados brutos fornecidos pelo STI/UFPB, 2018.

4.2 Engenharia de Computação

Neste curso, criado em 2011, o cenário futuro é de tendência de crescimento da defasagem entre homens e mulheres (Figura 2), pois há diminuição na entrada de mulheres a 0,7%aa. A média de homens é 87% e de mulheres de 13%. Na ausência de ações afirmativas para modificar esse processo, no ano 2031 não haverá mais nenhuma mulher neste curso, na contramão da inclusão feminina.

Gráfico 5: Desigualdade de sexo/gênero na Engenharia de Computação

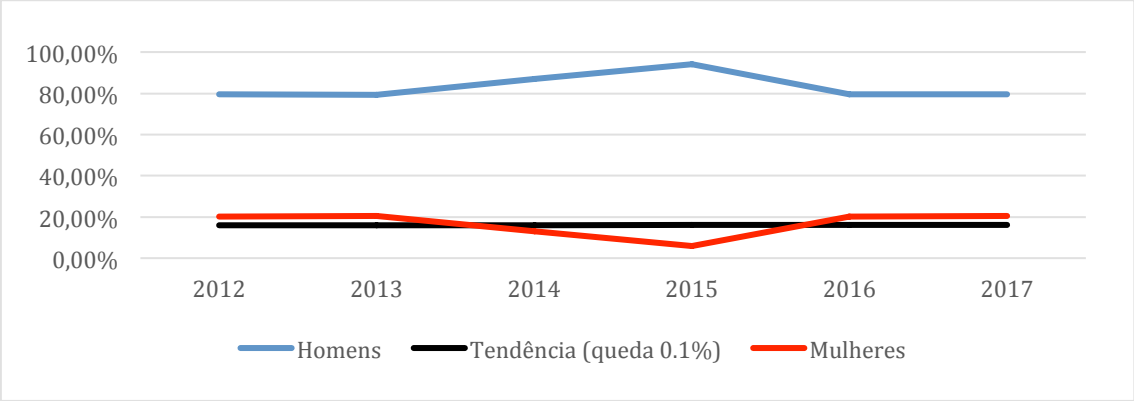


Fonte: Elaboração própria a partir de dados brutos fornecidos pelo STI/UFPB, 2018.

4.3 Matemática Computacional

Também recente (criado em 2012), este curso tem média de homens de 83,2%, de mulheres de 16,8%. Seu cenário é indicativo de estagnação, pois há uma tendência de diminuição das mulheres em 0,01% ao ano (Figura 3).

Figura 3: Desigualdade de sexo/gênero na Matemática Computacional



Fonte: Elaboração própria a partir de dados brutos fornecidos pelo STI/UFPB, 2018.

5. Comentários Finais

Quando se considera a média de ingressantes em cada período analisado, o ingresso no último ano e a tendência projetada por sexo para cada curso, conforme os gráficos apresentados e a Tabela 2, constata-se que o cenário futuro segue na contramão da equidade de gênero proposta pelos Objetivos de Desenvolvimento Sustentável (ONU, 2015). Portanto, se não houverem ações que interfiram nestas tendências, dificilmente haverá igualdade de sexo/gênero na IFES pesquisada.

Tabela 2: Resumo das tendência de (des)igualdade de sexo/gênero em cursos da área de Computação, UFPB, 2017

Cursos e período analisado	Média no período		Matrículas 2017.2		Tendência
	Homens	Mulheres	Homens	Mulheres	
Ciência da Computação 32 anos	78,7%	21,3%	88,9%	11,1%	Aumento da desigualdade a uma taxa 0,4% aa
Engenharia de Computação 7 anos	87,1%	12,9%	90,25%	9,75%	Aumento da desigualdade a uma taxa de 0,7% aa
Matemática Computacional 7 anos	83,2%	16,8%	79,63%	20,37%	Estagnação (taxa de 0,01% aa de aumento da desigualdade)

Fonte: Elaboração própria a partir de dados brutos fornecidos pelo STI/UFPB, 2018.

Sabemos que a inclusão das mulheres em áreas da Ciência e Tecnologia não depende somente de políticas curriculares focadas na igualdade de gênero na educação básica e de programas de ação afirmativa que estimulem seu ingresso na educação superior, o que não se fez ainda no Brasil. Mais amplamente, é preciso mudar a organização patriarcal da família e do trabalho, através de "um entendimento crítico de gênero, de como ele funciona na ciência e na sociedade" (Schienbinger, 2001, p.39).

Nesse contexto, entendendo inclusão de gênero nas dimensões quantitativa e qualitativa, como políticas de paridade e transversalidade, consideramos que a universidade deveria assumir o protagonismo na realização dessas políticas, intra e extramuros. Uma das possibilidades de ação são os projetos de extensão junto às escolas da comunidade (Dos Santos, 2017; Mattos, 2015). Isso compreende informar, conscientizar e sensibilizar sobre o problema, especialmente os professores, sem esquecer que para as mulheres o primeiro passo para o empoderamento é tomar consciência dos preconceitos.

Hill, Corbett e St. Rose (2010) fazem várias recomendações inadiáveis nessa direção, principalmente se direcionadas ao ensino médio:

- divulgar as contribuições históricas e os sucessos atuais de mulheres em STEM e promover modelos femininos nas respectivas carreiras;
- atrair e reter mais alunas, cultivando ambientes acadêmicos que acolham, apoiem e reconheçam seus interesses e realizações, investindo em currículos que enfatizem aplicações concretas do conhecimento, e atentando para a influência de preconceitos e vieses (geralmente inconscientes) no ensino, orientação e avaliação;

- atrair e reter docentes mulheres para a área de tecnologia, provendo mentoras, quando possível, para as alunas. Apesar de ser uma ação sobre o quadro docente, espera-se que impacte na atração de alunas para esta grande área.

A nossa recomendação é quanto à criação e adoção de políticas públicas que impactem nas escolhas de carreiras por parte das alunas do Ensino Médio, a fim de que estas sintam-se capazes, empoderadas e estimuladas a adentrar as profissões tipicamente masculinas da área de tecnologia. Consequentemente, poderá haver uma mudança nos números e nos cenários futuros de subáreas da tecnologia como a Ciência da Computação, objeto de estudo do nosso trabalho.

Referências

Blickenstaff, J. C. Women and science careers: leaky pipeline or gender filter? Gender and Education, 17 (4), 369-386. 2015.

Bourdieu, P. A Dominação Masculina. Rio de Janeiro: Bertrand Brasil, 1999.

BRASIL MCTI. Estratégia Nacional de Ciência, Tecnologia e Inovação 2016 – 2019. Brasília: Ministério da Ciência Tecnologia e Inovação (MCTI). 2016. Disponível em http://www.mcti.gov.br/documents/10179/1712401/Estrat%C3%A9gia+Nacional+de+Ci%C3%AAncia%2C%20Tecnologia+e+Inova%C3%A7%C3%A3o+2016-2019/0cfb61e1-1b84-4323-b136-8c3a5f2a4bb7, acessado em Jan/2017.

BRASIL. (2013) Presidência da República. Secretaria Nacional de Políticas para as Mulheres. Plano Nacional de Políticas para as Mulheres. Brasília: Secretaria Nacional de Políticas para as Mulheres, 2013, 114 p. Disponível em http://www.compromissoeatitude.org.br/wpcontent/uploads/2012/08/SPM_PNPM_2013.pdf, acessado em Abr/2015.

Carvalho, M. E, Silva, R. M., Aires, J. A. Estimulando o Ingresso de Meninas na Ciência e Tecnologia Através do Ensino de Computação. Encuentro Científico Internacional del Comité FES de Sociología del Género, Valencia, 2018.

Cooper, J., Eddy, P., Hart, J., Lester, J., Lukas, S., Eudey, B., Glazer-Raymo, J., Madden, M. Improving gender equity in postsecondary education. In: S. S. Klein (Gen. Ed.). Handbook for Achieving Gender Equity through Education (2 ed., pp. 631-653). New York and London: Routledge. 2010.

Dos Santos, J. M. O., Souza, C. M., Santos, T. A., Alves, P. M. B. F., & Santos, D. A. Contribuições da Extensão Universitária na formação social, acadêmica e profissional dos estudantes de Computação. Em XXXVII CSBC, 25º Workshop de Educação em Computação (WEI-2017), São Paulo, 2017.

Hill, C., Corbett, C. e St. Rose, A. Why so few? Women in Science, Technology, Engineering, and Mathematics. AAUW, Washington DC. 2010.

JUNG, Eunjin, APEDOE, Xornam. Changing Young Women's Perceptions of CS via Outreach. ITiCSE'13, the 18th Annual Conference on Innovation and Technology in Computer Science Education, Inglaterra. 2013.

Kjeldala, S. E., Rindfleisha, J. e Sheridana, A. Deal□making and rule□breaking: behind the façade of equity in academia. Gender and Education, 17 (4), p. 431-447. 2005.

Lima, B. S. e Costa, M. C. da. Gênero, ciências e tecnologias: caminhos percorridos e novos desafios. Cadernos Pagu, Campinas, n. 48. 2016. Disponível em http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-83332016000300304&lng=pt&nrm=iso, acessado em Fev/2018.

Mattos, G. O, Silva, D. Moreira, J.A. A Utilização de Kits de Robótica como Ferramenta para o Ensino de Programação à Meninas do Ensino Médio. Em XXXV Congresso da Sociedade Brasileira de Computação, 23o Workshop de Educação em Computação (WEI-2015), Recife, 2015.

ONU. Gender mainstreaming. An overview. New York: Office of the Special Adviser on Gender Issues and Advancement of Women. Department of Economic and Social Affair, 2002. http://www.un.org/womenwatch/osagi/pdf/e65237.pdf. Acessado em 01/03/2018.

ONU. Objetivos de Desenvolvimento Sustentável – ODS Agenda 2030. 2015. http://www.onumulheres.org.br/planeta5050/. Acessado em 21/02/2018.

Rees, T. (2001). Mainstreaming Gender Equality in Science in the European Union: The 'ETAN Report'. Gender and Education, 13(3), 243-260.

Schienbinger, L. (2001). O feminismo mudou a ciência? (R. Fiker, Trad.). Bauru, SP: EDUSC.

Schwartz, Peter. A arte da visão de longo prazo: planejando o futuro em um mundo de incertezas. São Paulo: Best Seller, 2000.

Silva, Ricardo Moreira da; SANTOS, João Luiz Fonseca. A Incorporação da Imaterialidade Humana na Construção de Cenários. XXIX Encontro Nacional de Engenharia de Produção. Salvador, BA, Brasil, 06 a 09 de outubro de 2009.

STI/UFPB. Superintendência de Tecnologia da Informação da Universidade Federal da Paraíba. 2018.

Triola, Mario F. Introdução A Estatística - Atualização da Tecnologia - 11ª Ed. Editora LTC. 2013.

Uma Abordagem para Orquestração do Conhecimento como Suporte ao Planejamento Curricular em Ciência da Computação

Anderson Felinto Barbosa[1,2], **Janderson Jason Barbosa Aguiar**[1], **Ulrich Schiel**[1]

[1]Universidade Federal de Campina Grande (UFCG)
Campina Grande – PB – Brasil

[2]Instituto Federal da Paraíba (IFPB) – Campus Monteiro
Monteiro – PB – Brasil

{anderson.fbarbosa, janderson.jason}@gmail.com,

ulrich@computacao.ufcg.edu.br

Abstract. *This paper presents an approach for Orchestration of Curricular Knowledge in Computer Science based on the subjects (Knowledge Categories) of the Reference Curriculum for Computer Science and curricular structures of undergraduate courses. For this, semiautomatically, the disciplines of the structures are categorized in on the of the Categories of Knowledge, with this it is possible to measure the representativeness of each Category and to identify the possible relations of dependencies between them. A case study was carried out with 457 subjects, making it possible to identify, in the context of the 7 bachelor's courses in Computer Science, the representativeness of the Knowledge Categories and, in addition, the dependency relations between them.*

Resumo. *Este artigo apresenta uma abordagem para Orquestração do Conhecimento Curricular em Ciência da Computação baseado nas matérias (Categorias de Conhecimento) do Currículo Referência para a Ciência da Computação e em estruturas curriculares de cursos de graduação. Para isso, de forma semiautomática, as disciplinas das estruturas curriculares são categorizadas em uma das Categorias de Conhecimento, com isso é possível mensurar a representatividade de cada Categoria e identificar as possíveis relações de dependências entre elas. Um estudo de caso foi realizado com 457 disciplinas, possibilitando identificar, no contexto dos 7 cursos de bacharelado em Ciência da Computação utilizados, a representatividade das Categorias de Conhecimento e, além disso, as relações de dependência entre elas.*

1. Introdução

Na literatura, o termo *Design* Instrucional (DI) está associado ao processo de planejamento de um conjunto de métodos, técnicas e atividades que serão entregues aos alunos durante o processo de aprendizagem [Filatro 2008]. Com diferentes níveis de granularidade (Macro, Meso e Micro), o *Design* Instrucional pode ser utilizado em diferentes contextos inerentes ao processo de planejamento educacional [Barbosa et al. 2015].

O DI na granularidade meso, por exemplo,pode ser utilizado para estruturar cursos e, consequentemente, definir qual será o conjunto de disciplinas necessárias para a

formação do perfil do aluno esperado pelas instituições de ensino, além dos objetivos que devem ser alcançados pelas disciplinas, seus conteúdos e as relações de dependências (pré-requisitos) entre elas.

Para algumas áreas de conhecimento são criados documentos para auxiliar o processo de definição das disciplinas, conteúdos e objetivos educacionais. Com isso, o processo de DI que utiliza esses documentos está alinhado aos parâmetros estabelecidos por comitês, sociedades e/ou outras entidades responsáveis ou que colaboram para a educação. No Brasil, a Ciência da Computação (CC) é uma área que possui um conjunto de documentos com esta finalidade, seja no contexto nacional, por meio dos documentos criados pela Sociedade Brasileira de Computação (SBC), ou internacional, criados pela *Association for Computing Machinery* (ACM).

O documento das recomendações curriculares para a Ciência da Computação descreve de forma clara e detalhada o conjunto de conteúdos necessários para a formação dos alunos. No contexto internacional, o *Computer Science Curricula 2013* (CS2013) [Joint Task Force on Computing Curricula e Society 2013] destaca os conteúdos organizados em três diferentes granularidades, os objetivos e relações entre os conteúdos. No Brasil, a SBC disponibiliza o Currículo Referência para cursos de Ciência da Computação e Engenharia da Computação (CR05) [SBC 2005] contendo um conjunto de matérias que podem ser abordados nos cursos de graduação do país. Contudo, as relações entre os diferentdes conteúdos (matérias) não são destacadas, deixando tal definição a critério das Instituições de Ensino Superior (IES).

Diante disso, neste artigo, é apresentada uma abordagem para Orquestração do Conhecimento Curricular em Ciência da Computação com base no CR05 e em estruturas curriculares de cursos de bacharelado em CC. Para isso, de forma semiautomática e *bottom-up*, as disciplinas dos cursos de graduação das IES são categorizadas em uma das 57 Categorias de Conhecimento (matérias do CR05) utilizando-se de técnicas de categorização textual estatística com Redes Bayesianas, possibilitando a mensuração do Grau de Ocorrência da Categoria (*GOc*) e o Grau de Dependência entre Categorias (*GDep*), ambos variando entre 0 e 1.

Um estudo de caso foi realizado com finalidade de validar a abordagem proposta. Inicialmente, foi criado um conjunto de treinamento contendo os conteúdos descritos no CR05, nas diretrizes do Enade 2014 (Exame Nacional de Desempenho dos Estudantes) [INEP 2014] e nas estruturas curriculares de 8 IES brasileiras de diversas regiões e com diferentes conceitos, juntamente com a Categoria de Conhecimento correspondente. Após isso, outras 7 estruturas curriculares, totalizando 457 disciplinas, foram submetidas à abordagem para identificação da categoria correspondente e aplicação das métricas *GOc* e *GDep*.

Os resultados mostraram que das 57 Categorias possíveis, 34 apresentaram *GOc* com representatividade entre 0.5 e 1.0; 20 apresentaram valor maior que 0.1 e menor que 0.5; e 3 não pontuaram nesse quesito. Quando analisado o *GDep*, verificou-se a existência de algumas relações entre diferentes Categorias de Conhecimento, principalmente naquelas que possuem conteúdos abordados em disciplinas a partir do 2º período dos cursos analisados. Logo, com base nos resultados e identificação de ambas as métricas, é possível afirmar que a abordagem possibilita a Orquestração do Conhecimento Curricular

em Ciência da Computação.

O restante do artigo está organizado da seguinte forma: na seção 2, são comentados os conceitos fundamentais para este trabalho; na seção 3, é descrita a abordagem proposta; na seção 4, são apresentados e discutidos os resultados; e, na seção 5, estão as considerações finais.

2. Referencial Teórico

Inicialmente, o termo Orquestração pode remeter-se a uma orquestra musical na qual os integrantes precisam se combinar ou arranjar para alcançar um objetivo. Por sua vez, este termo já é utilizado em pesquisas aplicadas à educação, principalmente nas que objetivam orquestrar as atividades em Ambientes de Aprendizagem Online [Oliveira et al. 2017] ou processos didáticos [da Silva e Cavalcante 2016]. Porém, neste trabalho, a Orquestração será empregada com a finalidade de organizar o Conhecimento Curricular, ou seja, como o conhecimento deverá entregue ao aluno durante a sua formação.

Diversas pesquisas objetivam modelar o Conhecimento Curricular que deve ser entregue aos alunos durante o processo de aprendizagem. Tal modelagem, comumente, é realizada em diferentes níveis de granularidade, a fim de maximizar seu uso em diferentes aplicações. Conforme descrito na seção anterior, algumas áreas de conhecimento apresentam documentos bem fundamentados que descrevem o conhecimento necessário para cada área.

A organização do Conhecimento Curricular para a área da Ciência da Computação, pode ser auxiliada por um conjunto de documentos que descrevem o conhecimento necessário para a formação do aluno. O CS2013 [Joint Task Force on Computing Curricula e Society 2013] descreve o conhecimento curricular da CC organizado em um Corpo de Conhecimento (do inglês *Body of Knowledge*) composto por: Áreas de Conhecimento, Unidades de Conhecimento e Tópicos. O CR05 [SBC 2005], outro documento, descreve o conhecimento da CC organizada hierarquicamente em: 6 Núcleos, 58 Matérias, e um conjunto de Tópicos associados às matérias.

4. Fundamentos da Computação (F)

F1. Análise de Algoritmos
F2. Algoritmos e Estrutura de Dados
F3. Arquitetura e Organização de Computadores
F4. Circuitos Digitais
F5. Fundamentos de Sistemas
F6. Linguagens de Programação
F7. Linguagens Formais, Autômatos e Computabilidade
F8. Organização de Arquivos e dados
F9. Sistemas Operacionais
F10.Teoria dos Grafos

F1. Análise de Algoritmos

Medidas de Complexidade, Análise Assintótica de Limites de Complexidade, Técnicas de Prova de Cotas Inferiores. Notação "Big O", "Little o", "Omega" e "Theta". Medidas Empíricas de Performance. O Uso de Relações de Recorrência para Análise de Algoritmos Recursivos. Análise de Algoritmos Iterativos e Recursivos.

Figura 1. Fragmento do CR05 - Núcleo Fundamentos da Computação [SBC 2005]

Na Figura 1, é apresentada a modelagem em diferentes granularidades, no qual o Núcleo **Fundamentos da Computação** é composto por 10 Matérias. No exemplo, a Matéria **F1 - Análise de Algoritmos** é detalhada em uma série de **Tópicos** que podem auxiliar o processo de planejamento curricular das disciplinas de cursos de graduação ou

pesquisas que objetivam identificar dependências entre diferentes níveis do conhecimento, conforme a pesquisa de [Marshall 2014], que estruturou o conhecimento do CS2013 utilizando grafos e utilizou a estrutura criada para comparar os currículos de cursos em CC.

3. Abordagem para Orquestração do Conhecimento Curricular

A abordagem proposta decorre do fato de que o CR05 não apresenta relações de pré-requisitos entre os diferentes conhecimentos. Porém, tais relações podem ser úteis para o processo de planejamento curricular dos cursos de graduação e/ou estruturação do conhecimento da CC em sistemas automatizados, uma vez que, comumente, os conhecimentos não estão isolados em cursos.

Diante disso, é proposta uma abordagem *bottom-up* para Orquestrar o Conhecimento Curricular em Ciência da Computação a partir das Categoria de Conhecimento, denominada neste trabalho para referenciar as Matérias do CR05, e das relações de dependência existente entre as disciplinas dos cursos de graduação [Barbosa 2016], conforme a Figura 2.

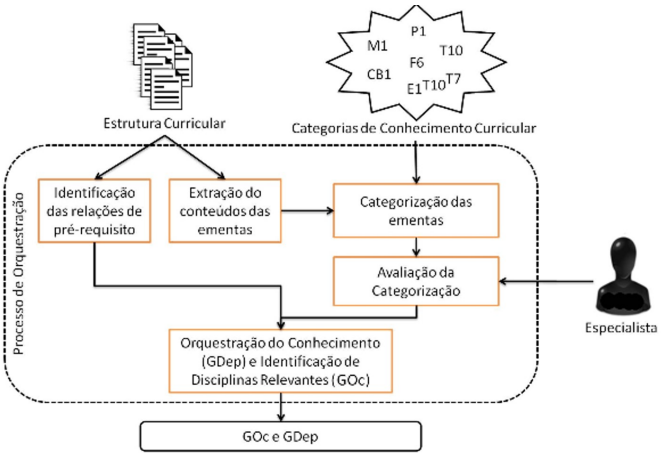


Figura 2. Arquitetura da Abordagem de Orquestração do Conhecimento

Partindo do pressuposto de que os cursos de bacharelado em CC do país baseiam-se na estruturação do conhecimento descrita no CR05, acredita-se que será possível mapear as disciplinas das IES para uma das 57 Categorias de Conhecimento. A partir disso, para cada curso analisado, deve ser realizada uma análise das diferentes Categorias identificadas para cada disciplina mapeada, e a Categoria da disciplina de pré-requisito da mesma, quando existir. Consequentemente, essa análise identificará a(s) relação(ões) de dependência(s) entre diferentes Categorias.

Como as Matérias do CR05 apresentam Conteúdos que podem ser estruturadas em mais de uma disciplina dos cursos de graduação, também foi proposto um processo de sumarização das Categorias, com a finalidade de agrupar as categorias semelhantes. Com isso, é mensurado o Grau de Ocorrência das Categorias (*GOc*) e do Grau de Dependência entre Categorias (*GDep*), fundamentais para a Orquestração. As subseções a seguir descrevem os principais conceitos e técnicas empregadas na abordagem.

3.1. Categorias de Conhecimento e Estruturas Curriculares

As Categorias de Conhecimento $C = \{c_1, c_2, ..., c_{57}\}$ utilizadas na abordagem correspondem às Matérias do CR05 (Figura 3). Das 58 possíveis, a Categoria **P8 - Estágio** foi desconsiderada por não apresentar Tópicos que pudessem ser detalhados em disciplinas dos cursos de graduação.

Matemática (M)
M1. Álgebra Linear
M2. Análise Combinatória
M3. Cálculo Diferencial e Integral
M4. Equações Diferenciais
M5. Geometria Analítica
M6. Lógica Matemática
M7. Matemática Discreta
M8. Probabilidade e Estatística
M9. Variáveis Complexas

Ciências Básicas (CB)
CB1. Física

Eletrônica (E)
E1. Circuitos Eletrônicos

Fundamentos da Computação (F)
F1. Análise de Algoritmos
F2. Algoritmos e Estrutura de Dados
F3. Arquitetura e Organização de Computadores
F4. Circuitos Digitais
F5. Fundamentos de Sistemas
F6. Linguagens de Programação
F7. Linguagens Formais, Autômatos e Computabilidade
F8. Organização de Arquivos e dados
F9. Sistemas Operacionais
F10.Teoria dos Grafos

Tecnologia da Computação (T)
T1. Análise de Desempenho
T2. Bancos de Dados
T3. Circuitos Integrados
T4. Compiladores
T5. Computação Gráfica
T6. Automação e Controle
T7. Engenharia de Software
T8. Inteligência Artificial
T9. Interação Humano- Computador
T10. Matemática Computacional
T11. Métodos Formais
T12. Modelagem e Simulação
T13. Processamento Digital de Sinais
T14. Processamento de Imagens
T15. Programação Paralela
T16. Redes de Computadores
T17. Segurança e Auditoria de Sistemas
T18. Sistemas Digitais
T19. Sistemas Distribuídos
T20. Sistemas Embarcados
T21. Sistemas Multimídia
T22.Tolerância a Falhas
T23. Telecomunicações

Contexto Social e Profissional (P)
P1. Administração
P2. Computadores e Sociedade
P3. Comunicação e Expressão
P4. Contabilidade e Custos
P5. Direito e Legislação
P6. Economia
P7. Empreendedorismo
P9. Filosofia
P10. Informática na Educação
P11. Inglês
P12. Métodos Quantitativos Aplicados à Administração de Empresas
P13. Sociologia
P14. Psicologia

Figura 3. Categorias de Conhecimento [SBC 2005]

Outro documento utilizado na abordagem é a estrutura curricular dos cursos de graduação em CC das IES. Comumente, essas estruturas descrevem o conjunto de disciplinas do curso $D = \{d_1, d_2, ..., d_{n-1}, d_n\}$, as ementas, conteúdos, relações de pré-requisitos de cada $d_i \in D$.

3.2. Categorização de Textos

O processo de Categorização de Textos (CT) ou Classificação de Textos foi utilizado com a finalidade de classificar as disciplinas das estruturas curriculares $d_i \in D$ em apenas uma das Categoria de Conhecimento $c_i \in C$. Para isso, foi utilizado o método de Categorização de Textos estatístico baseado em Redes Bayesianas [Feldman e Sanger 2007, Witten e Frank 2005] e a ferramenta WEKA[1].

A CT utiliza os conteúdos/tópicos descritos nas ementas das disciplinas para classificá-la automaticamente em uma Categoria do Conhecimento $D \times C$. Para isso, deve ser criado um conjunto de treinamento contendo o máximo de dados textuais possíveis para cada Categoria. A abordagem propõe a criação do conjunto a partir dos dados apresentados no Corpo do Conhecimento do CR05, dos conteúdos descritos nas diretrizes do Enade, e das ementas de disciplinas dos cursos de graduação que apresentaram nota máxima no exame do Enade. Devido à utilização do software de Mineração de Dados WEKA, optou-se por criar um único arquivo com extensão .arff contendo os dados extraídos de todos os documentos. O arquivo criado não possui um tamanho máximo, porém, o layout do arquivo (Figura 4) é composto por um atributo do tipo *string* que

[1]Disponível em: https://www.cs.waikato.ac.nz/ml/weka/

deve os conter conteúdos abordados na Categoria (atributo @document_ementa) e outro para o rótulo da Categoria de Conhecimento (atributo @disc_sbc) que deve ser marcado manualmente.

```
@relation Modelo_ARFF

@attribute document_ementa string
@attribute disc_sbc{algebraLinear, analiseComb,

@data

"limite e continuidade, derivada, integral inde
"vetores no plano e no espaco; retas no plano e
"algebra booleana e portas logicas; simplificac
"introducao e conceitos; logida da programacao;
"estruturas lineares e encadeadas: listas,matri
"sistemas de equacoes lineares; espacos vetoria
"componentes de computadores; numeros, artmetic
"logica, tecnicas de demonstracao e validacao d
```

Figura 4. Formato do arquivo .arff utilizado na CT

Após a criação do conjunto treinamento, devem ser criados os documentos que serão categorizados. Cada disciplina das Estruturas Curriculares dos cursos teve seu conteúdo organizado em um arquivo no formato .arff com layout semelhante ao apresentado na Figura 4, porém, neste caso, o atributo @disc_sbc deve conter uma "?" (interrogação), pois o mesmo será submetido ao processo de CT que irá mensurar o valor desse atributo, ou seja, a Categoria de Conhecimento. Devido ao uso da técnica de Redes Bayesianas, o resultado da CT indicará a distribuição de probabilidade da disciplina (termos dos conteúdos) pertencer às Categorias de Conhecimento $D \times C$ variando entre 0 (zero) e 1 (um).

Devido à natureza da técnica empregada, uma disciplina $d_i \in D$ pode ter seu conteúdo categorizado em mais de um $c_i \in C$. Diante disso, todo o resultado da CT $(D \times C)$ deve ser submetido a um processo de validação que consiste em:

1. Se $D \times C = 1 \rightarrow$ Aceitar a Categoria de Conhecimento resultante;
2. Se $0 < D \times C < 1 \rightarrow$ Deve-se analisar o nome da disciplina e conteúdo com a finalidade de:
   (a) Aceitar o resultado da CT;
   (b) Recusar o resultado e recategorizar manualmente em outra;
   (c) Recusar o resultado e recategorizar como categoria excedente, ou seja, não apresenta correspondente válido;
3. Se $D \times C = 0 \rightarrow$ Não aceitar a Categoria de Conhecimento.

Devido a não padronização dos documentos utilizados, a extração dos textos utilizados no conjunto treinamento ou das disciplinas que serão categorizadas ocorreu de forma manual. Ressalta-se que, após a extração e criação dos arquivos .arff, estes passaram por uma etapa de pré-processamento com a finalidade de remover caracteres especiais, além da redução dos documentos com a criação de *tokens*, definição da frequência mínima dos termos e remoção das *stopwords* em português.

3.3. Orquestração do Conhecimento Curricular

O processo de Orquestração do Conhecimento consiste em um conjunto de ações com a finalidade de mensurar o Grau de Ocorrência das Categorias (*GOc*) e o Grau de Dependência entre Categorias (*GDep*) identificadas após o processo de categorização.

3.3.1. Grau de Ocorrência das Categorias (*GOc*)

O resultado da CT pode mostrar diversas disciplinas $d_i \in D$ categorizadas em uma mesma $c_i \in C$ ou apresentar Categorias que não foram abordadas em uma estrutura curricular D, o que resulta em uma representação variável das Categorias de Conhecimento nas estruturas curriculares. Diante disso e objetivando identificar a relevância de determinada $c_i \in C$ nas diferentes estruturas de cursos $D_k = \{D_1, D_2, ..., D_n\}$ utilizadas na abordagem, é proposto o Grau de Ocorrência das Categorias (*GOc*).

$$GOc(c_i) = \frac{\sum_{k=1}^{n}(goc(k, c_i))}{n} \quad (1)$$

O cálculo do *GOc* (Equação 1) baseia-se na representatividade booleana da Categoria de Conhecimento na estrutura analisada. Portanto, quando houver representatividade da Categoria na estrutura do curso, o valor será 1 (um), caso contrário, o valor será 0 (zero). Após o processamento de todas as estruturas, é calculada para cada $c_i \in C$ a média aritmética da quantidade de Categorias identificadas. Para isso, utiliza-se a quantidade n de estruturas curriculares analisadas e a função $goc(k, c_i)$ que verifica o valor de cada $c_i \in C$ representada na estrutura k analisada. Devido à natureza da fórmula, o resultado será um $0 \leq GOc \leq 1$.

3.3.2. Grau de Dependência entre Categorias (*GDep*)

As relações de dependências entre as Categorias de Conhecimento são identificadas a partir da análise das relações dos pré-requisitos entre as disciplinas submetidas ao processo de CT. Devido à natureza da categorização, vários registros $d_i \in D$ podem ser categorizados em uma única categoria $c_j \in C$. Logo, poderão podendo existir registros $d_i \in D$ que apresentem relações de pré-requisito com disciplinas categorizadas na mesma Categoria de Conhecimento. Para esta situação, foi criado um algoritmo para sintetizar e identificar os relacionamentos entre as Categorias identificadas na coleção de documentos D, considerando as seguintes situações:

1. Se a disciplina categorizada apresentar relação de pré-requisito com outra disciplina de mesma categoria. Então, a relação entre as categorias será anulada (Figura 5).
2. Se a disciplina apresenta como pré-requisito uma disciplina de Categoria diferente a dela. Então, existirá uma relação correspondente entre as duas Categorias (Figura 6).
3. Se houver mais de um pré-requisito para d_i categorizada em c_i. Então, a dependência de c_i será a união de todas as Categorias de Conhecimento identificadas por meio das relações de dependência válida, conforme é apresentado na Figura 7.

Após o processo de sintetização e identificação das relações de dependências válidas, é calculado o Grau de Dependência entre Categorias ($GDep$) com a finalidade de quantificar a intensidade da relação de dependência entre diferentes Categorias do Conhecimento. Inicialmente, para cada estrutura D analisada, cria-se uma matriz $N_{57\times57}$,

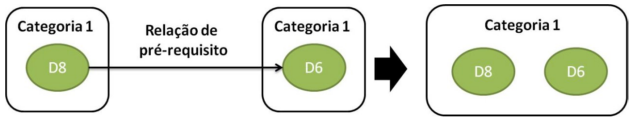


Figura 5. Exemplo de Dependência Nula entre Categorias

Figura 6. Exemplo de Dependência Válida entre diferentes Categorias

em que $N_{i,j}$ apresenta valor 0 ou 1, indicando, respectivamente, a ausência ou presença da relação de dependência entre as Categorias representadas pela linha e coluna da matriz.

As matrizes individuais N são utilizadas no cálculo do $GDep$ das Categorias de Conhecimento. Representado por uma matriz quadrada $M_{57\times57}$ e com estrutura semelhante à matriz N, o resultado do $GDep$ mostra o elemento $M_{i,j}$, com i e j variando até 57, apresentando valor $0 \leq GDep(i,j) \leq 1$. Este valor é calculado na média aritmética da função $gd(k, N_{i,j})$ que retorna o grau de relevância do elemento $N_{i,j}$, ou seja, o grau do relacionamento entre as categorias c_i e c_j, da $k-ésima$ estrutura analisada, conforme a Equação 2.

$$GDep(i,j) = \frac{\sum_{k=1}^{n}(gd(k, N_{i,j}))}{n} \tag{2}$$

Ressalta-se que o uso de ambas as métricas foi necessário, pois, se fosse considerada apenas uma delas, haveria perda de informações que poderiam ser úteis para comunidade. Por exemplo, se apenas o GOc fosse mensurado, não seria possível a identificação e mensuração das relações entre as Categorias do Conhecimento. Por outro lado, se apenas o $GDep$ fosse mensurado, seriam perdidas informações sobre a relevância das Categorias de Conhecimento nas estruturas da curriculares, pois, provavelmente algumas não seriam identificadas por não se relacionar com outras.

4. Resultados e Discussão

Conforme descrito na Seção 3, a abordagem proposta mescla atividades manuais e automatizadas baseadas em um conjunto de documentos, ferramentas e análises. De forma geral, a condução do estudo de caso baseou-se na **seleção das estruturas curriculares; na execução do processo de orquestração; e na verificação do uso das métricas identificadas no processo de planejamento e avaliação das estruturas curriculares**.

Diante disso, a seleção de estruturas curriculares utilizadas na criação do conjunto de treinamento e no processo de CT para a Orquestração baseou-se no conceito Enade 2014[2] dos cursos de graduação das IES. As estruturas selecionadas para o processo de Orquestração basearam-se na ordem decrescente das IES que obtiveram nota do conhecimento especifico (CE) do Enade 2014 superior a 50,0 pontos e que apresentavam suas

[2]Conceito Enade. Disponível em:http://portal.inep.gov.br/web/guest/conceito-enade

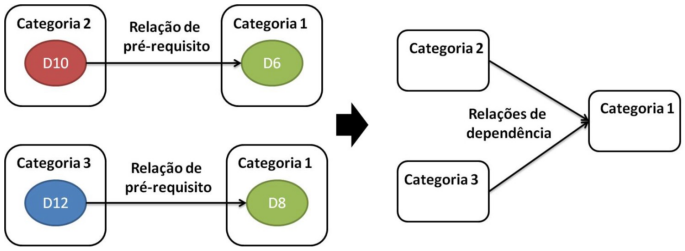


Figura 7. Exemplo de Dependência Válida entre mais de uma Categoria

estruturas curriculares disponíveis na web (Tabela 1). Para o conjunto treinamento, foram selecionadas 8 IES de diferentes regiões do país que apresentaram CE superior a 50,0 pontos e suas estruturas disponíveis na web (Tabela 2). Tal escolha deu-se por achar necessário abordar não apenas IES com nota máxima, além de diversificar o padrão de escrita dos documentos.

Tabela 1. Distribuição dos Cursos Utilizados no Processo de Orquestração do Conhecimento Curricular

Identificador do Curso	Nota Bruta - CE	Conceito ENADE
Curso 1	61,4	5
Curso 2	60,2	5
Curso 3	60,0	5
Curso 4	55,8	5
Curso 5	59,9	5
Curso 6	55,5	4
Curso 7	58,2	5

Tabela 2. Distribuição dos Cursos Utilizados no Conjunto de Treinamento

Identificador do Curso	Região	Conceito ENADE
Curso 1	Sul	5
Curso 2	Sudeste	5
Curso 3	Norte	5
Curso 4	Nordeste	4
Curso 5	Centro-oeste	3
Curso 6	Sudeste	3
Curso 7	Nordeste	3
Curso 8	Norte	2

Após a seleção, as estruturas foram analisadas para a extração dos textos e criação dos arquivos .arff (Figura 4) do conjunto de treinamento. Todo esse processo ocorreu manualmente devido a não padronização dos documentos das diferentes IES, e os dados extraídos foram estruturados em um único arquivo com 519 registros rotulados com as diferentes Categorias de Conhecimento, com distribuição conforme a Figura 8. Por outro lado, os arquivos utilizados no processo de Orquestração foram organizados em sete arquivos .arff (um para cada estrutura de IES analisada), com tamanhos variáveis, sem o rótulo da possível Categoria de Conhecimento, distribuídos conforme a Figura 9.

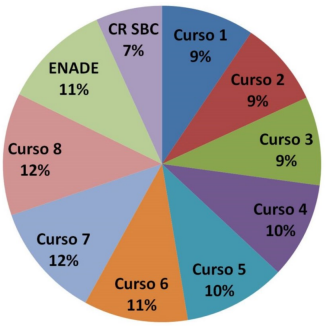


Figura 8. Distribuição dos documentos utilizados no Conjunto de Treinamento

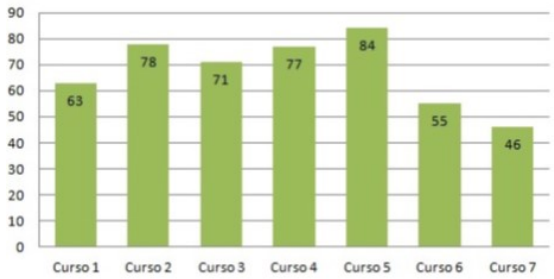


Figura 9. Distribuição da quantidade de disciplinas por curso utilizado na Orquestração

A distribuição das disciplinas por curso mostra 457 disciplinas que devem ser categorizadas em uma Categoria de Conhecimento. O uso da Mineração de Textos possibilitou a automatização desse processo. Na Tabela 3, é apresentado o resultado do processo de categorização indicando a quantidade de registros categorizados que foram aceitos ou que precisaram de recategorização (Tabela 4), conforme o algoritmo descrito na Seção 3.

Tabela 3. Resultado do Processo de Categorização

Curso	Quant. Disc.	Disc. Categ. em 1	Disc. submetidas à análise
1	63	47	16
2	71	55	16
3	78	42	36
4	77	48	29
5	84	45	39
6	55	39	16
7	46	32	14

Com as disciplinas das estruturas categorizadas, foram mensurados o GOc e o $GDep$ das Categorias de Conhecimento fundamentais para a Orquestração do Conhecimento Curricular. Ambas as métricas devem considerar todas as 57 categorias propostas, porém, conforme destacado anteriormente, as estruturas podem não abordar algumas categorias. Nesse contexto, ressalta-se que, quão mais próximo de 1 (um) for o resultado do GOc e o $GDep$, respectivamente, isso indica que a Categoria apresenta um alto grau de representatividade nas estruturas analisadas, e que a relação dela com outras também

Tabela 4. Resultado da Avaliação da Categorização dos Registros que Necessitavam de Análise

Curso	Quant. Disciplinas.	Disciplinas Aceitas	Disciplinas Recategorizadas	Disciplinas Excedentes
1	16	8	6	2
2	16	7	8	1
3	36	20	11	5
4	29	13	10	6
5	39	17	15	7
6	16	7	4	5
7	14	9	3	2

apresenta-se como representativa.

Na Figura 10, é apresentado o resultado do Grau de Ocorrência da Categoria de Conhecimento do Núcleo Fundamentos da Computação. Nesse exemplo, é possível verificar diferentes GOc variando entre $0,5 - 1$, destacando-se como representativa no estudo realizado, até os menos representativos que variam entre $0 - 0,49$. O resultado detalhado de todas as Categorias de Conhecimento pode ser visto neste link[3].

Categoria de Conhecimento	Obrigatoria	Optativa	Geral	GOC obrigatório	GOC Optativa	GOC
F1. Análise de Algoritmos	5		5	0,71	0,00	0,71
F2. Algoritmos e Estrutura de Dados	6		6	0,86	0,00	0,86
F3. Arquitetura e Organização de Computadores	7		7	1,00	0,00	1,00
F4. Circuitos Digitais	6	1	7	0,86	0,14	1,00
F5. Fundamentos de Sistemas	1	3	4	0,14	0,43	0,57
F6. Linguagens de Programação	7		7	1,00	0,00	1,00
F7. Linguagens Formais, Autômatos e Computabilidade	7		7	1,00	0,00	1,00
F8. Organização de Arquivos e dados	1		1	0,14	0,00	0,14
F9. Sistemas Operacionais	7		7	1,00	0,00	1,00
F10.Teoria dos Grafos	5	1	6	0,71	0,14	0,86

Figura 10. GOc do Núcleo - Fundamentos da Computação

Das 57 Categorias de Conhecimento utilizadas na abordagem, o $GDep$ identificou que apenas 45 apresentam relações de pré-requisitos com outras 32. Ressalta-se que uma Categoria de Conhecimento $c_i \in C$ pode relacionar-se com outras 56 Categorias, pois uma auto-relação é considerada nula, conforme descrito no processo de sintetização (Figura 5).

O resultado do $GDep$ é apresentado em uma Matriz com dimensão 57×57, na qual as linhas e colunas referem-se às Categorias de Conhecimento utilizadas[4], o que impossibilita a inserção dela no artigo. Diante disso, optou-se por apresentar o resultado $DGep$ de Categorias pertencentes ao mesmo Núcleo de Conhecimento. Na Figura 11, mostra-se o $GDep$ calculado entre as Categorias do núcleo de conhecimento Matemática, sendo possível identificar relações de pré-requisito entre as Categorias, tendo como mais representativa a relação entre **M3 - Cálculo Diferencial e Integral** e **M8 - Probabilidade e Estatística**, mensurada em 0.86, (ou seja, essa mesma relação foi identificada em dife-

[3]Resultado Completo do GOc - Disponível em https://goo.gl/5T28ii

[4]Resultado completo $GDep$ - Disponível em: https://goo.gl/bEN6PH

	M1	M2	M3	M4	M5	M6	M7	M8	M9
M1. Álgebra Linear	0,00	0,00	0,00	0,00	**0,29**	0,00	0,00	0,00	0,00
M2. Análise Combinatória	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
M3. Cálculo Diferencial e Integral	**0,14**	0,00	0,00	0,00	**0,14**	0,00	0,00	0,00	0,00
M4. Equações Diferenciais	0,00	0,00	0,00	0,00	0,00	0,00	**0,14**	0,00	0,00
M5. Geometria Analítica	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
M6. Lógica Matemática	0,00	0,00	0,00	0,00	0,00	0,00	**0,29**	0,00	0,00
M7. Matemática Discreta	0,00	0,00	0,00	0,00	0,00	**0,14**	0,00	0,00	0,00
M8. Probabilidade e Estatística	0,00	0,00	**0,86**	0,00	0,00	0,00	**0,14**	0,00	0,00
M9. Variáveis Complexas	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Figura 11. $GDep$ **entre Categorias do núcleo Matemática**

	M1	M2	M3	M4	M5	M6	M7	M8	M9
F1. Análise de Algoritmos	0,00	0,00	0,00	0,00	**0,14**	0,00	**0,14**	**0,14**	0,00
F2. Algoritmos e Estrutura de Dados	0,00	0,00	0,00	0,00	0,00	0,00	**0,29**	0,00	0,00
F3. Arquitetura e Organização de Computadores	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
F4. Circuitos Digitais	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
F5. Fundamentos de Sistemas	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
F6. Linguagens de Programação	0,00	0,00	0,00	0,00	0,00	0,00	**0,14**	0,00	0,00
F7. Linguagens Formais, Autômatos e Computabilidade	0,00	0,00	0,00	0,00	0,00	**0,43**	**0,57**	0,00	0,00
F8. Organização de Arquivos e dados	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
F9. Sistemas Operacionais	0,00	0,00	0,00	0,00	0,00	0,00	0,00	**0,14**	0,00
F10.Teoria dos Grafos	0,00	0,00	0,00	0,00	0,00	0,00	**0,14**	0,00	0,00

Figura 12. $GDep$ **entre Categorias do núcleo Fundamentos da Computação e Matemática**

rentes estruturas curriculares analisadas). Por outro lado, na Figura 12, apresenta-se uma relação $GDep$ representativa que envolve os núcleos Fundamentos da Computação e Matemática, conforme é verificado na relação entre **F7 - Linguagens Formais, Autômatos e Computabilidade** e **M7 - Matemática Discreta**, mensurada em 0.57.

Os resultados apresentados com a mensuração do GOc e $GDep$ comprovam que o conhecimento curricular em Ciência da Computação pode ser orquestrado a partir da abordagem proposta. Os resultados podem auxiliar o processo de planejamento curricular, uma vez que, dentre as Categorias de Conhecimento estabelecidas, foram identificadas aquelas que apresentam maior representatividade e, além disso, foram identificadas relações entre elas. Por fim, acredita-se que a orquestração do conhecimento também pode contribuir para pesquisas que necessitem de uma organização curricular com base nas matérias do CR05, como também em análises que verifiquem a conformidade da estrutura curricular dos cursos com o CR05.

5. Considerações Finais

A partir da mensuração das métricas GOc e $GDep$, alcançadas por meio da abordagem para Orquestração do Conhecimento proposta, foi constatado o que foi pressuposto no início do estudo: (i) as disciplinas das grades curriculares dos cursos de CC do país baseiam-se no conhecimento descrito no CR05, e (ii) é possível Orquestrar o Conhecimento Curricular com base nos pré-requisitos das disciplinas analisadas.

Com valores variando entre $0 - 1$, para o Grau da Ocorrência da Categoria (GOc) e o Grau de Dependência entre Categorias ($GDep$), conseguiu-se mensurar quão representativa é cada Categoria de Conhecimento, e como se dá a relação dela com as demais,

observando o contexto de pré-requisito. Ressalta-se que, quando mensurada a representatividade das categorias, verificou-se que mais de 50%, das 57 Categorias possíveis, apresentaram-se com representatividade acima de 0.5. Por outro lado, no contexto do estudo realizado, constatou-se que, do mesmo total de Categoria possíveis, 47 delas se relacionaram com outras 32.

Os números destacados no parágrafo anterior e detalhados nos resultados gerados mostram que, para o contexto analisado, a abordagem conseguiu responder aos questionamentos realizados e, além disso, agrega valor ao conhecimento descrito no CR05, uma vez que possibilitou a Orquestração do Conhecimento Curricular (na granularidade Matéria), podendo ser útil para o planejamento e avaliação da conformidade de um curso com o "padrão" identificado.

Contudo, ressalta-se que limitações de automatização do processo foram identificadas, porém, não inviabilizaram o objetivo final do trabalho. Ressalta-se ainda que o uso da abordagem não se restringe apenas em identificar um padrão que possa ser útil para sequenciar disciplinas de um curso de graduação em CC, ou avaliar a disposição das disciplinas no curso, mas, também, acredita-se que o resultado seja de grande valia para a comunidade que estuda o ensino em computação ou que necessita estruturar o conhecimento em outras área do conhecimento.

Referências

Barbosa, A., Nunes, I., Menezes, D., Schiel, U. (2015). O design instrucional e seu uso como arquitetura pedagógica: Uma análise das publicações em informática na educação no brasil. In *Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação (CBIE)*. Maceió, Alagoas.

Barbosa, A. F. (2016). Uma abordagem para orquestração do conhecimento com suporte ao planejamento e avaliação curricular em ciência da computação. Dissertação de Mestrado, Universidade Federal de Campina Grande.

da Silva, P. A. Cavalcante, P. S. (2016). Orquestrando processos didáticos com design thinking. In *Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE)*. Uberlândia, Minas Gerais.

Feldman, R. Sanger, J. (2007). *Text Mining Handbook: Advanced Acpproaches in Analyzing Unstructured Data.* Cambridge University Press, New York, NY, USA.

Filatro, A. (2008). *Design Instrucioinal na Prática*. Pearson Education do Brasil, 1ª edição.

INEP (2014). Portaria inep nº 238, de 02 de junho de 2014. Disponível em: http//download.inep.gov.br/educacao_superior/enade/legislacao/2014/diretrizes_cursos_diplomas_bacharel/diretrizes_bacharel_computacao.pdf. Acessado: 31 mar. 2018.

Joint Task Force on Computing Curricula, A. f. C. M. A. Society, I. C. (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, New York, NY, USA. 999133.

Marshall, L. (2014). A grapf-based framework for comparing curricula. Dissertação de Mestrado, University of Pretoria. Disponível em: https://repository.up.ac.za/handle/2263/37060. Acessado em 30 mar. 2018.

Oliveira, I. V. P. D., Gomes, A. S., Brito, J. A., Filho, I. J. M. (2017). Learning orchestration in distributed learning environments scenarios. In *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–4.

SBC (2005). Currículo de referência da SBC para cursos de graduação em bacharelado em ciência da computação e engenharia de computação. Disponível em: http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005/. Acessado: 31 mar. 2018.

Witten, I. H. Frank, E. (2005). *Data mining : practical machine learning tools and techniques*. Elservier, San Francisco, CA.

A inserção de Computação como disciplina no Ensino Fundamental: Desafios e Conquistas em Estágio Supervisionado

Arianne Sarmento Torcate, Marcos Uryel F. de Farias, Sônia Regina Fortes da Silva, Cleiton Soares Martins

Universidade de Pernambuco, Campus Garanhuns (UPE)

CEP 55.294-902 – Garanhuns – PE - Brasil

{ariannesarmento0, uryel.farias, fortes.sonia}@gmail.com, cleiton.martins@upe.br

Abstract. *The present article analyzes the teaching experience in undergraduate degree in Computing of the University of Pernambuco (UPE) - Campus Garanhuns, in the final years of elementary education, in the teaching of the discipline of Educational Computing. The purpose was to implement an educational training with young people, applying the axes: computational thinking, the world and digital culture, articulating this knowledge with students of the 6th grade B, Elementary School, School of Application Professor Ivonita Alves Guerra in Garanhuns / PE, using as a means of learning leisure activities and software. It was verified that the pedagogical action promoted a greater understanding of the students on the proposed contents.*

Resumo. *O presente artigo analisa a experiência docente em curso de Licenciatura em Computação da Universidade de Pernambuco (UPE) - Campus Garanhuns, nos anos finais do ensino fundamental, no ensino da disciplina de Computação Educacional. A finalidade foi implementar uma formação educativa com jovens, aplicando os eixos: pensamento computacional, mundo e cultura digital, articulando este conhecimento com educandos do 6º ano B, do Ensino Fundamental, da Escola de Aplicação Professora Ivonita Alves Guerra em Garanhuns/PE, utilizando como meio de aprendizagem atividades e softwares lúdicos. Constatou-se que a ação pedagógica promoveu um maior entendimento dos educandos sobre os conteúdos propostos.*

1. Introdução

As pessoas e as sociedades em que vivem testemunham mudanças de conceitos, valores e tecnologias constantemente. Devido ao ciclo natural do homem em procurar descobrir e aprender coisas novas, viver em sociedade requer apropriar-se de conhecimentos e habilidades específicas dependendo de suas necessidades e do contexto em que está inserido. Nessa perspectiva, diversos países estão buscando o acesso ao conhecimento com o uso de novas tecnologias, deste modo esta oferta tem sido muitas vezes experimentada por meio da informatização das escolas.

É notável o interesse e o progresso em relação às Tecnologias da Informação e Comunicação (TIC), que vários países estão tendo ultimamente, emergindo desse contexto iniciativas que buscam introduzir tecnologias no processo educativo e adaptando o uso da Computação, enquanto ciência, ao currículo escolar. Mas sabendo que a Computação não está inserida na base comum curricular brasileira, esta realidade acaba por retardar o processo de desenvolvimento tanto do país como também não prepara o estudante para um mercado de trabalho vasto e que está em pleno processo de ascensão.

Para todos os profissionais ainda em formação, o estágio é um dos momentos mais importantes e significativos, e quando se trata de um estágio na área de licenciatura, é nesse momento que o futuro profissional tem oportunidade de entrar em contato direto com a realidade profissional no qual será inserido, além de concretizar pressupostos teóricos adquiridos pela observação de determinadas práticas específicas e do diálogo com profissionais mais experientes, assim obtendo a troca de experiências.

No curso de Licenciatura em Computação da Universidade de Pernambuco (UPE) - campus Garanhuns, seu discente deve estar preparado para a realidade atual do Brasil em relação a sua área de atuação, assim como estar apto a ingressar em seu âmbito profissional no papel de professor. E para isto, o componente curricular Estágio Supervisionado II tem como função de preparar o futuro docente a atuar no âmbito escolar, especificamente no Ensino Fundamental, que neste caso o projeto de ensino deste componente curricular foi realizado na Escola de Aplicação Professora Ivonita Alves Guerra, localizada na cidade de Garanhuns/PE.

O projeto da dimensão de Ensino é baseado a partir do Programa de Componente Curricular elaborado pelos professores supervisores do Estágio Supervisionado II, que tem como objetivo inserir os futuros docentes no âmbito escolar, onde foram encarregados de ministrar o componente curricular denominado Computação Educacional, contemplando a turma do 6º ano B durante o segundo semestre de 2017, a fim de desenvolver o Pensamento Computacional dos educandos, contextualizando com o mundo e Cultura Digital da sociedade ao qual estão imersos.

Este artigo está organizado em 6 seções. Além da introdução, a seção 2 descreve alguns trabalhos que estão relacionados à nossa proposta de Ensino, na seção 3 é dado o início ao relato com informações gerais sobre o projeto, seguido de uma subseção detalhando as experiências vivenciadas. A seção 4 destaca quais foram os desafios que surgiram ao longo do projeto, como também o que se pôde aprender a partir deles. Na seção 5 mostra claramente os efeitos do trabalho realizado, assim como os resultados apresentados pelos educandos envolvidos na disciplina Computação Educacional e na seção 6 são relatadas as opiniões críticas dos autores deste artigo quanto a toda execução do projeto de Ensino e das experiências que foram vivenciadas.

2. Trabalhos Relacionados

Os trabalhos citados nesta seção estão diretamente ligados com o contexto do presente artigo, onde são expostas reflexões sobre à inserção do Ensino de Computação e as experiências vivenciadas ao promover esse ensino, seja ele de forma interdisciplinar ou não. É de grande valia esse momento de leitura de trabalhos que já foram executados, pois é através deles que podemos melhorar e adaptar nosso projeto, tornando-o mais robusto e coeso.

Em Silva et al. (2017), os autores relatam a experiência vivenciada através da disciplina de Estágio Curricular V do curso de Licenciatura em Computação da Universidade Federal Rural de Pernambuco, onde foi proporcionado a professores de escolas públicas um curso de formação de professores em Pensamento Computacional, com o intuito de promover Computação como Ciência interdisciplinar. É válido mencionar que os autores observaram uma desmotivação partindo de alguns dos professores convocados para participação do curso por não terem conhecimento sobre a temática do projeto e os que participaram se surpreenderam ao descobrir a relação do tema com a tecnologia.

Em Reis et al. (2017), os autores apresentam e descrevem como se deu a experiência do ensino de conceitos de Ciências da Computação aplicada no ensino fundamental, sempre relacionando com o Pensamento Computacional e buscando formas de ensino que proporcionem uma aprendizagem significativa para os alunos. Para que isso fosse possível, os autores utilizaram metodologias diferenciadas, como: Storytelling, Computação Desplugada e Gamificação, com o intuito de despertar nos alunos habilidades que estão diretamente associadas ao Pensamento Computacional.

No trabalho de Torcate et al. (2017), os autores relatam a experiência do Ensino de Computação de forma interdisciplinar com Português, onde foram abordados conceitos fundamentais de Computação, exemplo: Números Binários, Pensamento Computacional e Estrutura Condicional. O referido artigo traz resultados relevantes, como o desenvolvimento de um Objeto de Aprendizagem criado em cima das dificuldades e necessidades dos alunos, foram também utilizados softwares para auxiliar os alunos na criação de histórias em quadrinhos, como por exemplo o HagáQuê.

Em França et al. (2012) são apresentadas as experiências adquiridas no ensino de Computação numa escola pública do Estado de Pernambuco, onde utilizou-se como metodologia de ensino a Computação Unplugged e o Scratch. No decorrer do trabalho referido, os autores detalham em como se deu a execução do projeto, no que se refere à avaliação, os alunos foram acompanhados continuamente, onde era observado o comportamento e o desempenho dos mesmos e a cada fim de aula era aplicado um questionário para obter informações relevantes, como por exemplo as dificuldades encontradas e a verificação do nível de absorção dos conteúdos abordados.

Silva et al. (2015) traz a discussão política e desperta a reflexão da inserção do ensino de Computação na educação básica juntamente com o relato de experiência de estagiários do curso de Licenciatura em Computação, o projeto foi realizado em três escolas públicas no ensino fundamental, onde foi proposto o desenvolvimento de jogos 2D utilizando a ferramenta Game Maker e o ensino de conceitos básicos de Ciências da Computação.

Diante das fundamentações teóricas abordadas, é notável os desafios que foram e ainda são enfrentados para a inclusão do Ensino de Computação no âmbito escolar, mas é evidente os benefícios que são proporcionados e as contribuições que a Computação tem para a educação e todas as suas áreas de conhecimento. Portanto, tendo em vista esse contexto, a Computação e seus respectivos conteúdos têm proporcionado momentos relevantes e diferenciados, trazendo a dimensão tecnológica para sala de aula, quebrando paradigmas e proporcionando o ensino e a aprendizagem significativa com o auxílio de softwares e ferramentas lúdicas.

3. Relato de Experiência

Como foi dito brevemente na primeira seção deste artigo, está seção tem como objetivo detalhar a experiência do estágio docente, que ocorreu no segundo semestre de 2017 na Escola de Aplicação, na turma do 6º ano B com participação de 12 alunos, na qual foram ministradas aulas da disciplina Computação Educacional, pelos discentes autores deste artigo, aulas estas que ocorriam duas vezes por semana, uma aula na segunda e outra na quarta com duração de uma hora e todas elas contando com a supervisão de pelo menos um dos professores de Estágio Supervisionado.

O início das atividades de estágio se deu pela elaboração de um planejamento baseado a partir de um documento denominado Programa de Componente Curricular, criado pelos próprios professores supervisores de estágio, documento este que serviu como referência para elaboração e execução das aulas de Computação Educacional.

Os conteúdos programados podem ser vistos a seguir na Tabela 1, onde estão subdivididos em três seções, que por meio da atuação dos discentes como professores, tem como objetivo construir e desenvolver competências e habilidades nos educandos a

partir dos conteúdos apresentados, fazendo isso de forma ativa através do desenvolvimento prático e teórico da disciplina.

Tabela 1. Conteúdo de Computação Educacional.

Conteúdo programático
1- Pensamento Computacional e Algoritmos 1.2 – Algoritmo utilizando linguagens visuais e de programação. 1.3 - Trabalhar a construção de soluções de problemas.
2 – Ecossistemas Digitais 2.1 – Tratar a Relação entre os Hardwares, Softwares e as camadas dos Sistemas Operacionais. 2.2 – Estrutura e funcionamento da internet.
3 – Culturas Digitais 3.1 - Utilização das tecnologias no contexto escolar, social e profissional trazendo seu risco, responsabilidade e impactos quando são utilizados. 3.2 - Manipulação de dados e seus softwares. 3.3 - Lógica de ordenamento de resultados e sua utilização para novas aprendizagens.

Inicialmente o conteúdo proposto pode parecer algo complexo, visto que dificilmente algum aluno estudou algo referente à Computação dentro da escola, em virtude disso, grande parte dos conteúdos desenvolvidos no planejamento da disciplina foram trabalhados de forma lúdica, por meio de atividades desplugadas, ou seja, um tipo de atividade oriunda da Computação, onde não há necessidade do uso de um computador para sua execução.

Também foram adotadas algumas outras práticas como o uso das tecnologias educativas a serem utilizadas pelos educandos de forma individual ou em equipe, como por exemplo: jogos educacionais, dinâmicas em sala de aula e atividades na Web. É válido destacar que a escola sede do projeto possui um laboratório ativo, o que possibilitou as aulas práticas laboratório.

3.1 Execução do Projeto

Em prol de tornar as aulas da disciplina algo notável e atraente ao olhar dos educandos, foi tido como objetivo se distanciar ao máximo da metodologia tradicional de ensino, onde o professor transmite informação e os educandos passivamente apenas absorvem essas informações. Desta forma, durante todas as aulas da disciplina os educandos eram sempre estimulados a participarem ativamente das aulas, seja por meio de um diálogo sobre algum conteúdo da aula, solucionar uma atividade no quadro e até mesmo dar a liberdade para eles relatarem alguma ideia, opinião ou questionamento que vinhesse a surgir durante as aulas.

Foram utilizadas juntamente com as aulas teóricas atividades práticas com o intuito de fixar o conteúdo proposto, sendo elas desplugadas, adaptadas para despertar nos alunos o raciocínio lógico. Sabendo disso, algumas das atividades desplugadas tendo como base o livro de Tim Bell et al (2011), foram: Cidade Enlameada, Happy Maps e Pintando números binários. Já para as atividades adaptadas, foram utilizadas: Estrela mágica e Reconhecendo os valores. Quanto às aulas práticas no laboratório, foram utilizadas os seguintes portais: Code.org, Blockly Games e Lightbot.

Figura 1. Educandos participando da atividade desplugada.

Nas aulas teóricas com atividades desplugadas representada pela Figura 1, foi percebido o real interesse dos educandos, visto que o engajamento deles nessas atividades era algo notável, pois eles sempre demonstravam curiosidade para saber que tipo de atividade nova seria aplicada em sala de aula, tudo isso foi relatado pelos próprios educandos, como pode ser visto pelas palavras de alguns deles “como é bom fazer esse tipo atividade, é bem mais legal aprender assim”.

Já nas aulas práticas, onde os educandos eram levados ao laboratório de informática da escola, o interesse deles aumentava ainda mais, e com softwares e sites educacionais de programação já citados, foi possível disseminar o Pensamento Computacional e a construção de algoritmos por meio de jogos de programação lúdicos que contém uma linguagem de programação visual e de fácil interpretação e aprendizagem, como é demonstrado na Figura 2, onde o aluno apresenta um algoritmo para solução de um problema na plataforma de programação Blockly Games.

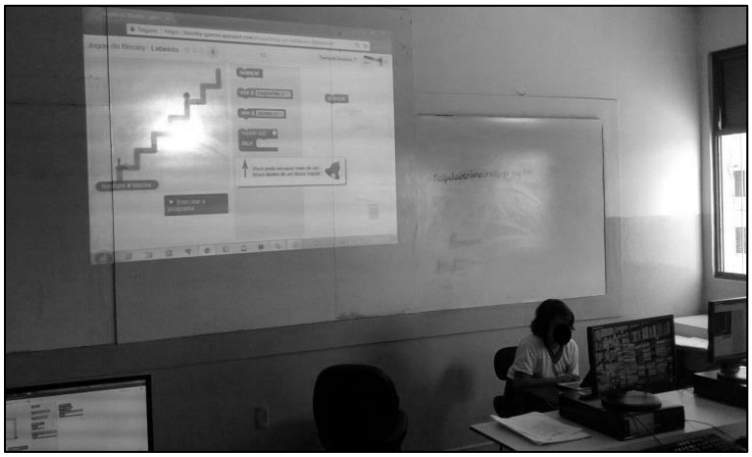


Figura 2. Educando apresentando seu algoritmo no Blockly Games.

Com a finalidade de dinamizar e sair um pouco da rotina, também foi pensado em uma dinâmica para simular em como as informações passam pelos componentes do computador até chegar realmente ao usuário, para isso foi formada uma fila com os alunos, onde o último da fila passava a informação para o próximo até chegar ao primeiro da fila, essa dinâmica foi utilizada a fim de demonstrar e contextualizar com os educandos em o quanto as informações são mudadas e adaptadas até chegar aos usuários.

Sendo assim, vale reforçar que todas essas estratégias foram utilizadas com o intuito maior de quebrar os paradigmas tradicionais, onde o professor é visto como o único "dono do saber", sendo assim, andamos lado a lado com os alunos, construindo um conhecimento mais significativo e mostrando que isso é possível através de diálogos e troca de informações, que consequentemente torna o processo de ensino e aprendizagem uma experiência rica e de qualidade em termos de Educação.

4. Limitações e Lições aprendidas

Tendo em vista toda trajetória vivenciada pelos autores deste artigo, é válido relatar que como esta foi uma experiência nova e desafiadora, tivemos nossas limitações que consequentemente ocasionaram no atraso do nosso projeto e também no adiantamento de alguns assuntos, atraso este que se deu por praticamente a ausência de aulas no mês de Outubro nas Segundas e Quartas.

Sabendo disto, é válido ressaltar que no mesmo dia que eram ministradas aulas no sexto ano A, também havia aula para o sexto ano B da mesma disciplina, mas com outros estagiários, sendo assim, em relação às aulas práticas, tivemos que revezar o uso do laboratório para que todos pudessem ter acesso aos computadores e as ferramentas online, já que só tem um laboratório na escola.

Enquanto profissionais ainda em formação, pode-se dizer com propriedade que todos os imprevistos que ocorreram durante a execução do projeto contribuíram fortemente para melhoria da metodologia dos estagiários, onde foi possível rever

também a didática e criar a habilidade de adaptar as atividades de acordo com a realidade dos alunos e sempre possuir um plano b, tornando assim o planejamento mais flexível e menos propício à situações que venham a prejudicar o projeto.

5. Resultados

Os resultados adquiridos foram notáveis, através do acompanhamento contínuo dos educandos, isso foi possível através da participação, desempenho e coerência ao responder as atividades propostas, também foram aplicadas duas atividades avaliativas que foram realizadas individualmente com base nos conceitos de Computação trabalhados durante as aulas, como por exemplo: Pensamento Computacional, Redes de Computadores, Sistemas Operacionais, Criptografia e Números Binários.

Tendo como base os resultados das duas atividades avaliativas, foi possível notar a evolução significativa dos educandos por meio de uma análise minuciosa das notas da atividade avaliativa I, em que a média total da turma foi de 6,8 e na atividade avaliativa II onde a média total aumentou para 9,0. Diante desse resultado, pôde-se concluir que houve um desempenho gradativo dos educandos durante o decorrer da disciplina, que é visivelmente notável através da comparação entre as médias das atividades avaliativas.

Além disso, durante as aulas provocamos a curiosidade dos alunos em relação a alguns assuntos, como: Linguagens de programação e Sistemas Operacionais, a fim de utilizar a curiosidade dos mesmos como uma estratégia para estimular seu interesse e solicitar pesquisas, assim fazendo com que o educando buscasse informações e trouxessem para sala, com o intuito de socializar o conhecimento adquirido com os colegas.

Também foi possível ver o pensamento crítico dos educandos quanto às tecnologias que agora estão tão inseridas na cultura do país e do mundo, por meio das aulas sobre Cultura Digital, nas quais eram debatidos temas como: O modo como às tecnologias influenciam no nosso modo de viver e de se comunicar, como ela pode e é usada no âmbito escolar, considerando suas vantagens e desvantagens. Os educandos conseguiram argumentar de forma bem interessante e construtiva nestas aulas, expondo suas opiniões e debatendo sobre os temas propostos, de fato foram momentos bastante prazerosos de serem vivenciados em sala de aula.

Diante de alguns depoimentos dados pelos professores/supervisores do Estágio Supervisionado II e da direção da escola, vale ser mencionado também o espaço que conquistamos e em como passamos a ser vistos pelas pessoas que compõem o âmbito escolar no qual estávamos inseridos, isso foi notável através do reconhecimento do trabalho que foi realizado e compromisso com os educandos que foi assumido desde o início do projeto de Ensino.

6. Considerações Finais

Durante esse período de estágio, pode-se perceber a evolução profissional dos estagiários que puderam atuar como professores titulares da disciplina de Computação Educacional.

O projeto que foi executado pode claramente ser considerado uma grande conquista, pois é do conhecimento de todos que ainda não há na grade curricular a Computação como disciplina nas escolas, e sabe-se de todas as lutas para que isso se firme e o quanto o caminho é longo para que a Computação seja inserida realmente como uma disciplina, sendo assim, o Estágio Supervisionado nos proporcionou essa vivência que tornou-se essencial para nossa formação.

Diante do trabalho que foi desenvolvido e relatado neste artigo, pode-se notar a necessidade e os benefícios da inclusão da Computação no âmbito escolar, isso foi perceptível através das competências e habilidades que foram despertadas nos educandos. Tendo em vista que a tecnologia já faz parte do cotidiano dos alunos, é importante mostrar que a mesma não só serve para momentos de lazer, mas se usada de forma correta torna-se uma grande ferramenta auxiliadora no processo de ensino e aprendizagem.

Referências

Bell, T.; Witten, I. e Fellows, M. (2011). “Computer Science Unplugged – Ensinando Ciência da Computação sem o uso do Computador”. Tradução de Luciano Porto Barreto, 2011. Disponível em: <http://csunplugged.org/>. Acesso em: 26 de agosto 2017.

França, R. T., Silva, W. C., Amaral, H. J. C. (2012). “Ensino de Ciência da Computação na Educação Básica: Experiências, Desafios e Possibilidades”. XX Workshop sobre Educação em Computação (WEI).

Reis, F. M., Oliveira, F. C. S., Martins, D. J. S., Moreira, P. R. (2017). “Pensamento Computacional: Uma Proposta de Ensino com Estratégias Diversificadas para Crianças do Ensino Fundamental”. XXIII Workshop de Informática na Escola (WIE 2017).

Silva, S. F., Barbosa, A. F., Souza, A. A., Silva, E. G., Oliveira, M. L. S., Neto, S. R. S., and do Santos, O. W. (2015). “Relato de Experiência de Ensino de Computação no Ensino Fundamental em Estágio Supervisionado da Universidade de Pernambuco no Campus Garanhuns”. Workshop sobre Educação em Computação.

Silva, V., Silva, K., França, R. T. (2017). “Pensamento computacional na formação de professores: experiências e desafios encontrados no ensino da computação em escolas públicas”. XXIII Workshop de Informática na Escola (WIE 2017).

Sociedade Brasileira de Computação. “Referenciais de Formação em Computação: Educação Básica”, 16 de agosto de 2017. Disponível em: <http://www.sbc.org.br/noticias/10-slideshow-noticias/1996-referenciais-de-formacao-em-computacao-educacao-basica>. Acesso em: 25 de agosto de 2017.

Torcate, A. S., Farias, M. U. F., Santos, H. R. M. (2017). “Relato de Experiência do PIBID: Promovendo o Ensino de Computação de forma interdisciplinar com Português no Ensino Fundamental”. XXIII Workshop de Informática na Escola (WIE 2017).

Percepção dos Estudantes sobre a Implantação de uma Disciplina Regular de Pensamento Computacional em um Colégio de Educação Básica

André Luís Alice Raabe[123], Elieser Ademir de Jesus[1], Eduardo Alves da Silva, Natália Ellery Ribeiro Couto[3]

[1]Laboratório de Inovação Tecnológica na Educação - CTTMar - UNIVALI
[2] Programa de Pós-Graduação em Educação
[3] Mestrado em Computação Aplicada

{raabe, eas, elieser}@univali.br, nataliaellery@gmail.com

Abstract. *This paper presents the students' perception about the implementation of a regular Computational Thinking discipline in a Basic Education school. The discipline is offered from the 6th year of elementary school to the 3rd year of high school. The teaching strategies applied during the classes are presented, as well as the data collected from students' different perceptions about Computational Thinking. Most part of the students perceive the instigation of logical reasoning, creativity, working together, the importance of applying the Computational Thinking in their personal lives and in society. Other results synthesize the opinions about the difficulties of the contents and activities, the interdisciplinarity and the technological empowerment.*

Resumo. *Este artigo apresenta a percepção dos estudantes acerca da implantação de uma disciplina regular de Pensamento Computacional em uma escola de Educação Básica. A disciplina é ofertada do 6º ano do Ensino Fundamental ao 3º ano do Ensino Médio. São apresentadas as estratégias de ensino aplicadas durante as aulas, bem como os dados coletados sobre diferentes percepções dos estudantes sobre o Pensamento Computacional. Em sua maioria, os alunos percebem a instigação do raciocínio lógico, criatividade, trabalho em conjunto, a importância da aplicação do Pensamento Computacional. Outros resultados sintetizam as opiniões sobre as dificuldades das atividades, a interdisciplinaridade e o empoderamento tecnológico.*

1. Introdução

O termo Pensamento Computacional (PC) surgiu em um artigo de opinião publicado por Wing em 2006, sendo conceituado como a síntese do conjunto de habilidades cognitivas que os profissionais da área da Computação geralmente desenvolvem em seu processo de formação. Em seu artigo ela sugere que, assim como habilidades fundamentais como a leitura, escrita e aritmética, que são as referências da alfabetização americana, o PC deve ser encorajado desde a Educação básica (Wing 2006). Desde então surgiram diversas iniciativas para incluir a abordagem do PC em escolas de educação básica.

Após 10 anos da publicação de Wing apresentando o PC, Heintz *et al* (2016) conduziram uma pesquisa para sintetizar a implantação de disciplinas que envolvem PC e Ciência da Computação desde a alfabetização até o Ensino Médio (EM) em 10 países:

Austrália, Inglaterra, Estônia, Finlândia, Nova Zelândia, Noruega, Suécia, Coréia do Sul, Polônia e Estados Unidos. Alguns destes países incorporaram a Ciência da Computação no currículo oficial, enquanto outros estão no processo de incorporação. O sistema educacional é diferente em cada país, mas através da análise dos autores é possível perceber no Ensino Fundamental (EF) uma tendência em apresentar a Ciência da Computação por meio do PC, lógica de programação ou competências digitais. Enquanto no EM a Ciência da Computação é apresentada de forma mais ampla, assim como seus impactos na sociedade.

Na literatura estrangeira existem trabalhos avaliando os resultados da implantação de disciplinas envolvendo a computação. Sabbagh *et al* (2017) apresentam um relatório de experiência sobre o currículo de computação no EF II em Qatar. O currículo aborda conceitos de programação, análise, pensamento lógico, e habilidades de resolução de problemas no contexto de criação de animações. Foram realizadas entrevistas em grupo e aplicados questionários para investigar a satisfação com as aulas e as impressões sobre as ferramentas utilizadas.

Nos Estados Unidos, DeLyser *et al* (2016) investigaram 4 escolas que possuem a computação em sua matriz curricular. Foram aplicados questionários com perguntas referentes ao efeito pedagógico, experiência com o professor, engajamento dos colegas na sala de aula e interesse dos estudantes no assunto. Já Dettori *et al* (2016) apresentam em seu trabalho a experiência de realizar um *workshop* com professores tendo como objetivo a disseminação do currículo Exploring Computer Science. Após o *workshop*, os professores aplicaram o currículo em suas aulas, e 7 destes professores aplicaram questionários para investigar a percepção de seus estudantes sobre as áreas da computação, utilidade do campo da computação e satisfação com o currículo ECS.

No Brasil também existem iniciativas para abordar o PC em escolas de educação básica. Aono *et al* (2017) realizaram um minicurso através do *Scratch* enquanto Reis *et al* (2017) utilizaram o site *code.org* e as atividades da Computação Desplugada em um curso para estudantes do EF. O SESI-PB, através de seu Programa Educação Básica articulada com Educação Profissional, integrou ao seu currículo o ensino de Robótica para o 1º ano do EM. Souza *et al* (2016) apresentam os resultados da abordagem do PC integrado à disciplina de Robótica. Apesar de existirem diversas abordagens, elas não acontecem no formato de uma disciplina específica sobre PC presente na matriz curricular. Este formato (disciplina regular de PC) traz desafios ainda não explorados nas publicações encontradas e são importantes para o amadurecimento da área no Brasil.

Este artigo relata a percepção dos estudantes sobre a implantação do PC como disciplina obrigatória no EF e EM no Colégio de Aplicação da Universidade do Vale de Itajaí (UNIVALI) . A disciplina é ofertada para todos os estudantes do 6º ano do EF até o 3º ano do EM em 3 unidades do colégio, atendendo aproximadamente 700 estudantes. O artigo também apresenta as estratégias e ferramentas de ensino utilizadas pelos professores responsáveis pelas turmas em que os dados foram coletados.

O artigo está organizado como segue: Na Seção 2 detalha-se a estratégia de implantação do PC no colégio. Na seção 3 é apresentada a metodologia utilizada para coletar os dados referentes às percepções dos alunos sobre a disciplina. Por fim, na seção 4 são apresentadas as conclusões do trabalho e recomendações para gestores interessados em implantar uma disciplina semelhante.

2. Implantação da Disciplina de Pensamento Computacional no Colégio de Aplicação da UNIVALI

Em 2017, o Colégio de Aplicação da UNIVALI (CAU) reorganizou sua matriz curricular com o objetivo de aderir às novas políticas públicas do Brasil. Desta maneira foi adotada a modalidade de ensino integral para os alunos entre o 6º ano do EF II e 3º ano do EM. Com esta oportunidade surgiram novas disciplinas obrigatórias e optativas, fazendo com que pesquisadores de Informática na Educação da UNIVALI realizassem a recomendação de inclusão de duas disciplinas.

A primeira disciplina, denominada Laboratório Maker, é optativa e aborda a vivência de projetos "mão na massa" incorporando diversas modalidades como marcenaria, costura, robótica, impressão 3D, entre outros. A segunda disciplina, denominada Pensamento Computacional, estimula a fluência em computação nos estudantes para que possam aplicar os conhecimentos e recursos em resolução de problemas e também a criação de aplicações por meio da programação.

A disciplina de PC passou então a ser oferecida de forma obrigatória para todos estudantes do 6º ano do EF ao 3º ano do EM nas três unidades do Colégio, localizadas nos municípios de Itajaí, Balneário Camboriú e Tijucas no estado de Santa Catarina.

2.1. Definição dos conteúdos

No primeiro ano de implantação a matriz curricular foi criada para que todos os estudantes pudessem ter um contato inicial com a disciplina, já que estariam tendo os primeiros contatos com o PC. Conforme os estudantes forem avançando para o ano seguinte, a matriz curricular será reformulada possibilitando uma progressão do 6º ano do EF até o 3º ano do EM. No primeiro ano as ementas foram divididas em 3 grupos:

- EF II - 6º e 7º ano - Estudo dos dispositivos computacionais, Mecanismos de busca, Noção de algoritmos, Práticas de computação, Jogos de lógica, Construção de narrativas usando programação com blocos;
- EF II - 8º e 9º ano - Fundamentos de computação, Funcionamento dos mecanismos de busca, Algoritmos com condições e repetições, Animação e som, Construção de animações usando programação com blocos;
- EM - Fundamentos de computação, Funcionamento dos mecanismos de busca, Algoritmos com condições e repetições, Animação e Som, Construção de animações usando programação, Algoritmos que manipulam dados, Projetos de aplicações.

2.2. Estratégias e Ferramentas Utilizadas nas Aulas

A disciplina de PC foi ministrada por 2 professores e 1 professora: o primeiro responsável por 4 turmas de 6º e 7º ano do CAU de Itajaí; o segundo responsável por 10 turmas de 8º ano do EF a 3º ano do EM do CAU de Itajaí; e a professora responsável por 10 turmas de 8º ano do EF a 3º ano do EM do CAU de Balneário Camboriú. As aulas possuem duração de 45 minutos e acontecem uma vez na semana. Cada professor criou seus planos de ensino e utilizou diferentes estratégias para trabalhar os conteúdos da ementa definidos para a disciplina, as quais são apresentadas a seguir.

2.2.1. Estratégias e Ferramentas utilizadas com o 6º e 7º anos

As turmas dos 6º e 7º anos (4 turmas no total) vivenciaram pela primeira vez o PC

enquanto disciplina regular. Grande parte da disciplina aconteceu sob a forma de pequenos projetos práticos, realizados na sala de aula ou no laboratório de informática ao longo de algumas semanas (1 encontro por semana).

Já na primeira aula os estudantes foram desafiados a descobrir o que era "Pensamento Computacional" e ficaram muito felizes ao saber que poderiam usar o telefone celular para fazer a pesquisa. Em seguida, o professor apresentou uma noção geral de algoritmo e a turma realizou uma atividade na qual cada dupla escreveu a sequência de passos (usando setas direcionais) para sair da sua carteira e chegar até a porta da sala (o chão da sala de aula é quadriculado).

Desde a primeira aula ficou bastante evidente a pouca capacidade dos estudantes (entre 10 e 12 anos) de se manterem focados durante explicações orais. Por conta disso, todo o restante da disciplina foi pensada com o objetivo de reduzir (e às vezes até eliminar) tais explicações, substituindo-as por pesquisas e consulta de materiais (tutoriais, video aulas, etc) selecionados e indicados pelo professor.

Na aula seguinte, ainda em sala de aula, foram utilizados jogos de lógica (BloxorZ[1] e LightBot[2]), os estudantes anotavam os algoritmos usados para passar alguns níveis, e depois trocavam os algoritmos entre si e executavam/testavam os algoritmos dos colegas buscando, por exemplo, passos faltantes para completar os níveis do jogo.

Na terceira aula a turma foi para o laboratório de informática (algo muito aguardado por eles) e os estudantes resolveram (em duplas) o desafio do jogo MineCraft no site *code.org*. Esta atividade teve uma enorme importância em grande parte do que aconteceu em seguida na disciplina. O *Scratch*[3] foi utilizado como principal ferramenta durante todo o ano e o desafio do site code.org foi, na visão do professor, a introdução mais intuitiva possível à ideia da programação em blocos (noção fundamental para a utilização do *Scratch*).

Na aula seguinte o *Scratch* foi apresentado para a turma, e após a experiência com o site code.org os estudantes não tiveram dificuldades para compreender o funcionamento da ferramenta, conseguindo realizar atividades simples (animar personagens, fazer o personagem "falar", etc). Nas 3 aulas seguintes os estudantes (em duplas ou trios) trabalharam no primeiro projeto prático da disciplina: uma história interativa feita no *Scratch*, incluindo diálogos, animação de personagens e troca de tela de fundo.

Nas aulas seguintes o professor fez breves explicações sobre como movimentar os personagens do *Scratch* usando o teclado do computador e como verificar se os objetos estavam "tocando" uns nos outros. Em seguida os estudantes dedicaram as aulas restantes do 1º bimestre para trabalhar (em duplas ou trios) no 2º projeto prático: um jogo simples (tema livre) no qual seria possível controlar pelo menos um personagem e deveria haver teste de colisão. Durante a realização destes projetos práticos o professor auxiliou cada equipe com as dúvidas específicas do seu projeto. Apesar de muito intensa e cansativa esta abordagem mostrou-se mais produtiva do que a tradicional "explicação que vale para todos", já que este público (entre 10 e 12 anos de idade) demonstrou desde o início uma grande dificuldade em acompanhar explicações orais.

[1] http://www.coolmath-games.com/0-bloxorz
[2] http://lightbot.com/
[3] https://scratch.mit.edu/

Ao final do segundo projeto prático com o *Scratch* as duplas apresentaram seus trabalhos para toda a turma. As quatro turmas mostraram-se muito empolgadas pela possibilidade de apresentar seus projetos para os colegas, tendo o jogo projetado em uma tela grande através do data show.

Após a realização dos dois primeiros projetos práticos os estudantes estavam visivelmente cansados da ferramenta *Scratch*. Por este motivo, o projeto realizado a seguir foi uma máquina de Rube Goldberg, uma atividade totalmente desplugada. Em um primeiro momento os estudantes (divididos em quartetos) esboçaram em papel como seria a máquina, quais materiais seriam usados para construí-la e quem ficaria responsável por trazer cada material necessário para a construção. Na aula seguinte, para surpresa do professor, a sala estava repleta de caixas de dominó, papelão, bolinhas de gude, canos de PVC, patins, ventilador, secador de cabelo, pistolas de cola quente, etc., todos os materiais trazidos de casa pelos estudantes (ver Figura 1).

Figura 1. Máquina de Rube Goldberg feita com brinquedos no chão da sala

As aulas seguintes foram dedicadas à montagem e teste da máquina de Rube Goldberg (cada equipe projetou sua própria máquina) na sala de aula, usando todos os objetos disponíveis no ambiente (chão, mesas, carteiras, quadro, etc.). Além de desenvolver a resiliência e a capacidade de trabalhar em equipe esta atividade propiciou um intenso exercício de aprimoramento da habilidade de resolução de problemas. Todas as equipes se depararam com inúmeras situações nas quais a máquina real não funcionava como aquela desenhada no papel, tendo que buscar alternativas e em algumas situações até mesmo pensar em outra máquina.

Este projeto gerou excelentes resultados, muito engajamento por parte dos estudantes e também de alguns pais. A grande dificuldade observada foi a curta duração das aulas (aproximadamente 45 minutos), forçando a montagem e desmontagem das máquinas (algumas relativamente grandes) em cada aula. Ao final do projeto as máquinas em funcionamento foram filmadas, e cada equipe teve seu projeto avaliado pelos colegas de sala e também pelas outras 3 turmas, um momento de muito alegria e aprendizagem. Os vídeos das máquinas construídas estão disponíveis em http://pc-cau.appspot.com/.

O último projeto prático do ano letivo foi um projeto no *Scratch*, no qual os estudantes (em duplas) programaram uma simulação de uma das máquinas de Rube Goldberg construídas no trabalho prático anterior. Novamente cada equipe apresentou (geralmente com muito orgulho) seu trabalho para toda a turma.

De forma geral, a abordagem baseada em pequenos projetos práticos com temas escolhidos pelos estudantes mostrou-se bastante satisfatória, tanto do ponto de vista dos estudantes quando do ponto de vista do professor. Vale mencionar que, ao contrário do que era esperado pelo professor, as diferenças entre as turmas de 6º e 7º anos foi muito significativa. A maturidade e capacidade de lidar com assuntos mais abstratos era notoriamente menor com o 6º ano, exigindo do professor uma série de adaptações nas estratégias utilizadas.

Em uma das turmas havia uma estudante com deficiência visual. Isto gerou uma série de desafios no planejamento das aulas, já que o *Scratch* (principal ferramenta utilizada) não se integra com os softwares leitores de tela usados pelos deficientes visuais. Isto reduziu drasticamente as possibilidades para esta estudante, cuja aprendizagem depende muito de *feedback* auditivo.

2.2.2. Estratégias e Ferramentas usadas com 8º e 9º anos e EM

As aulas do 8º ano do EF ao 3º ano do EM da unidade de Itajaí foram conduzidas pela mesma professora. Ao iniciar o ano letivo foram criados planos de aula baseados nas duas primeiras atividades do material de apoio da Ciência da Computação Desplugada[4]: Números Binários e Colorindo através de Números para tratar do tópico da disciplina PC: "Fundamentos de Computação". Em seguida os tópicos da ementa "Algoritmos com Condições e Repetições", "Animação e Som" e "Construção de Animações usando programação com Blocos" foram abordados simultaneamente ao utilizar a ferramenta *Scratch*. Os estudantes iniciaram as atividades no *Scratch* através da construção de animações, onde foram aplicados na prática os conceitos de algoritmos com repetição para programar o movimento e aparência dos componentes da animação. Da mesma forma o som foi abordado para complementar as animações construídas pelos estudantes.

O conceito de *condições* foi abordado através da criação de jogos com a ferramenta *Scratch*, os estudantes utilizaram algoritmos com condições para verificar colisão entre personagens, acionar eventos através da verificação de cliques do mouse ou teclas do teclado pressionadas. A seguir foi introduzido um novo tópico: "Algoritmos que manipulam dados", o qual não consta na ementa do EF II, mas que foi introduzido a estes alunos devido à facilidade dos mesmos em realizar as atividades referentes aos tópicos anteriores. Ao observar o desempenho dos estudantes do EF II foi percebido que eles conseguiram realizar as mesmas atividades que os estudantes do EM.

No fim do segundo bimestre foi possível completar os tópicos da ementa, então por decisão da professora, os bimestres seguintes foram utilizados para trabalhar os mesmos assuntos porém com o uso de ferramentas mais complexas, aprofundando os conhecimentos adquiridos e demonstrando aos estudantes que diversas ferramentas e linguagens de programação possuem a mesma base conceitual, a qual também é aplicada a processos de pensamento utilizados através do PC como: abstração, reconhecimento de padrões, sequenciamento de passos, entre outros. As aulas se deram em torno da ferramenta *AppInventor*[5] que permite o desenvolvimento de aplicativos para dispositivos móveis.

[4] https://csunplugged.org/en/
[5] http://appinventor.mit.edu

As primeiras aulas do terceiro bimestre envolveram o desenvolvimento de projetos simples para entender a nova plataforma, a qual é voltada para componentes de dispositivos móveis. Foi possível perceber que os estudantes tiveram dificuldades com o novo ambiente, e não se mostraram empolgados com a criação de projetos pré-definidos pela professora. Para tornar as aulas mais agradáveis e produtivas foram realizadas votações para que cada turma pudesse escolher modelos de projetos para desenvolver, como por exemplo: jogos para celular, aplicativos de bate-papo, quiz, entre outros.

A abordagem de ensino escolhida para o terceiro bimestre acabou se tornando complexa demais para ser finalizada em um bimestre, então, com as turmas que concluíram as atividades - uma turma do 3º ano, duas turmas do 2º ano e duas turmas do 1º ano-, foi possível abordar uma terceira ferramenta: o *Portugol Studio*[6], enquanto as outras turmas - duas do 8º ano, duas do 9º e uma turma do 3º ano - continuaram a trabalhar nos projetos do *AppInventor*. Esta nova ferramenta trouxe uma nova realidade: a programação escrita. Os estudantes não tinham mais os comandos em blocos para arrastar e encaixar, nem a facilidade gráfica de montar o projeto arrastando os componentes. Com esta ferramenta foram abordados os conceitos de manipulação de dados através de variáveis e o uso de desvios condicionais. Ao invés de trabalhar com foco em pequenos projetos, com o *Portugol Studio* foram lançados desafios de programação.

Através das aulas ministradas foi possível perceber que tanto os alunos do EF II quanto os alunos do EM foram capazes de realizar as mesmas atividades propostas. Ainda foi possível observar casos em que alunos do EF II alcançaram resultados mais complexos do que alunos do EM. Através desta observação seria possível sugerir que uma disciplina de PC não seja necessariamente separada por idades, e sim por nível de habilidade. Quanto aos temas abordados, o EF II se mostrou bastante empolgado na criação de jogos e animações, enquanto o EM apresentou maior interesse pelo *Portugol Studio*, ferramenta que proporciona programação através de linguagem escrita ao invés de blocos.

Um fator que chamou a atenção foi a integração das estudantes nas atividades e participação das aulas: em algumas turmas as estudantes mostraram maior interesse do que os estudantes. Atualmente ainda existem poucas garotas cursando o ensino superior na área da computação (Santos 2017), o que pode sugestionar que garotas não gostam desta área ou que possuem dificuldades com este tipo de conteúdo. Entretanto, não foi o que aconteceu durante as aulas de PC.

3. Percepção dos Estudantes sobre a Disciplina de PC

Foram utilizados dois questionários para a coleta de dados em relação à percepção dos estudantes sobre a disciplina. O *Questionário 1*[7] teve como objetivo uma análise exploratória com perguntas de respostas abertas sobre a opinião dos estudantes em relação aos pontos positivos, negativos e melhorias que poderiam acontecer nas aulas. Foi aplicado no 2º bimestre de 2017 com as turmas do 8º ano do EF ao 3ºano do EM da unidade de Itajaí. A escolha da amostra para o *Questionário 1* ocorreu por questões de logística, já que estas turmas eram ministradas pela mesma professora. O questionário foi facultativo e em anonimato - com o objetivo de que os estudantes respondessem de

[6] http://lite.acad.univali.br/portugol/
[7] https://bit.ly/2jloka4

forma sincera sem o receio de serem prejudicados posteriormente. No *Questionário 2*[8], as questões foram elaboradas com base nas respostas analisadas do *Questionário 1*, com a finalidade de investigar em maior profundidade a opinião dos estudantes em relação aos tópicos sugeridos por eles mesmos em suas respostas. Este questionário também foi facultativo e em anonimato, aplicado no último dia de aula e contou com a resposta de 403 estudantes do 8º ano do EF ao 3ºano do EM do CAU de Itajaí e Tijucas, os estudantes do CAU de Balneário Camboriú não participaram por falta de disponibilidade no período de encerramento das aulas.

Os detalhes sobre cada questionário aplicado aos estudantes são apresentados a seguir.

3.1. Questionário 1

O Questionário 1 foi composto por 3 questões abertas. Nas duas primeiras os estudantes descreveram os pontos positivos e negativos sobre PC enquanto disciplina obrigatória, e na Pergunta 3 sugeriram melhorias para as aulas.

Um total de 138 estudantes responderam ao questionário. Foram criadas categorias baseadas nas respostas textuais dos estudantes e algumas respostas acabaram se enquadrando em mais de uma categoria, já que a pergunta sugere uma resposta aberta. Por exemplo, a resposta "*O aumento do raciocínio lógico e a quebra da 'sala de aula'. "* foi enquadrado na categoria *Estimular o pensamento lógico e/ou criatividade* e também na categoria *Dinâmica diferenciada das aulas tradicionais* - neste caso a resposta de um estudante contabilizou para duas categorias, então o total de itens categorizados ultrapassa o número de respostas dos estudantes. A Tabela 1 apresenta as categorias de respostas para a Pergunta 1 (*Descreva algo positivo de consequência da disciplina de Pensamento Computacional)* e a sua frequência nos dados coletados.

Tabela 1. Categorias para as respostas da Pergunta 1

Categoria	Frequência	Percentual
1 - Aprender coisas novas	82	48.52%
2 - Dinâmica diferenciada das aulas tradicionais	51	30.17%
3 - Estimular pensamento lógico e/ou criatividade	27	15.97%
4 - Influenciar a realizar graduação na área	1	0.59%
5 - Não opinou	8	4.73%
Total	169	100%

A categoria *Aprender coisas novas* aparece mais frequentemente entre as respostas com 48.52%, o que é o resultado esperado já que este foi o primeiro ano de implantação da disciplina e os estudantes não possuíam conhecimentos sobre o assunto através da escola.

A categoria *Dinâmica diferenciada das aulas tradicionais* aparece em 30.17% das respostas, podendo ser justificada pelo fato de a maioria das aulas serem realizadas no laboratório de informática e em duplas, e normalmente requerendo mais "mão-na-massa" do que a atenção voltada para o professor explicando novos conceitos.

A categoria *Estimular pensamento lógico e/ou criatividade,* com 15.97% das respostas, está relacionada aos trabalhos e atividades realizados durante as aulas. Como

[8] https://goo.gl/forms/AhoagSnTwHHiUCWS2

em muitas dessas atividades o tema era livre os estudantes utilizavam a criatividade para criar projetos relacionados do seu interesse.

A categoria *Influenciar a realizar graduação na área* foi apontada em apenas uma das respostas, ou seja, apenas um estudante mencionou este argumento como ponto positivo.

Por fim, *Não opinou* foi a categoria utilizada para aqueles estudantes que responderam o questionário, porém não expressaram suas opiniões sobre os pontos positivos da disciplina, representando 4.73% das respostas.

A Pergunta 2 teve como objetivo analisar os pontos negativos que a aula pode ter proporcionado aos estudantes em sua vida acadêmica, já que este estudo tem como objetivo analisar a percepção dos estudantes como um todo, não apenas os benefícios. É necessário descobrir os pontos negativos para saber em quais pontos as aulas poderiam ser reformuladas, quais estratégias de ensino podem não funcionar como desejado. A Tabela 2 apresenta as categorias criadas a partir das respostas para a segunda pergunta: *Descreva algo de negativo de consequência da disciplina de Pensamento Computacional*.

Tabela 2. Categorias para as respostas da Pergunta 2

Categoria	Frequência	Percentual
1 - Estratégia de ensino	40	25.31%
2 - Não vêem importância	29	18.35%
3 – Infraestrutura	13	8.22%
4 - Não opinou	60	37.97%
5 – Dificuldade	10	6.32%
6 - Comportamento dos colegas	6	3.79%
TOTAL	158	100%

Com 37.97% do total, a maioria dos estudantes não expressou sua opinião. A próxima categoria mais frequente é a *Estratégia de ensino* com 25.31% das respostas. Analisando as respostas foi possível perceber muitas críticas sobre o uso excessivo da ferramenta *Scratch*, utilizada em 15 aulas (aproximadamente 4 meses). De acordo com as respostas dos estudantes este tempo utilizando uma mesma ferramenta tornou as atividades repetitivas e cansativas.

A categoria *Não vêem importância* apareceu em 18.35% das respostas, o que se refletia na sala de aula, pois alguns estudantes afirmaram que não reconheciam a importância da disciplina em sua vida acadêmica ou futuro profissional.

A *Infraestrutura* foi mencionada como ponto negativo para 8.22% dos estudantes. Nesta categoria foram encontradas respostas sobre o aumento ou diminuição da carga horária, tornar a disciplina opcional, a distância entre o laboratório e a sala de aula, a necessidade de um professor extra, entre outros.

A categoria *Dificuldade* envolveu 6.32% das respostas e 3.79% das respostas trataram da categoria *Comportamento dos colegas* devido à bagunça excessiva que ocorria durante as aulas.

A Pergunta 3 foi criada com o intuito de entender como melhorar os pontos negativos apresentados pelos estudantes. Por ter sido aplicado no segundo bimestre, o questionário também poderia ser utilizado para melhorar as aulas do terceiro e quarto

bimestres. A Tabela 3 apresenta as categorias criadas para as respostas da pergunta *Existe algo na disciplina de Pensamento Computacional que você acha que poderia melhorar? O quê?*

Tabela 3. Categorias para as respostas da Pergunta 3

Categoria	Frequência	Percentual
1 - Sem opinião	48	33.56%
2 - Alterar estratégia de ensino	78	54.54%
3 - Alterar a infraestrutura	9	6.29%
4 - Rigidez comportamental	7	4.89%
5 – Tudo	1	0.69%
TOTAL	143	100%

A maioria das respostas envolveram a sugestão de alterar a ferramenta utilizada nas atividades de sala de aula na categoria *Alterar estratégia de ensino* com 54.54% das respostas. Este resultado é consistente com as respostas relacionadas aos pontos negativos mencionados na Pergunta 2, na qual a segunda categoria mais frequente também é a *Estratégia de ensino*. Um total de 33.56% dos respondentes não apresentou opinião sobre possíveis melhorias para as aulas.

As respostas que incluíram a categoria *Alterar a infraestrutura* consistiram em 6.29% do total, sugerindo alterar o tempo de aula, o auxílio de um segundo professor para sanar as dúvidas específicas de cada estudante ou transformar a disciplina em opcional. Por fim, a categoria *Rigidez comportamental* totalizou 4.89% das respostas, sugerindo tratamento mais duro com estudantes agitados. Um estudante sugeriu que tudo deveria ser melhorado, representando 0.69% dos respondentes.

3.2. Questionário 2

O Questionário 2 foi composto por 14 questões as quais foram respondidas através de uma escala numérica com valores de 0 a 5. O valor 5 representando "*Muito*", o valor zero representando "*Nada*", o valor 1 representando "*Pouco*", e os números 2, 3 e 4 representando valorações intermediárias entre pouco e muito. As perguntas foram elaboradas com base nas categorias criadas a partir das respostas do Questionário 1. A realização do questionário foi facultativa e também se deu de forma anônima. O questionário foi aplicado no último dia de aula e contou com a resposta de 403 estudantes do CAU de Itajaí e Tijucas.

A Questão 1 e 2 investigaram o uso da criatividade e o estímulo do raciocínio lógico na disciplina de PC. Nos dois casos a resposta mais frequente foi 5 - *Muito*. Isso pode ser confirmado pela observação em sala de aula, os estudantes tiveram bastante liberdade para escolher diferentes temas para a criação de seus projetos e o pensamento lógico estava presente nas construções de algoritmos e resolução de problemas.

A Questão 3 tinha o objetivo de compreender as diferenças da disciplina de PC em comparação com as outras disciplinas regulares. Novamente a resposta mais frequente foi 5 - *Muito*. Este resultado pode ser justificado pelo frequente uso do laboratório de informática, trabalhos práticos em dupla e atividades voltadas a desenvolvimento de projetos.

A Questão 4 abordou o interesse pessoal em aprender a fazer jogos, aplicativos ou programar. Essa questão foi elaborada porque no Questionário 1 apareceram 29

respostas enquadradas na categoria *Não vêem importância*, apontada como ponto negativo. As respostas mostraram que 162 estudantes acharam *Muito* (5) interessante aprender a fazer jogos, aplicativos ou programar, sendo esta a resposta mais frequente, enquanto apenas 17 respondentes (4.2%) não demonstraram interesse. A Figura 2 apresenta a distribuição das respostas.

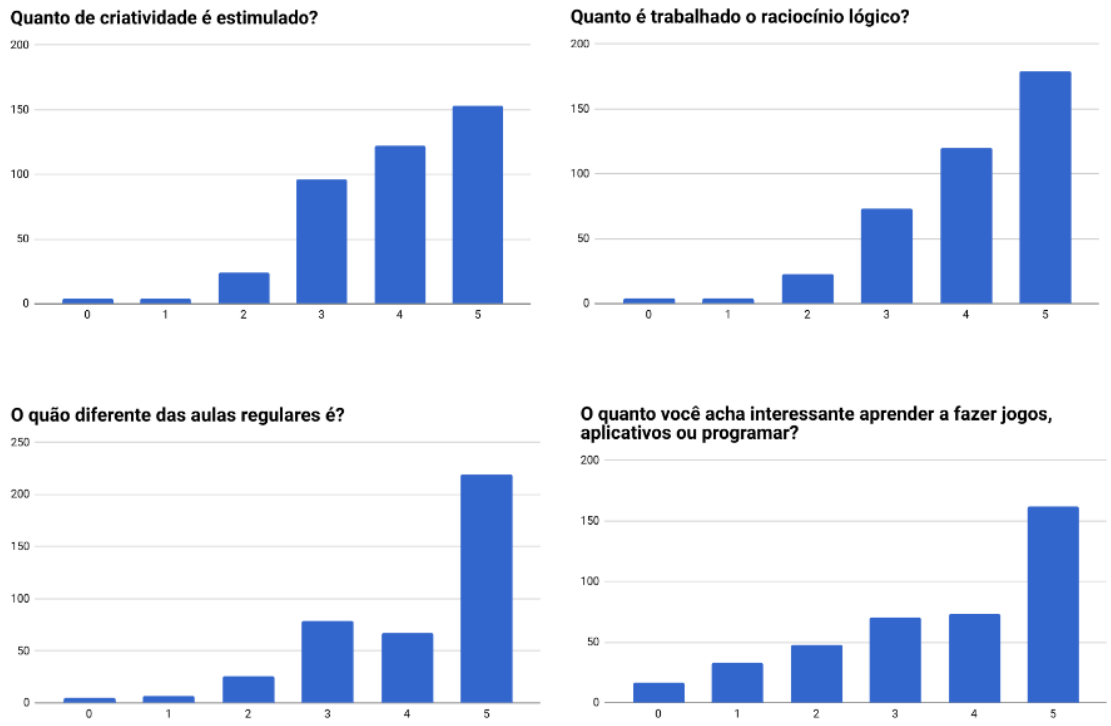


Figura 2. Respostas da Questão 1 (superior esquerdo), Questão 2 (superior direito), Questão 3 (inferior esquerdo) e Questão 4 (inferior direito)

A Questão 5 trata da experiência em realizar trabalhos em equipe devido ao número limitado de computadores nos laboratórios de informática. A resposta mais frequente, com 137 respostas, foi novamente 5 (*Muito*). Através da observação foi possível perceber estudantes discutindo e planejando estratégias para desenvolver os projetos e resolver as atividades propostas.

A Questão 6 tratou da dificuldade em realizar as atividades e entender o conteúdo na disciplina de PC, trazida no Questionário 1 por 10 respondentes como um ponto negativo da disciplina. A resposta mais frequente foi o grau 3, valor intermediário entre muito difícil (5) e pouco difícil (1). Durante as aulas era normal observar conversas paralelas e estudantes focando em outras coisas ao invés das explicações, isso dificultou um pouco a compreensão do conteúdo em diversos momentos.

A Questão 7 foi elaborada para tentar compreender quão boa a estratégia de ensino foi para a disciplina, já que este foi um ponto negativo apontado por 40 estudantes no Questionário 1. A resposta mais frequente foi 5, com 145 votos. Aqui se percebe uma mudança de opinião dos estudantes em relação a estratégia utilizada. Esta mudança de opinião pode ter ocorrido pelo fato de que a ferramenta que foi apontada

como repetitiva e cansativa (*Scratch*) foi substituída por outras.

A Questão 8 tinha como objetivo investigar se a disciplina de alguma forma auxiliou na aprendizagem de outras disciplinas. A resposta *não* (0) apareceu em 23.1% do total, mas considerando que todas as outras alternativas afirmam que há algum auxílio mesmo que este seja pouco (1), é possível então afirmar que 76.9% dos estudantes percebem contribuição (ainda que pouca) do PC em outras disciplinas. A síntese destes dados é apresentada na Figura 3.

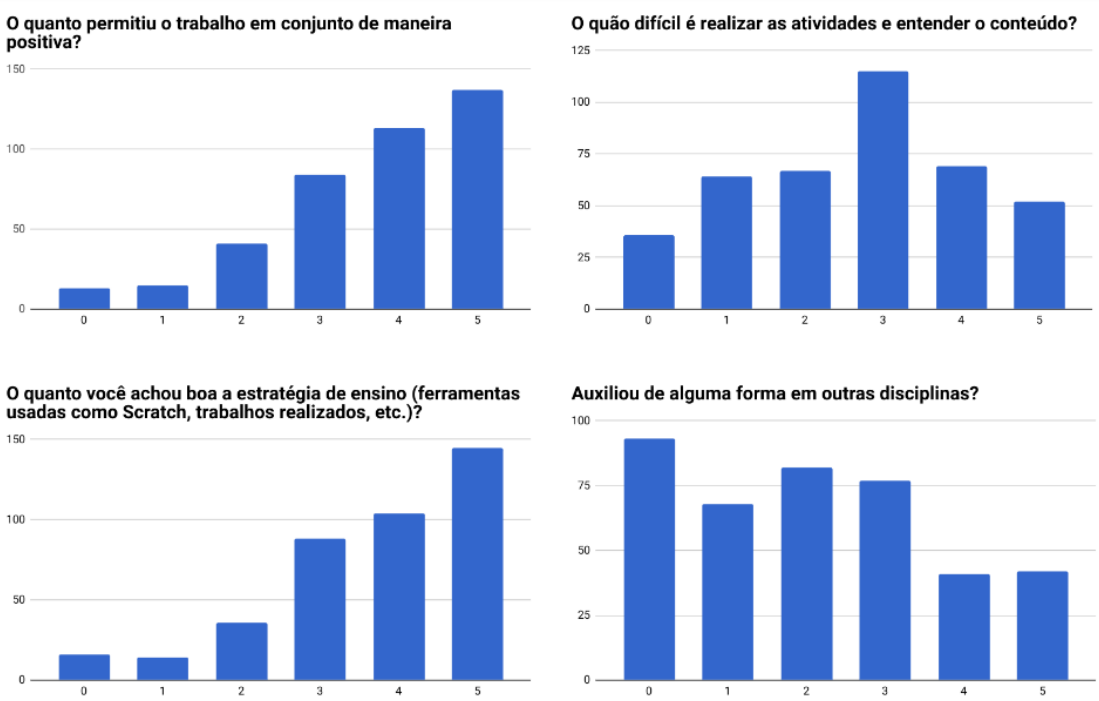


Figura 3. Respostas da Questão 5 (superior esquerdo), Questão 6 (superior direito), Questão 7 (inferior esquerdo) e Questão 8 (inferior direito)

As Questões 9 (*O quanto você se interessava pela computação antes de ter a disciplina de pensamento computacional?*) e 10 (*O quanto você se interessa pela computação agora?*) investigaram o interesse dos estudantes na área de computação antes e depois de participar da disciplina. O resultado das comparações mostrou um total de 210 estudantes informando que seu interesse em computação aumentou, enquanto 39 afirmam que o interesse diminuiu. Um total de 154 estudantes mantiveram o mesmo nível de interesse antes e depois, mas entre esses 79 escolheram a opção de interesse 5, ou seja, por ser a opção máxima não era possível explicitar um aumento do interesse.

A questão 11 abordou se o estudante se sentia capaz de criar um aplicativo ou programa que lhe podia ser útil. A resposta não (0) foi a mais frequente com 15.4% do total, mas todas as outras alternativas demonstraram uma percepção que existia alguma capacidade, mesmo que pouca (1). Sendo assim, é possível afirmar que 84.6% dos estudantes percebiam-se com alguma capacidade de criar aplicativos ou programas úteis para si mesmos.

A questão 12 investigou o interesse dos alunos em cursar o ensino superior na área da computação. A resposta mais votada foi 0 - não influenciou na decisão de cursar - com 130 votos (32.3%). Enquanto 47 estudantes (11.7%) mencionaram que a disciplina influenciou um pouco (1), 49 alunos (12.2%), manifestaram que influenciou muito (5), e os 177 estudantes restantes (43.9%) utilizaram valores intermediários (1 a 3) para definir o grau de influência da disciplina a cursar o ensino superior na área da computação.

As questões 13 e 14 tiveram como objetivo investigar em maior profundidade o resultado do Questionário 1, no qual 29 estudantes apontaram a falta de importância ou utilidade da disciplina. Com os resultados da questão 13 é possível perceber que a maioria dos alunos (117) percebe o PC como muito útil (5) em suas vidas. Apenas 34 estudantes votaram 0, o que representa 8.4% das respostas. A questão 14 apresentou 46.2% das respostas afirmando que o PC é muito útil (5) na sociedade, apenas 3.2% afirmam que não possui utilidade (0). A síntese destes dados é apresentada na Figura 4.

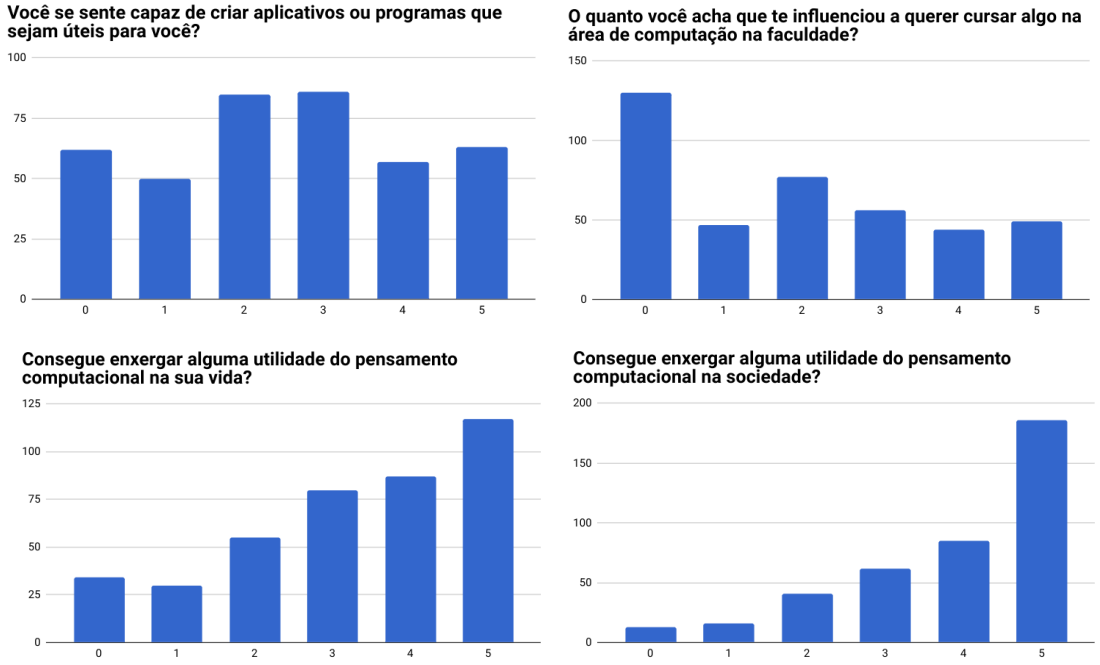


Figura 4. Respostas da Questão 11 (superior esquerdo), Questão 12 (superior direito), Questão 13 (inferior esquerdo) e Questão 14 (inferior direito)

4. Conclusões

Ao analisar os resultados é possível perceber que a maioria dos estudantes acredita que a disciplina de PC promoveu a criatividade, estimulou o raciocínio lógico, o trabalho em conjunto, e propiciou uma aula diferente das aulas tradicionais. As respostas apontam que os estudantes acharam interessante aprender a desenvolver jogos, aplicativos ou trabalhar com programação, e percebem a utilidade das habilidades desenvolvidas em sua vida pessoal e na sociedade.

A maioria dos estudantes afirma que as estratégias de ensino adotadas foram muito boas. Sobre a dificuldade atribuída à disciplina: caso a maioria das respostas apontasse para muito difícil (5) ou não difícil (0) poderíamos concluir que esta é a

característica da disciplina, mas devido à grande variação das respostas é possível que elas reflitam o grau de envolvimento de cada um com a atenção às aulas, atividades, trabalhos, afinidade com o professor, afinidade com os colegas, entre outras variáveis que afetam a dificuldade de estudantes diante de assuntos abordados na escola.

Ao responderem sobre a própria capacidade de desenvolver programas ou aplicativos úteis para si, as respostas foram bem variadas. É possível que apenas um ano cursando a disciplina não seja tempo suficiente para desenvolver a confiança dos estudantes nas suas próprias capacidades, minando a percepção do PC como ferramenta de empoderamento tecnológico.

Em um âmbito escolar com uma disciplina obrigatória de PC existem diferentes possibilidades de abordagem para o assunto. Da maneira que a disciplina foi implantada em 2017 se mostrou necessário desenvolver uma base inicial com os estudantes, abordando os fundamentos da computação e do PC, assim como apresentar diferentes ferramentas que tornassem possível a aplicação destes fundamentos. Após desenvolver a base inicial foi possível iniciar uma abordagem interdisciplinar. Para trabalhos futuros podem ser realizadas parcerias com professores de outras disciplinas para aplicar o PC na construção do conhecimento de diversas áreas, formulando o plano de ensino com base neste objetivo.

As respostas referentes ao interesse em cursar a área da computação no ensino superior mostraram que a maioria não tinha interesse em cursar a área, Isto vai de encontro com a ideia proposta por Wing (2006), na qual todos devem desenvolver o PC, mas não necessariamente se tornar um cientista da computação. Mas é possível perceber que 67.7% dos respondentes demonstraram algum interesse em cursar a área, criando a possibilidade de aumentar o número de profissionais na área de computação ou afins.

Por ser uma disciplina obrigatória, muitos estudantes que não tinham afinidade com computadores se mostraram relutantes em participar das atividades e realização dos trabalhos. Através do sistema de notas, os estudantes que não possuíam esta afinidade sentiram a necessidade de realizar as atividades e trabalhos para alcançar a média ao final do ano. A participação destes estudantes proporcionou uma experiência que não seria possível caso a disciplina não fosse obrigatória. Foi possível perceber a transformação em alguns destes estudantes, que ao iniciar as atividades perceberam do que são capazes através do PC.

O sistema de notas da escola também afetou os estudantes mais esforçados e interessados no assunto. Alguns estudantes ao perceberem que alcançaram a média necessária para ser aprovado na disciplina começaram a não participar mais das aulas, não realizar os trabalhos e atividades. Ao serem questionados sobre este comportamento afirmaram que ainda tinham interesse pela disciplina, mas que estavam cansados de tantas atividades e tantos trabalhos (em outras disciplinas), e gostariam de descansar no final do ano.

Para trabalhos futuros pretende-se: apresentar a progressão da disciplina; realizar nova coleta de dados sobre os estudantes para comparar com os dados apresentados neste artigo; e mensurar o desempenho dos alunos através de testes voltados ao PC.

4.1. Recomendações para a Implantação da Disciplina de PC

Com base na experiência da disciplina de PC no CAU recomendamos as seguintes

estratégias de implantação:

1. A realização de pequenos projetos práticos com temas escolhidos pelos estudantes apresenta bons resultados, entretanto é necessário observar a duração destes projetos evitando que sejam excessivamente longos, fazendo com que os estudantes se desmotivem e construam uma ideia negativa das ferramentas e estratégias utilizadas em aula.
2. A utilização de laboratório de informática é extremamente recomendada para uma disciplina de PC, entretanto, se a abordagem de projetos for adotada recomenda-se fortemente que existam 2 professores no laboratório, visto que a demanda de auxílio para as equipes é muito alta.
3. Recomendamos que as atividades no laboratório de informática sejam realizadas em duplas ou trios. As atividades realizadas individualmente, apesar de possibilitarem uma avaliação particularizada do desempenho de cada estudante, não possibilitam o desenvolvimento da capacidade de trabalhar em equipe e impõem a necessidade de uma grande quantidade de computadores nos laboratórios, gerando custos proibitivos para a maioria das instituições de ensino.
4. Algumas atividades podem ser realizadas em sala de aula com o telefone celular, evitando a ida até o laboratório de informática (o que consome um tempo precioso em aulas de 45 minutos). No CAU de Itajaí onde realizamos este experimento praticamente todos os estudantes possuíam telefone celular. Ainda que esta situação não represente a totalidade do Brasil sabemos que a presença dos dispositivos móveis somente tende a crescer em todas as classes sociais.
5. O tempo reduzido das aulas (45 minutos) impõe dificuldades para a realização de projetos práticos (que podem durar algumas semanas), então recomenda-se que a disciplina ocupe duas aulas por semana, e se possível duas aulas seguidas.
6. Recomendamos também que a disciplina de PC se integre com as demais disciplinas da matriz curricular, fomentando o envolvimento de professores e demais profissionais. Por exemplo, o sistema de movimentação de personagens no *Scratch* pode ser facilmente relacionado com atividades de matemática que abordam o plano cartesiano. Ao final do ano letivo os estudantes vivenciaram uma experiência na qual uma das atividades de uma gincana era cumprir um desafio de PC[9] no laboratório de informática. O desafio foi realizado pelas equipes com muita energia, e acreditamos que este é um bom exemplo de como as atividades de PC podem se relacionar com atividades físicas, que envolvem movimento. Além disso, vale mencionar também que nesta gincana a realização da atividade não valia uma nota tradicional, porém gerava uma recompensa (maior pontuação para a equipe da gincana). Em nossa experiência ficou evidente que o sistema tradicional de notas não recompensa de maneira significativa, enquanto sistemas alternativos, como a gincana, parecem gerar bons resultados.

[9] http://gincana-cau.appspot.com/

References

Aono, A. H., Rody, H. V. S., Musa, D. L., Pereira, V. A. e Almeida, J. (2017) "A Utilização do Scratch como Ferramenta no Ensino de Pensamento Computacional para Crianças" In: 25º WEI - Workshop sobre Educação em Computação

DeLyser, L. A., Mascio, B. e Finkel, K. (2016) "Introducing Student Assessments with Evidence of Validity for NYC's CS4All" In: Proceedings of the 11th Workshop in Primary and Secondary Computing Education

Dettori, L., Greenberg, R.I., McGee, S., Reed, D. (2016) "The impact of the exploring computer science instructional model in Chicago Public Schools" In: Computing in Science & Engineering

Heintz, F., Mannila, L. e Färnqvist, T. (2016) "A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K–12 Education" In: Frontiers in Education Conference. Los Alamitos.

Reis, F. de M., Oliveira, F. C. S., Martins, D. J.da S. e Moreira, P. da R. (2017) "Pensamento Computacional: Uma Proposta de Ensino com Estratégias Diversificadas para Crianças do Ensino Fundamental" In: Anais do Workshop de Informática na Escola. Vol. 23.

Sabbagh, S. A., Gedawy, H., Alshikhabobakr, H. e Razak, S. (2017) "Computing Curriculum in Middle Schools: An Experience Report" In: ACM Conference on Innovation and Technology in Computer Science Education

Santos, W. O. dos (2017) "Mulheres na Computação: Uma Análise da Participação Feminina nos Cursos de Licenciatura em Computação" In: Workshops do VI Congresso Brasileiro de Informática na Educação, Recife.

Souza, I. M. L., Rodrigues, R. S. e Andrade, W. L. (2016) "Introdução do Pensamento Computacional na Formação Docente para Ensino de Robótica Educacional" In: Workshops do Congresso Brasileiro de Informática na Educação, Uberlândia.

Wing, J. M. (2006) "Computational Thinking" In: Communications of the ACM, v.49, n.3, p.33-35.

BrC: Proposta de uma Biblioteca em Português para Ensino de Programação em Linguagem C

Luciana Rita Guedes[1], Aleksander Sade Paterno[2]

[1]Departamento de Ciência da Computação – [2]Departamento de Engenharia Elétrica - Universidade do Estado de Santa Catarina (UDESC) - Centro de Ciências Tecnológicas
Rua Prof. Paulo Malschitzki, 200 – 89.219-710 – Joinville – SC – Brasil

[luciana.guedes,aleksander.paterno]@udesc.br

Abstract. *This paper presents the proposal of a C language compatible library that replaces reserved words of this programming language with equivalent words in Portuguese and can be applied in different integrated development environments (IDE). In addition, the basic input and output functions have their syntax simplified. The aim is to facilitate the learning of native Portuguese language programming beginners and to allow a smooth transition to the use of the C language. The library is easy to use and publicly available and no additional software is required for its application, besides the environment already used. The library was successfully used in beginner classes with undergraduate students from different courses for six semesters.*

Resumo. *Este artigo apresenta a proposta de uma biblioteca compatível com a linguagem C que substitui palavras reservadas desta linguagem por vocábulos equivalentes em português e pode ser aplicada em diferentes ambientes de desenvolvimento. Adicionalmente, as funções básicas de entrada e saída têm a sua sintaxe simplificada. O objetivo é facilitar a aprendizagem de iniciantes em programação nativos da língua portuguesa e permitir uma transição suave para o uso da linguagem C. A biblioteca é de fácil utilização e está publicamente disponível, não sendo necessária a instalação de nenhum software adicional para sua aplicação, além do próprio ambiente já utilizado. A biblioteca foi utilizada com êxito em turmas iniciantes com estudantes universitários de diferentes cursos durante seis semestres.*

1. Introdução

Estudantes iniciantes de programação costumam apresentar dificuldades no entendimento da lógica do computador, independentemente da linguagem de programação escolhida. O aprendizado de programação de computadores é considerado uma tarefa não simples e não trivial [Vahldick et al. 2015, Scaico et al. 2012]. Diversas propostas para melhorar o ensino de programação têm sido elaboradas e testadas nos últimos anos. Algumas enfatizam o uso de ferramentas [Scratch, Codecademy, Code.org 2018][Resnick et al. 2009], outras tem foco na metodologia [de Jesus, Raabe 2009][Denton et al. 2005]. Do ponto de vista da linguagem de programação escolhida, é fato que algumas apresentam estruturas sintáticas mais complexas que outras. A linguagem C (ANSI) possui uma sintaxe mais complexa, com mais elementos e maior rigor estrutural que outras linguagens conhecidas como Pascal e Python, por exemplo.

No entanto, a linguagem C costuma ser a escolha de muitos cursos superiores como linguagem inicial, por suas características de aplicações e robustez.

Muitos estudos têm sido realizados, em todo o mundo, especialmente nos últimos anos com objetivo de aprimorar o ensino de algoritmos e programação de computadores, tanto no aspecto metodológico como no desenvolvimento e utilização de ferramentas [Valhdick 2014, Zanchet et al. 2017, Laporte; Zaman 2017].

No cenário brasileiro, esta também é uma temática recorrente nos principais eventos e periódicos da área de Computação. No caso do Brasil, outra questão considerada é que, embora fosse altamente desejável que os ingressantes de cursos superiores já tivessem um domínio da língua inglesa, ao menos em termos de comunicação básica, a realidade encontrada nas universidades não está de acordo com este requisito [Anido 2015, Noschang 2014]. Um estudo mais recente sobre o índice de proficiência em inglês [EPI 2017] indica que o Brasil teve um pequeno descréscimo entre 2016 e 2017, caindo da 40ª para a 41ª posição, entre 80 países pesquisados. O mesmo estudo indica, ainda, que estudantes universitários de Engenharia e Ciências (exatas) são os que têm os piores índices de proficiência. Em contraste com estes dados, o ensino de programação de computadores costuma ser aplicado justamente nestes cursos.

Para amenizar as dificuldades com a língua inglesa, alguns professores optam pelo uso de pseucódigo como Portugol ou outras estratégias como desenvolvimento de fluxogramas ou outros diagramas similares. Algumas destas abordagens exigem que o estudante crie seus algoritmos em papel sem visualizar os resultados de execução de seus códigos. Outros casos usam ferramentas cuja sintaxe difere da linguagem de programação que será utilizada, duplicando o esforço de aprendizagem do estudante [Noschang 2014].

Neste artigo será apresentada uma biblioteca que permite a escrita do código-fonte usando a mesma sintaxe da linguagem de programação C e o mesmo ambiente de desenvolvimento já escolhido pelo professor, porém usando palavras em língua portuguesa. A seção 2 apresenta trabalhos relacionados e a seção 3 traz o detalhamento sobre o desenvolvimento da biblioteca. Por fim, os resultados alcançados com o experimento são apresentados na seção 4.

2. Trabalhos correlatos

O ensino de programação para iniciantes tem sido amplamente discutido no meio científico. Alguns trabalhos de revisão de literatura têm comprovado a quantidade de publicações desta área. Vihavainen et al. (2014) apresentam uma revisão sistemática da literatura (RSL) feita nas bases de dados da ACM e da IEEE, com foco nos índices de reprovação em cursos de programação para iniciantes. De modo a fazer uma análise comparativa com estes resultados, Ramos et al. (2015) também realizaram uma RSL, porém com foco regional, considerando apenas publicações no Brasil, confrontando os dados com aquele primeiro estudo. O primeiro trabalho resultou em mais de 1000 artigos e, no segundo caso, a busca retornou 394 artigos sobre o tema.

Com enfoque mais voltado ao ensino de programação para estudantes da educação básica, Silva (2017) obteve um total de 914 artigos em sua revisão bibliográfica, também restrita a eventos e periódicos brasileiros, considerando apenas a trabalhos publicados num período de cinco anos. Com relação a iniciativas de uso de

ferramentas que auxiliam no ensino, o trabalho de revisão da literatura de Aureliano e Tedesco (2012), com foco em publicações feitas no Brasil, concluiu que cerca de 65% dos estudos estão voltados a estas ferramentas. Entre as ferramentas de apoio ao ensino de programação para iniciantes no Brasil, pode-se destacar o trabalho de Leite et al. (2013) que apresenta um estudo de caso do VisuAlg. A ferramenta permite a escrita de pseudocódigos em português usando a sintaxe da linguagem Pascal. Petry e Rosatelli (2006) apresentam a ferramenta AlgoLC, que também permite a construção de algoritmos em língua portuguesa, com sintaxe que também assemelha-se à linguagem Pascal. Blatt et al. (2017) realizaram um mapeamento sistemático da literatura em busca de metodologias e ferramentas utilizadas no ensino algoritmos e programação e destacaram o uso do Scratch em mais de 40% dos artigos pesquisados. O Scratch [Maloney et al. 2008] é uma ferramenta visual que utiliza uma técnica denominada programação por blocos. Tem também uma abordagem lúdica que usa personagens e desenvolvimento de jogos como recursos de ensino e aprendizagem e está disponível em português. Noschang et al. (2014) apresentaram o Portugol Studio, um ambiente integrado de desenvolvimento que utiliza comandos escritos em língua portuguesa para o desenvolvimento de algoritmos computacionais. O trabalho de Anido (2015) apresenta o ambiente SACI, desenvolvido para o ensino de algoritmos voltado aos competidores da Olimpíada Brasileira de Informática. O SACI tem toda interface em língua portuguesa, mas mantém o código-fonte na linguagem de programação escolhida (JavaScript ou Python).

Muitos outros estudos apresentam propostas de ferramentas, muitas delas em língua portuguesa, demonstrando um esforço da comunidade científica na busca de soluções que atendam as especificidades dos estudantes nativos desta língua.

3. Metodologia de criação da biblioteca BrC

Para criação da biblioteca BrC foi feito, inicialmente, o levantamento das palavras reservadas da linguagem C que, segundo o padrão ANSI [Kernighan e Ritchie 1987] são um total de 32 palavras, conforme mostradas no Quadro 1.

Quadro 1. Palavras reservadas da linguagem C (padrão ANSI). Fonte: [Kernighan e Ritchie 1987]

1. auto	9. double	17. int	25. struct
2. break	10. else	18. long	26. switch
3. case	11. enum	19. register	27. typedef
4. char	12. extern	20. return	28. union
5. const	13. float	21. short	29. unsigned
6. continue	14. for	22. signed	30. void
7. default	15. goto	23. sizeof	31. volatile
8. do	16. if	24. static	32. while

Em seguida, foi feita uma tradução de cada uma destas palavras para equivalentes no idioma português. Por questão de organização, as 32 palavras reservadas foram divididas em quatro blocos distintos: 1) tipos de variáveis; 2) comandos de sistema; 3) comandos de seleção e 4) comandos de iteração. Ressalta-se que esta divisão não era obrigatória, sendo apenas uma decisão de implementação, sem nenhum impacto no resultado final. Após isso, foi criado um arquivo de extensão .h (header ou biblioteca) que, com o uso da diretiva de compilação *#define* permitiu "redefinir" as palavras para que fossem interpretadas pelo compilador na sua versão em português no lugar do original em inglês. Esta biblioteca foi nomeada BrC, como referência à linguagem C escrita na língua do Brasil, o idioma português (que poderia ser chamada de *Brazilian C, ou C "brasileiro"*).

Além das palavras reservadas da linguagem C, outros elementos foram incorporados à biblioteca. Inicialmente, no cabeçalho desta biblioteca, foram adicionadas diretivas do tipo *#include* para incluir as bibliotecas mais comumente utilizadas que são *stdio.h* e *stdlib.h*. Esta alteração foi feita para simplificar o código do estudante iniciante de modo que ele só tenha que incluir a própria biblioteca brc.h em seu programa, sem preocupar-se com outras bibliotecas básicas.

Outra alteração implementada foi a modificação das funções básicas de entrada e saída (leitura e escrita de dados). Aqui cabem algumas considerações, pois a observação do processo de aprendizagem de linguagens de programação, especialmente das funções de entrada e saída, tem demonstrado que os estudantes costumam ter dificuldades de compreensão sobre o significado destas funções. A função de entrada de dados via teclado representa, ao mesmo tempo, uma pausa na execução do código para aguardar a interação do usuário e também um processo de armazenamento na memória, que guarda o dado digitado. Não bastasse esta complexidade inerente à função, a sintaxe de **scanf**, na linguagem C, possui dois parâmetros, um dos quais com uso de ponteiros, como em scanf("%d",&x). A biblioteca do BrC possui palavras para representar a entrada e saída de dados relativas a scanf e printf que são, respectivamente, leiaC e escrevaC. Neste caso, a sintaxe é rigorosamente idêntica às funções originais em C. No entanto, com o objetivo de simplificar o entendimento da entrada e saída de dados, optou-se por tomar emprestados os comandos da linguagem C++, **cin** e **cout**. Isto tornou possível criar as funções **leia** e **escreva** que possuem sintaxe simplificada e facilitam o entendimento. Isto evita que o iniciante em programação tenha que preocupar-se com os detalhes da sintaxe quando ainda está compreendendo o processo envolvido na função, seja ela de entrada ou saída de dados.

Finalmente, foram incorporadas mais duas funções à lista já existente: a função *main* foi redefinida com a palavra "principal" e a função *system* foi redefinida com a palavra "sistema". Mais uma vez, foi apenas uma decisão de implementação, apenas para evitar o uso de palavras em inglês no código fonte do estudante, em consonância com a ideia do BrC.

Após estes procedimentos, a biblioteca BrC ficou conforme aparece no Quadro 2, que apresenta o arquivo brc.h integralmente, da forma como foi criado e está disponível publicamente no endereço <https://goo.gl/jB8hTh>.

Para usar a biblioteca brc.h, é necessário apenas copiá-la para a pasta onde encontram-se as outras bibliotecas já utilizadas pelo compilador. Por exemplo, no caso do uso do **gcc**, há uma pasta denominada **include** onde já existem as demais bibliotecas-

padrão usadas pelo compilador. Basta encontrar esta pasta e copiar o arquivo **brc.h** para dentro dela. No sistema operacional Windows, a biblioteca foi usada tanto no ambiente de desenvolvimento Dev-C++ como no CodeBlocks, pois ambos utilizam o gcc como compilador. No sistema operacional Linux, a biblioteca foi usada com o ambiente Geany.

Após a inserção da biblioteca na pasta correta, pode-se criar os programas usando-se o ambiente de desenvolvimento escolhido e substituindo as palavras da linguagem C pelas equivalentes na biblioteca brc.h. O Quadro 3 apresenta o exemplo de um programa em C e seu equivalente em BrC.

Quadro 2. Conteúdo da biblioteca brc.h

```
//Cabeçalhos
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <iostream>
using namespace std;

//Tipos de variáveis
#define local auto
#define caractere char
#define constante const
#define dobro double
#define enumerador enum
#define externo extern
#define real float
#define inteiro int
#define longo long
#define registrador register
#define curto short
#define sinalizado signed
#define tamanhode sizeof
#define estatico static
#define estrutura struct
#define tipodef typedef
#define naosinalizado unsigned
#define uniao union
#define vazio void
#define volatil volatile
```

```
//Comandos de sistema
#define retorne return
#define quebre break
#define continua continue
#define padrao default

//Comandos de seleção
#define se if
#define senao else
#define escolha switch
#define caso case

//Comandos de iteração
#define faca do
#define enquanto while
#define para for
#define vapara goto

//Funções de entrada e saída
#define leiaC scanf
#define escrevaC printf
#define escreva(VALOR) cout << VALOR
#define leia(VALOR) cin >> VALOR

//Outras funções
#define principal main
#define sistema system
```

É importante observar que toda sintaxe da linguagem permanece inalterada (uso de chaves, ponto-e-vírgula, parêntesis, etc.), bem como os operadores relacionais (==,>,<,>=,<=,etc.). Os operadores lógicos (&&,||, !) também foram mantidos sem alteração. O objetivo é familiarizar o estudante com a sintaxe da linguagem, trocando apenas as palavras escritas em inglês.

Deste modo, a biblioteca criada representa uma tentativa de mudança de modalidade simbólica para compreensão de comandos e seu suporte sensório às modalidades de comunicação que os alunos já conhecem, de acordo com a experiência dos docentes.

Quadro 3. Exemplo de programa em BrC e seu equivalente em C.

```
// programa escrito em BrC
#include <brc.h>

principal(){

   real a,b;
   inteiro c;

   faca{
      escreva("Digite um valor:");
      leia(a);
      escreva("Digite outro valor:");
      leia(b);

      se(a==b){
         escreva("Valores iguais.\n");
      }
      senao{
         escreva("Valores diferentes.\n");
      }

      escreva("Digite 1 para repetir, ");
      escreva("outro valor para sair:");
      leia(c);
   } enquanto(c==1);
   retorne 0;
}
```

```
// programa equivalente escrito em C
#include <stdio.h>

main(){

   float a,b;
   int c;

   do{
      printf("Digite um valor:");
      scanf("%f",&a);
      printf("Digite outro valor:");
      scanf("%f",&b);

      if(a==b){
         printf("Valores iguais.\n");
      }
      else{
         printf("Valores diferentes.\n");
      }

      printf("Digite 1 para repetir, ");
      printf("outro valor para sair:");
      scanf("%i",&c);
   } while(c==1);
   return 0;
}
```

Outra observação diz respeito ao uso das funções **printf** e **scanf**, que aparecem na versão em C do Quadro 3. Conforme já mencionado, para simplificar o entendimento do estudante, na biblioteca brc.h os comandos **escreva** e **leia** equivalem, respectivamente, a **cout** e **cin**, emprestados da linguagem C++. No entanto, numa aula de linguagem C, os comandos usados para entrada e saída de dados costumam ser aqueles mencionados inicialmente (scanf e printf). Por este motivo, eles foram descritos como equivalentes aos comandos **escreva** e **leia** no exemplo apresentado.

4. Experimento e Resultados

O uso da biblioteca foi aplicado em diferentes turmas durante seis semestres, entre 2014 e 2017. As turmas eram mistas, compostas de estudantes de diferentes cursos de Engenharia e de Licenciatura. Os estudantes de engenharia eram de cursos como Civil, Elétrica e de Produção e o licenciandos eram de Física ou Matemática. Os estudantes eram, em sua grande maioria, iniciantes, ou seja, não tinham experiência alguma com linguagem de programação ou algoritmos computacionais. Isto foi verificado em entrevista realizada no primeiro dia de aula, como prática de apresentação conduzida e registrada pelo docente em todas as turmas. A disciplina de Algoritmos e Linguagem de Programação é ministrada com o objetivo do aprendizado da linguagem C para aplicação em resoluções de problemas em diversas áreas do conhecimento. A opção pela linguagem C é uma decisão dos colegiados dos cursos em acordo com o departamento de Computação, que oferece a disciplina na Universidade. As turmas tinham, em média, 20 estudantes e as aulas eram dadas em laboratório de informática com um equipamento para cada estudante. A disciplina possui uma carga horária total de 72 horas/aula e é distribuída em 4 aulas semanais, sendo 2 aulas teóricas e 2 práticas.

O uso da biblioteca BrC foi realizado a partir da segunda semana de aula, quando são iniciados os primeiros algoritmos computacionais, após as aulas introdutórias de apresentação da disciplina e funcionamento básico do hardware. O

grande diferencial ocorre nestas primeiras aulas, pois os estudantes já podem observar seus programas em execução no ambiente de desenvolvimento escolhido pelo professor. Eles foram incentivados a copiarem a biblioteca para seus computadores pessoais para executarem as tarefas extra-classe que servem de fixação dos conteúdos.

Embora existam várias outras iniciativas de uso de ferramentas de pseudocódigo ou similares [Petry e Rosatelli 2006, Leite et al. 2013, Noschang et al. 2014] que permitam a observação do programa em execução, o uso do BrC pemite que o estudante escreva seu código utilizando a sintaxe da linguagem desde os primeiros algoritmos (uso de ponto-e-vírgula, chaves, parêntesis, etc.) e já inicie no ambiente de desenvolvimento onde ele usará a linguagem C posteriormente.

Nas turmas onde o BrC foi aplicado, o uso prosseguiu até aproximadamente metade da carga horária da disciplina, quando os estudantes já haviam passado pelas estruturas básicas de programação como sequência, seleção e iteração. Após esta fase onde os comandos básicos e, principalmente, a lógica de programação básica já estava internalizada, foi feita a transição para a linguagem C. Esta transição também caracteriza uma vantagem do BrC pois trata-se basicamente da tradução das palavras para a linguagem C. A única diferença ocorre na apresentação dos comandos *scanf* e *printf*, que têm sua sintaxe apresentada pela primeira vez, já que a biblioteca utiliza uma simplificação dos comandos de entrada e saída. Vale ressaltar que isto ocorre quando as dificuldades iniciais já foram ultrapassadas é há uma vivência prática já adquirida do uso destes comandos com a biblioteca do BrC.

Não foram realizadas métricas de desempenho por nota ou medidas de satisfação dos estudantes nas turmas onde a biblioteca foi aplicada. No entanto, os resultados observados foram: 1) possibilidade de execução do código desde as aulas iniciais: em contraponto com os métodos que usam algoritmos feitos com base exclusivamente em lápis e papel, a observação da execução do programa desde as primeiras aulas traz ao estudante uma experiência de aprendizagem mais concreta; 2) o uso do mesmo ambiente de desenvolvimento durante toda a disciplina: em todas as turmas aplicadas, o ambiente de desenvolvimento (IDE) escolhido para o BrC foi usado igualmente após a mudança para o uso da linguagem C, em contraste com as práticas que utilizam um ambiente na fase de introdução aos algoritmos e outro ainda na fase em que a linguagem C passa a ser integralmente utilizada pelos algunos e 3) a transição suave para a linguagem de programação C: a atividade docente de apresentação dos comandos da linguagem C anteriormente necessitava de duas aulas de 50 minutos para ser concluída e, após a prática de uso do BrC, passou a tomar apenas um quarto (1/4) deste tempo, ou seja, aproximadamente 25 a 30 minutos, viabilizando mais tempo em aula para os estudantes fazerem exercícios e tirarem eventuais dúvidas diretamente com o professor.

4. Discussão e Conclusão

Este trabalho apresentou o desenvolvimento e aplicação de uma biblioteca compatível com a linguagem C que permite substituir palavras reservadas da linguagem para equivalentes em língua portuguesa. A biblioteca foi construída de modo bem simples e pode ser usada com os principais ambientes de desenvolvimento já existentes para a linguagem C.

A aplicação nas turmas apresentou características observadas como positivas sob o ponto de vista do professor, quando permitiu diminuir o tempo de aula expositiva em

alguns momentos e, sob o ponto de vista dos estudantes, trouxe algumas características que podem facilitar o processo de aprendizagem e de familiarização com a linguagem de programação C.

O uso do BrC é uma estratégia inspirada na necessidade de o discente ter alguma forma de suporte menos abstrato para auxiliá-lo no desenvolvimento da capacidade de abstração requerida pela linguagem C ordinária. Dessa forma, o BrC mantém o conteúdo da disciplina num nível inicial mais acessível ao aluno até que ocorra a transição quando não se utilizará mais a biblioteca de tradução.

Como os experimentos realizados não fizeram um estudo comparativo de desempenho dos estudantes com o uso da biblioteca em relação a outras metodologias tradicionais, sugere-se que trabalhos futuros possam buscar este direcionamento de pesquisa.

Por fim, a proposta aqui apresentada não pretende ser indicada como solução final para os inúmeros problemas do ensino de programação da linguagem C, mas sim indicar uma alternativa simples que pode ser facilmente replicada em contextos similares.

Agradecimentos

Registram-se agradecimentos ao estudante Carlos Henrique da Costa, monitor da disciplina de Algoritmos, e ao Prof. Dr. Rui Jorge Tramontin Junior, professor do Departamento de Ciência da Computação, pela contribuição dada na realização deste estudo. Este trabalho está vinculado ao projeto de pesquisa "Novas abordagens para o ensino de lógica de programação: estudo, análise e uso das estratégicas didático-pedagógicas e dos recursos tecnológicos" do Grupo de Pesquisa em Informática na Educação (GPIE) da UDESC.

Referências

Anido, R. (2015). “SACI – ainda outro ambiente para o ensino de programação”. In: *Anais do Workshop sobre Educação em Computação (WEI)* .

Aureliano, V. C. O., & Tedesco, P. C. D. A. R. (2012). “Ensino-aprendizagem de Programação para Iniciantes: uma Revisão Sistemática da Literatura focada no SBIE e WIE”. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)* (Vol. 23, No. 1).

Blatt, L., Becker, V., & Ferreira, A. (2017). "Mapeamento Sistemático sobre Metodologias e Ferramentas de apoio para o Ensino de Programação". In Anais do Workshop de Informática na Escola (Vol. 23, No. 1, p. 815).

CODECADEMY (2018). Codecademy: aprenda a programar. Website. Disponível em: <http://www.codecademy.com/>. Acesso em: 16/03/2018.

CODE.ORG (2018). Code.org: comece a aprender. Website. Disponível em: <http://www.code.org/>. Acesso em: 16/03/2018.

Denton, L. F., McKinney, D., & Doran, M. V. (2005, October). A melding of educational strategies to enhance the introductory programming course. In Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference (pp. F4G-7). IEEE.

EPI English Proficiency Index (2017). "Índice de Proficiência em Inglês da EF para escolas". Relatório anual da *Education First (EF) English Proficiency Index.* Disponível em <https://www.ef.com/__/~/media/centralefcom/epi/downloads/epi-s/2017/ef-epi-s-2017-portuguese.pdf>. Acesso em 02/04/2018.

Hinterholz, O. (2009). "Tepequém: uma nova Ferramenta para o Ensino de Algoritmos nos Cursos Superiores em Computação". In *XVII-Anais do Workshop sobre Educação em Informática* (Vol. 20, p. 21).

de Jesus, E. A., & Raabe, A. L. A. (2009, November). Interpretações da TAXONOMIA de Bloom no contexto da Programaçao Introdutória. In Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE) (Vol. 1, No. 1).

Kernighan, B. W., & Ritchie, D. M. (1987). "*C a Linguagem de Programação"*. Campus.

Laporte, L., & Zaman, B. (2017). "A comparative analysis of programming games, looking through the lens of an instructional design model and a game attributes taxonomy". *Entertainment Computing*.

Leite, V. M., Senefonte, H. C., & Seabra, R. D (2013). "VisuAlg: Estudo de Caso e Análise de Compilador destinado ao ensino de Programação". In *Nuevas Ideas en Informática Educativa* TISE 2013. p. 637-640.

Maloney, J., Peppler, K., Kafai, B., Y., Resnick, M., , and Rusk, N. (2008). "Programming by choice: urban youth learning programming with Scratch". ACM SIGCSE Bulletin, 40(1):367–371.

Noschang, L. F., Pelz, F., & Raabe, A. (2014). "Portugol Studio: Uma IDE para iniciantes em programação". *Anais do CSBC/WEI*, 535-545.

Petry, P. G., & Rosatelli, M. C. (2006). Ensino e Aprendizagem de Algoritmos com o AlgoLC. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)* (Vol. 1, No. 1, pp. 408-417).

Ramos, V., Wazlawick, R., Galimberti, M., Freitas, M., & Mariani, A. C. (2015). "A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional: Revisão sistemática da literatura em eventos brasileiros". In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)* (Vol. 26, No. 1, p. 318).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.

Scaico, P., Lopes, D., Silva, M. D. A., Silva, J. D., Neto, S. V. M., & Falcão, E. D. (2012). "Implementação de um Jogo Sério para o Ensino de Programação para Alunos do Ensino Médio Baseado em m-learning". In Anais do XX Workshop sobre Educação em Computação. Curitiba: PR.

SCRATCH (2018). Scratch: crie estórias, jogos e animações. Website. Disponível em: <http://www.scratch.mit.edu/>. Acesso em: 16/03/2018.

Silva, J. C. D. (2017). Ensino de Programação para alunos do Ensino Básico: Um levantamento das pesquisas realizadas no Brasil. Trabalho de Conclusão de Curso. Universidade Federal da Paraíba.

Vahldick, A., Mendes, A. J., & Marcelino, M. J. (2014). "A review of games designed to improve introductory computer programming competencies". In *Frontiers in Education Conference* (FIE), 2014 IEEE (pp. 1-7). IEEE.

Vahldick, A., Mendes, A. J., Marcelino, M. J., Hogenn, M., & Schoeffel, P. (2015). Testando a diversão em um jogo sério para o aprendizado introdutório de programação. 23º WEI-Workhop Sobre Educação Em Computação.

Vihavainen, A., Airaksinen, J., & Watson, C. (2014). "A systematic review of approaches for teaching introductory programming and their influence on success". In *Proceedings of the tenth annual conference on International computing education research* (pp. 19-26). ACM.

Zanchett, G. A., Vahldick, A., & Raabe, A. (2017). "Games for Programming as an Approach for First Programming Experiences". *International Journal on Computational Thinking* (IJCThink), 1(1), 39.

Cosmo: Um ambiente virtual de aprendizado com foco no Ensino de Algoritmos

Dilson José Lins Rabêlo Júnior[1], **Carlos de Salles Soares Neto**[1], **Antonio Carlos Raposo**[1], **Luis Alves dos Santos Neto**[1]

[1] Av. dos Portugueses, 1966 - Vila Bacanga, São Luís - MA, 65065-545

{dilsonjlrjr,csallesneto}@gmail.com

{antoniocarlosraposo93,alvesgamadossantos}@gmail.com

Resumo. *Um dos principais problemas enfrentados pelos cursos de graduação na área de informática é a alta taxa de desistência prematura. Deve-se então considerar o papel crítico da disciplina de algoritmos, já que tal disciplina é o primeiro contato que o discente possui com programação de computadores. Tornar o ensino de programação mais prático é um desafio e o processo de ensino-aprendizagem tem contado com o auxílio da tecnologia para construções de ferramentas. O Cosmo é um ambiente virtual de aprendizado, multitarefa e extensível a plugins, focado em atividades voltada ao estudo da disciplina de algoritmos. A proposta deste trabalho se insere no contexto do projeto do Cosmo, apresentando sua arquitetura e funcionalidades, como elas foram concebidas e estruturadas.*

1. Introdução

Nos cursos de graduação na área de informática, a disciplina de algoritmos tem papel fundamental. Isso porque, é através desta que a maioria dos alunos tem o primeiro contato com a programação de computadores. A disciplina de algoritmos ensina a aplicação do raciocínio lógico para resolução de problemas e sua descrição na forma de algoritmos. Apesar da sua importância, essa disciplina apresenta altos índices de reprovação e desistência. Cursos como Sistema de Informação e Ciência da Computação apresentam as maiores taxas de evasão dentre os cursos de educação superior no Brasil [Carvalho and Tafner 2006]. Segundo [Palmeira and Santos 2014], cursos de Ciência da Computação possuem uma taxa de ingressantes e concluintes de 14,3%. [dos Santos and Costa 2006] afirma que um dos motivos para desistência nos cursos é a dificuldade em aprender os conceitos básicos de programação.

Alguns fatores que causam a desmotivação no aprendizado de algoritmos é a dificuldade dos alunos em desenvolver raciocínio lógico, em conjunto da dificuldade em compreender as abstrações envolvidas no processo de programação e também a abordagem pedagógica [Rapkiewicz et al. 2007]. Diante disso, faz-se necessário buscar meios que tornem o ensino de algoritmos mais prático e atrativo para esse público.

A utilização de ambientes virtuais de aprendizagem no suporte ao ensino vem ganhando espaço no contexto educacional como ferramenta de apoio, devido a sua eficiência junto aos diferentes perfis de estudantes existentes. No ambiente virtual de aprendizagem (AVA), o processo de ensino-aprendizado pode ser mais dinâmico e personalizado.

Assim, em um mesmo AVA, podem ser oferecidos aos alunos diferentes mídias e atividades para auxiliá-los na assimilação do conteúdo. Essa possibilidade do aluno aprender sob diferentes formas é bem vista levando em consideração a existência de diferentes estilos de aprendizagem. Potencializando a oferta de um ensino cada vez mais eficaz, dado que o conteúdo da disciplina passa a ser adaptado ao estilo de aprendizado de cada discente.

Diante disso, uma forma de auxiliar o aprendizado dos alunos, é construir um ambiente que o permita conseguir estudar de acordo com suas características e preferências. Assim, é apresentado o Cosmo um ambiente virtual de aprendizado, focado em atividades voltada ao estudo da disciplina de algoritmos. O ambiente é uma plataforma multitarefa extensível a *plugins* que disponibiliza ao aluno uma experiência de aprendizado diversificada.

O artigo está organizado como segue. A Seção 2 descreve os trabalhos relacionados. A Seção 3 descreve o ambiente Cosmo. A Seção 4 descreve os testes de uso e resultados obtidos por uso do ambiente. A Seção 5 descreve os trabalhos futuros. Por fim, na Seção 6 descreve as considerações finais.

2. Trabalhos Relacionados

Várias pesquisadores estão dedicando esforços para desenvolver métodos e softwares educacionais para apoiar o aprendizado do ensino de algoritmos. Durante a pesquisa pode-se perceber que existe uma forte tendência no uso de jogos para como apoio ao ensino e aprendizado de algoritmos.

Professores e alunos do CSET (Ciência da Computação, Engenharia e Tecnologia) ressaltaram em seu trabalho [de Oliveira Brandão et al. 2012], a mesma problemática destacada na introdução deste artigo, sobre o processo de ensino e aprendizagem de algoritmos a alunos iniciantes nos cursos de computação. Como solução, eles apresentam o iVProg, uma aplicação derivada de um outro projeto chamado Alice [Conway and Pausch 1997], este faz uso da programação visual, no qual possibilita o aluno criar funções e métodos para construção de um programa com um simples arrastar do mouse. A ferramenta já traz consigo, elementos pré-definidos como comparação de variáveis, funções matemáticas e elementos básicos da programação principal como *if-then-else e loops*.

O trabalho desenvolvido por [Topalli and Cagiltay 2018], apresenta o uso da ferramenta *Scratch* sendo utilizada nas disciplinas de programação no curso de engenharia. Os estudantes foram acompanhados durante todo o curso. Nesse tempo, diversos projetos eram entregues aos alunos e o seu desenvolvimento deveria ser feito com auxílio do *Scratch*. Todo o progresso dos estudantes foi devidamente registrado e ao final da pesquisa os dados tabulados mostraram que o uso da ferramenta, apoiadas a outras metodologias, podem ajudar potencialmente no andamento do curso.

No artigo [Tillmann et al. 2014] apresenta o Code Hunt, um jogo educacional estilo caça código. O jogador, ou caçador de código, busca fragmentos que faltam para completar, ou desenvolver uma solução mais elegante para um código desafio. O jogador tem possibilidade de escrever os códigos em C# ou Java, o que exige dele um conhecimento básico ou mediano em uma das linguagens. O fluxo de execução do jogo, se inicia

quando o jogador se inscreve no site, logo em seguida é levado para o processo de seleção da linguagem e por último uma lista de fases são apresentadas, sendo inicialmente disponível somente a fase de treinamento. Cada fase aborda tópicos específicos sobre lógica de programação, como exemplo: estruturas de repetição, vetores, ordenação e etc.

Um jogo sério amplamente utilizado no ensino introdutório de programação de computadores pode ser visto no artigo do [Kazimoglu et al. 2012]. O Lightbot é um jogo com temática lúdica, ensina lógica de programação através comandos estruturados em blocos. Foi concebido com o intuito de capacitar alunos iniciantes e a lógica de programação. O seu ambiente aborda tópicos introdutórios de lógica, tais como a construção de algoritmo, depuração e simulação. O objetivo do jogo é fazer com o robozinho acendas as luzes durante um percurso e escape de obstáculos que possa encontrar no caminho. Sua mecânica está organizada em organizar os blocos de movimentos para que o robozinho possa se movimentar e acender as luzes durante o caminho.

A proposta do Cosmo se diferencia das demais, pois espera-se integrar os diversos tipos de atividades em um único ambiente. O modelo de agregar essa diversidade de atividades se objetiva em melhorar a experiência do usuário no uso da aplicação. Com modelo extensível a *plugins* é admissível o uso de qualquer modelo de atividade dentro do ambiente.

3. O Cosmo

Cosmo é um ambiente virtual de aprendizado, multitarefa e extensível a plugins, focado em atividades voltada ao estudo da disciplina de algoritmos.

Figura 1. Apresentação do Cosmo

Assim como listado na Figura 1, o Cosmo pode ser apresentado descrevendo os cinco itens na figura. Cada um dos itens será detalhado nas sub-seções a seguir.

3.1. Área de Dashboard

Ainda em sua versão *beta*, o Cosmo tem sua interface subdividida em duas áreas. A primeira corresponde a chamada de área ***Dashboard***, nela o aluno poderá observar todas as atividades disponíveis para resolução. A segunda corresponde ao **Histórico de atividades**, nela são listadas todas as atividades já respondidas pelo aluno. A Figura 2 apresenta a área de ***Dashboard*** da plataforma. No item 1 refere-se aos caixas de atividades, o item

2 a Ação de pular, o item 3 o Perfil do usuário, item 4 a Ação de responder e o item 5 o Menu do Usuário.

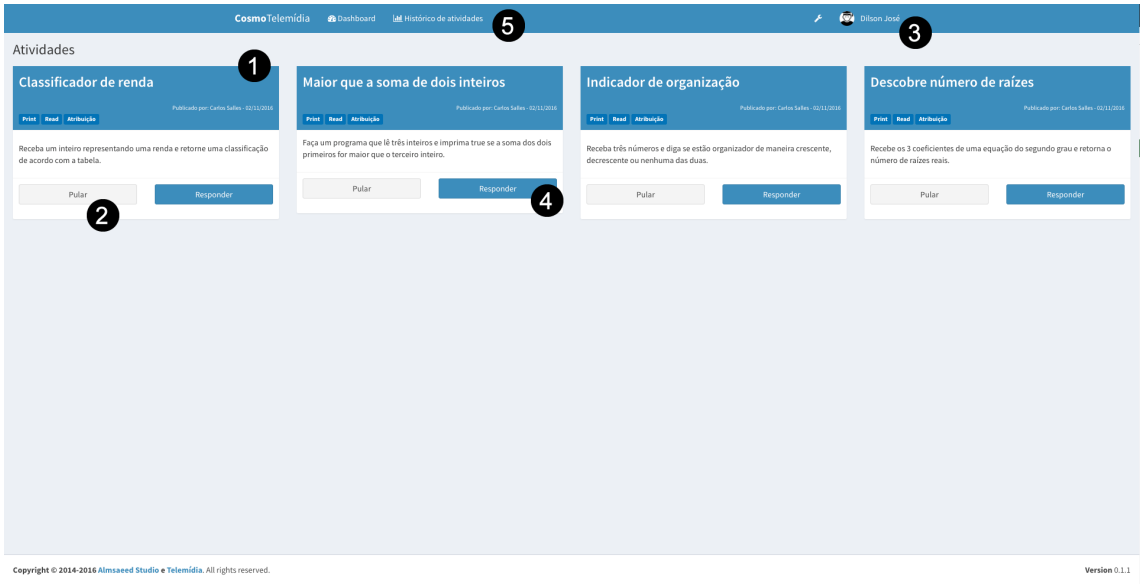


Figura 2. *Dashboard* do Cosmo

A composição das atividades no *Dashboard*, se dá de forma randômica. Cada caixa de atividade possui um título, um breve conceito para expor previamente o tipo de atividade, quem a criou e *tags*, elas indicam ao usuário de qual assunto que a questão está abordando.

3.2. Área de Resposta

Ao clicar no botão ”Responder”, em uma caixa de atividade, o usuário é levado a uma tela chamada de Área de resposta, conforme apresentado na Figura 3. O conteúdo que o usuário visualiza nessa tela é dinâmico, ou seja, deriva do *plugin* que está sendo instanciado no momento. Na Figura 3, vemos o item 1 que corresponde o enunciado da questão e em seguida o item 2 que apresenta a forma como o *plugin* aceita a entrada de resposta para a atividade. Neste exemplo pode ser observado o plugin ”Resolver Problemas”.

3.3. Arquitetura

A arquiteutra do Cosmo segue o modelo Cliente-Servidor. A Figura 4 apresenta a arquitetura do Cosmo.

No seu *Front-End* foram utilizados as tecnologias HTML5, CSS3 e Javascript. Sua interface foi desenhada com auxílio do AdminLTE [1] e suporte do JQuery [2]. Em seu *Back-End*, o Cosmo foi escrito com auxílio da linguagem de programação PHP [3].

3.4. Plugins

O Cosmo é um ambiente extensível a *plugins*, ele tem a capacidade de incorporar qualquer tipo de atividade que seja renderizada pelo HTML5. Isso possibilita que seja possível

[1] https://adminlte.io/themes/AdminLTE/index2.html
[2] https://jquery.com/
[3] http://www.php.net/

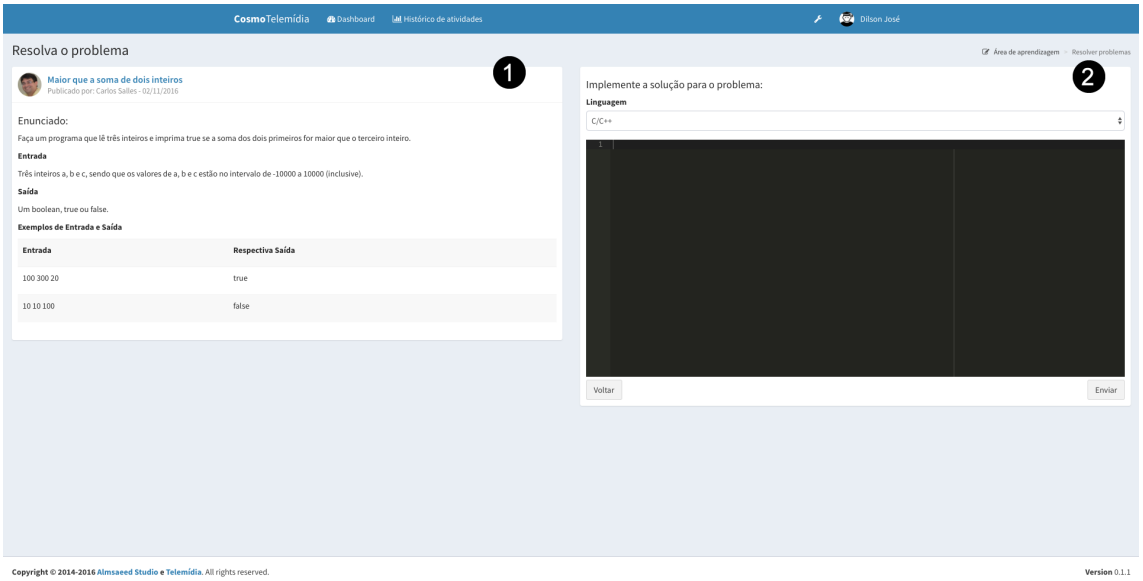


Figura 3. Área de Resposta

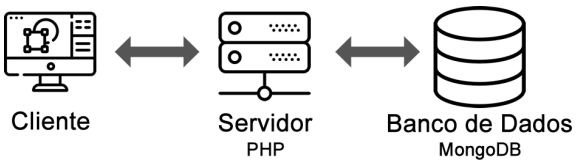


Figura 4. Arquitetura do Cosmo

utilizar um jogo como atividade, um vídeo direto de um provedor de *stream*, a leitura de um documento, as possibilidade são inúmeras.

A Figura 5 apresenta a estrutura de *plugin* do Cosmo.

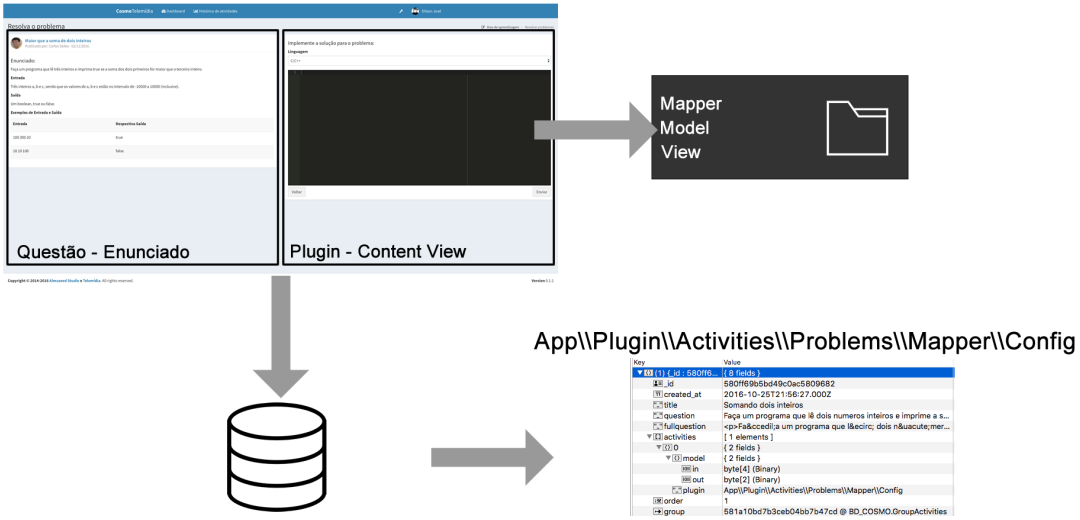


Figura 5. Estrutura do Plugin

A Área de Resposta, já apresentada anteriormente, é onde o *plugin* é instanciado e exibido. Ao ser instanciado o Cosmo monta a área de Enunciado e o *Content View*.

Cada *plugin* possui uma estrutura isolada do núcleo do ambiente e todo acesso ao núcleo é feito através de *interfaces*. Basicamente um *plugin* possui um estrutura de 3 pastas, a *Model*, a *View* e a *Mapper*.

O diretório *Model*, armazena todos os elementos que possam auxiliar a execução da tarefa. Nela pode ser aplicado elementos que não parte da estrutura do Cosmo.

O diretório *View*, armazena o HTML que será apresentado no *Content View*. Para auxiliar o desenvolvimento é utilizado um *template engine*, o Twig [4].

A *Mapper*, é responsável por mapear as configurações do tipo de tarefa para a estrutura de dados do Cosmo, validar a atividade, integrar a tarefa ao banco de dados, especificar a *view* e definir o modelo da tarefa.

4. Testes e Resultados

O Cosmo atualmente se encontra na versão *beta* e com a um tipo de tarefa disponível. Com essa versão da aplicação, foi preparado um teste preliminar com uma turma do 2º período da disciplina de algoritmos. Os testes foram realizados uma semana antes da primeira prova da disciplina. Estava previsto a presença de 56 membros, sendo confirmados 34 presenças e 22 ausências. O período de monitoramento da ferramenta ocorreu no prazo de duas semanas, nesse período, o sistema estava armazenando dados como tempo de uso da ferramenta em horas e dias, número de questões respondidas e o tempo médio de uso da ferramenta.

A base de dados do ambiente continha um total de 22 tarefas do tipo ”Resolver o Problema”.

Esse tipo de tarefa o aluno recebe um enunciado o tipo de entrada de dados e sua respectiva saída, com base nesses dados o aluno precisa desenvolver o código que atenda a bateria de testes contida dentro do Cosmo. Esse tipo de atividade oferece o suporte as linguagens Lua, C/C++ e Python.

Todas as tarefas foram elaboradas por um professor especialista, seguindo a temática da prova e classificadas em uma escala de Fácil Médio e Difícil.

A análise dos dados de respostas dos alunos foram cruzados com os resultados da prova, a partir delas pode-se concluir que: 7,14% dos alunos que participaram do teste, são os alunos que tiveram o maior tempo de uso, maior número de questões respondidas (entre 13 e 17 questões) e obtiveram na prova notas acima de 9,5. Outra porção dos alunos, cerca de 12,5%, foram os alunos que passaram em média 124 minutos de uso da ferramenta, responderam entre 10 e 12 questões e obtiveram notas entre 4,5 e 10. Os demais alunos da amostra, especificamente 80,36% foram caracterizados dados de corte da amostra.

Notou-se que, ao que se propõe, o Cosmo cumpriu com as expectativas, pois contribuiu no aprendizado dos alunos no período próximo a prova da disciplina de algoritmos. A assertiva foi coletada a partir de um questionário expectativa aplicado após a prova, ava-

[4]https://twig.symfony.com/

liando o uso do Cosmo para auxílio da disciplina de algoritmos. Mais de 55% dos alunos responderam que sim o ambiente pode auxilia-los a fixar o conteúdo da disciplina de algoritmos.

5. Trabalhos Futuros

Os primeiros testes do Cosmo foram satisfatórios pelo que o ambiente propõe. Durante os teste pode-se perceber que algumas funcionalidades precisam ser revisadas, como exemplo o *Log* da aplicação que precisa ser refinado.

Novos requisitos foram levantados durante os testes, a partir dos questionários de expectativas. Novos modelos de atividades foram propostas, pedido de tutoriais, área para reportar *bugs*, gamificação (já previsto pela equipe), dentre outros requisitos.

Na próxima versão do Cosmo serão incorporados novas atividades e uma versão beta de um algoritmo de tutoria esta sendo desenvolvido para recomendação de atividades ao aluno.

6. Conclusão

É evidente que um dos principais problemas enfrentados pelos cursos de graduação na área de informática é a alta taxa de desistência. Deve-se então considerar o papel crítico da disciplina de algoritmos. Tal disciplina representa para um grande grupo de estudantes seu primeiro contato com programação de computadores. Experiências negativas podem desencorajar os estudantes de um estudo mais aprofundado e afetar seriamente a sua motivação.

Buscar alternativas para reduzir as taxas de abandono é tornar o ensino de programação mais prático e concreto. Essa é a proposta apresentada pelo ambiente Cosmo, um ambiente virtual multitarefa focado para o ensino de algoritmos. Com a capacidade de incorporar novos *plugins*e permitir que seja possível diversificar a experiência do usuário no processo de ensino-aprendizagem.

Referências

Carvalho, M. d. and Tafner, P. (2006). Ensino superior brasileiro: a evasão dos alunos e a relação entre formação e profissão. *Anais do 30º Encontro anual da ANPOCS*.

Conway, M. J. and Pausch, R. (1997). Alice: easy to learn interactive 3d graphics. *ACM SIGGRAPH Computer Graphics*, 31(3):58–59.

de Oliveira Brandão, L., da Silva Ribeiro, R., and Brandão, A. A. (2012). A system to help teaching and learning algorithms. In *Frontiers in Education Conference (FIE), 2012*, pages 1–6. IEEE.

dos Santos, R. P. and Costa, H. A. X. (2006). Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *INFOCOMP Journal of Computer Science*, 5(1):41–50.

Kazimoglu, C., Kiernan, M., Bacon, L., and Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, 47:1991–1999.

Palmeira, L. B. and Santos, M. P. (2014). Evasão no bacharelado em ciência da computação da universidade de brasília: análise e mineração de dados. *Monografia do curso de Ciência da Computação da UnB–Universidade de Brasilia. Brasília: UnB*.

Rapkiewicz, C. E., Falkembach, G. A. M., Seixas, L. M. J. d., Santos, N. d. S. R. S. d., Cunha, V. V. d., Klemann, M., et al. (2007). Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE: revista novas tecnologias na educação [recurso eletrônico]. Porto Alegre, RS*.

Tillmann, N., Bishop, J., Horspool, N., Perelman, D., and Xie, T. (2014). Code hunt: searching for secret code for fun. In *Proceedings of the 7th International Workshop on Search-Based Software Testing*, pages 23–26. ACM.

Topalli, D. and Cagiltay, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with scratch. *Computers & Education*, 120:64–74.

Oficinas de Programação para Meninas: Despertando o Interesse Pela Computação

Giorgia de O. Mattos, Josilene A. Moreira, Ana Flávia S. A. Moura, Andrea B. Nascimento, Chaenne C. Oliveira

Centro de Informática – Universidade Federal da Paraíba (UFPB)
Rua dos Escoteiros s/n, Mangabeira – 58.055-000 – João Pessoa – PB – Brazil

{giorgia,josilene}@ci.ufpb.br, {anaflavia7x, andrea.bnec, chaecpnoliveira}@gmail.com}

Abstract. *The present report describes the activities carried out in an extension project that has been working in the education of students of the first year of high school, promoting, through programming workshops, the real contact with the area of computation. Data collected during the three years of the project show that most of the students do not have prior knowledge about programming and consider the activity difficult; it was also observed that they are motivated, interested and challenged in learning to program and perceive that the area of computation is also for them. From the evaluation at the end of the action, it can be seen that the workshops have been fulfilling their role in promoting female empowerment as well as contributing to the girls' citizenship.*

Resumo. *O presente relato descreve as atividades realizadas em um projeto de extensão que vem atuando na educação de alunas do primeiro ano do ensino médio, promovendo, através de oficinas de programação, o contato real com a área de computação. Dados coletados durante os três anos do projeto mostram que a maioria das alunas não tem conhecimento prévio sobre programação e consideram a atividade difícil; observou-se ainda que elas se mostram motivadas, interessadas e desafiadas em aprender a programar e percebem que a área de computação também é para elas. A partir da avaliação ao final da ação, pôde-se constatar que as oficinas vem cumprindo seu papel na promoção do empoderamento feminino bem como contribuindo com a formação cidadã das meninas.*

1. Introdução

A presença feminina nas áreas de ciência e tecnologia vem sendo amplamente estudada nos últimos anos [Beaubouef and Zhang 2011] [Oliveira et al 2014]. Segundo o último Censo da Educação Superior do Brasil [INEP 2016], em 2016, as mulheres representaram 44% dos alunos matriculados nos cursos de graduação porém apenas 14% delas estavam em cursos da área de Computação [SBC 2016].

Estudos mostram que existe uma série de fatores que levam ao desinteresse pela computação por parte das mulheres [Burge and Suarez 2005] [Beaubouef and Zhang 2011] [Eney et al 2013] [DuBow 2013] [Klawe 2013]. Dentre as principais causas estão a falta de "modelos femininos" na área, o fato das mulheres se sentirem excluídas em

ambientes (estudantis ou profissionais) dominados por homens, a falta de incentivo para que sigam uma carreira na área e o fato de não se sentirem à vontade com a cultura da computação/tecnologia.

Em 2014 o Centro de Informática da Universidade Federal da Paraíba realizou um diagnóstico com o intuito de levantar as principais dificuldades enfrentadas pelas alunas da graduação tanto para o ingresso como para a permanência nos cursos da área de Ciência da Computação do Centro de Informática [SALES et al 2014]. Segundo o diagnóstico as principais dificuldades para a sua permanência são com as disciplinas de Cálculo e com as disciplinas da área de programação onde apenas 17% das graduandas já tinham algum conhecimento sobre programação quando iniciaram o curso. Assim, o relato apresentado aqui descreve como o projeto de extensão vem atuando na educação de alunas do primeiro ano do ensino médio, promovendo através das Oficinas de Programação o desenvolvimento do raciocínio lógico e conceitos de programação de computadores. Através de um contato real com a área de programação, o projeto objetiva ser um agente transformador que atraia a atenção das alunas do ensino médio, possibilitando que elas possam conhecer de forma mais abrangente a carreira, antes de sua escolha para o vestibular.

O restante do artigo está organizado como segue: a Seção 2 traz estudos que tratam do ensino de programação para meninas; na Seção 3 são apresentados os materiais e métodos utilizados nas oficinas de programação; na Seção 4 são reportados os resultados obtidos e observados bem como algumas discussões a respeito; e, por fim, as conclusões e perspectivas futuras.

2. Trabalhos Correlatos

A programação de computadores é uma atividade base da área de computação e para muitos estudantes ela não é simples. A criatividade, o raciocínio lógico e a capacidade para a resolução de problemas são habilidades fundamentais para o sucesso na programação e estas devem ser estimuladas durante toda a formação escolar do aluno [Cristóvão 2008]. A prática intensiva e a motivação para resolver vários exercícios são itens importantes para o sucesso do aluno [Robins et al 2003].

A programação para meninas ainda é pouco explorada mas alguns estudos sugerem que intervenções feitas pelas universidades, com atividades de programação em escolas podem ser bem eficazes, contribuindo, inclusive, para a diminuição de estereótipos masculinos ligados à computação [Cohoon 2002]. Nos Estados Unidos, por exemplo, iniciativas como “Girls on the GO: The Mobile Computing College Experience” [Burge et al 2013] e a descrita em [Marcu et al 2010] ensinam programação para meninas do ensino médio, procurando incentivar o interesse feminino pela Ciência da Computação.

No Brasil, as pesquisas existentes relatam experiências com alunas do ensino fundamental e médio, onde a programação é utilizada, também, como meio de incentivar a presença feminina na área. O programa Meninas Digitais [Maciel and Bim 2016] da Sociedade Brasileira de Computação tem como objetivo divulgar a área de Computação para despertar o interesse de estudantes do ensino médio/tecnológico ou dos anos finais do ensino fundamental. É uma iniciativa voltada para o público feminino, para que elas conheçam melhor a área e, desta forma, sintam-se motivadas a

seguir carreira em Computação. O relato apresentado em [Souza et al 2013] descreve as atividades de programação com o Scratch, realizadas com alunas do ensino fundamental. Segundo os autores a utilização de jogos e o ambiente Scratch atraem a atenção e o interesse das meninas pois trabalham habilidades e conceitos básicos de programação de maneira descontraída e divertida; a principal dificuldade foi relacionada a lógica de programação. O projeto *Android Smart Girls* [Ramos et al 2015] ensina programação para meninas do ensino médio de uma escola de Campinas-SP. A ferramenta de desenvolvimento de aplicativos para dispositivos móveis MIT App Inventor foi utilizada. Em [Mattos et al 2015] os autores descrevem a utilização de kits de robótica educacional para o ensino de programação a meninas. As atividades descritas envolvem as fases de montagem do robô e a sua programação, com ambiente de programação próprio. De modo geral as alunas se mostraram motivadas, interessadas pela atividade e com vontade de aprender mais.

3. Materiais e Métodos

As atividades envolvendo as oficinas de programação vem sendo realizadas no Colégio da Polícia Militar Estudante Rebeca Simões, na cidade de João Pessoa, no período 2015-2017, junto às alunas do primeiro ano do ensino médio. A equipe executora é formada por duas professoras do Centro de Informática da UFPB e duas alunas de graduação dos cursos de Ciência da Computação e Engenharia da Computação. Os encontros aconteceram quinzenalmente, de maio a dezembro, com duração de aproximadamente 4 horas cada. As atividades realizadas compreendem aquelas que estão descritas na Tabela 1 bem como os conteúdos abordados em cada etapa e a ferramenta utilizada.

Tabela 1. Descrição das atividades realizadas e conteúdos abordados.

Atividade desenvolvida	Conteúdo abordado	Ferramenta utilizada
Diagnóstico do perfil da turma e palestra de abertura do projeto	- questões sobre o conhecimento prévio de programação e raciocínio lógico - apresentação dos cursos da área de Computação oferecidos pelo CI/UFPB	Questionário Palestras
Oficinas com jogos sérios	- definição do problema - conceito de instrução - sequência lógica de instruções	Lightbot
Oficinas de algoritmos	- variáveis - estrutura sequencial, condicional e de repetição - funções - depuração	Code.org
Oficinas de programação	- implementação dos conceitos aprendidos nas oficinas de algoritmos	Scratch App Inventor

3.1 Atividade 1: Diagnóstico do perfil da turma e palestra de abertura do projeto

Esta primeira atividade tem como objetivo conhecer a turma participante das oficinas e, para tal, são aplicados dois questionários. O primeiro contem 13 questões de cunho pessoal sobre programação (conhecimento prévio sobre o assunto) e a utilização de computadores, celular e internet (se utiliza, com que frequência e com que finalidade). O segundo questionário aborda 8 questões simples de lógica como ordenar palavras, sinônimos e antônimos. As informações relevantes destes questionários estão discutidas na próxima seção.

A palestra de abertura do projeto é de responsabilidade do grupo Meninas na Computação do Centro de Informática da UFPB, que desenvolve desde 2014 junto a alunas de escolas públicas, a divulgação dos cursos da área de computação sempre discutindo e estimulando a presença feminina na área. Nesta palestra as alunas conhecem um pouco sobre a atuação e o perfil do profissional de computação, o mercado de trabalho e as histórias de algumas mulheres de destaque na área.

3.2 Atividade 2: Oficinas com jogos sérios

A segunda atividade proposta utiliza o jogo sério Lightbot (http://armorgames.com/play/2205/light-bot) e tem como objetivo estimular o raciocínio lógico, a organização do pensamento computacional e a capacidade de solucionar problemas.

O Lightbot é um jogo online composto de três fases (básico, procedimentos e laços) onde um robô precisa se movimentar para acender as luzes dos ladrilhos mais escuros. Para tal, existe um conjunto de comandos (andar, acender, girar e pular) que devem ser organizados de forma que, quando executados em sequência, façam com que o robô atinja o objetivo de cada etapa. Na fase 1 são abordados conceitos básicos de algoritmos onde o objetivo é organizar os comandos disponíveis para que cada uma das etapas seja cumprida. Na fase 2 são apresentados conceitos de programação em bloco (procedimentos), ou seja, um bloco de comandos pode ser reaproveitado sempre que for necessário. A fase 3 traz conceitos de recursividade, onde um bloco pode chamar a si próprio simplificando e diminuindo a quantidade de comandos utilizados. O objetivo do jogo é completar em até 1 hora todas as fases e ele faz parte do programa “A Hora do Código”.

Esta atividade consistiu em duas partes: teórica e prática. Na parte teórica foram selecionadas algumas atividades da fase 1 que foram realizadas numa folha de respostas. Depois de explicado o objetivo do jogo, cada uma das instruções e a atividade proposta, as alunas listaram, na folha de respostas, os comandos necessários para cumprir cada uma das atividades. Na parte prática executaram, utilizando a ferramenta, todas as atividades presentes na folha de respostas e compararam as suas respostas com a realidade do jogo, identificando e entendendo os seus erros.

3.3 Atividade 3: Oficinas com algoritmos

A terceira atividade desenvolvida foi trabalhar os conceitos básicos de programação como as estruturas de controle de execução de um programa (sequencial, condicional e de repetição) e conceitos avançados como a programação em blocos (funções e procedimentos), a depuração e a recursividade.

O ambiente utilizado foi o Code Studio (http://studio.code.org) que é uma parte do Code.org que disponibiliza cursos de programação completos para iniciantes, em diversas faixas etárias, que estejam interessados em aprender a programar. O ambiente divide-se em três áreas: a descrição do problema a ser solucionado, os blocos disponíveis e a área de trabalho onde os blocos são estruturados para solucionar o problema. Traduzido como "A Hora do Código do Brasil", o projeto visa desmistificar a programação e apresentar material de qualidade para que professores e alunos tenham acesso as matérias da área de forma amigável.

Os conteúdos são abordados através de jogos e cada jogo é composto por níveis. O ambiente disponibiliza também os planos de aulas, estatísticas com os níveis concluídos e a quantidade de linhas de código escritas por nível e o progresso individual em cada fase, mostrando, por exemplo, se a fase foi iniciada, se a fase foi concluída com o número adequado de blocos ou com muitos blocos.

3.4 Atividade 4: Oficinas de programação

A quarta e última atividade proposta consiste em utilizar um ambiente de programação que permita a criação de um programa, jogo ou aplicativo completo que aborde os conteúdos estudados na atividade 3. As ferramentas utilizadas foram o Scratch para a criação de jogos simples e animados e o MIT App Inventor para a criação de aplicativos para dispositivos com o sistema Android.

O Scratch é uma linguagem de programação gráfica cujo objetivo é auxiliar a aprendizagem de programação de maneira lúdica e criativa, podendo ser usado por crianças desde 8 anos de idade e pessoas que não possuem nenhum conhecimento de programação.

A ferramenta MIT App Inventor é utilizanda por quem não tem experiência em programação para criar aplicativos. O MIT App Inventor é uma plataforma de programação visual criada pela Google em parceria com o MIT (Massachussets Institute of Technology), na qual se pode criar um aplicativo para Android utilizando blocos lógicos de maneira simples e intuitiva.

3.5 Avaliação

A avaliação do processo ensino-aprendizagem foi realizada observando-se a turma durante as oficinas e relatando os acontecimentos em cada encontro. As anotações incluíram as dificuldades encontradas para realizar o planejamento do dia (estrutura física de laboratório e internet, por exemplo), se as alunas cumpriram todas as atividades programadas para o encontro, se as alunas cumpriram os desafios propostos, o relato das alunas participantes e o relato das alunas de graduação que conduziram as oficinas.

Ao término do projeto anual foi elaborado um teste com 6 questões de lógica, nos mesmos moldes daquele aplicado na primeira atividade, e mais 4 questões retiradas de provas da Olimpíada Brasileira de Informática – Modalidade Iniciação, onde as tarefas das provas consistem em problemas de lógica e problemas de computação (lógica de programação, mas sem o uso do computador).

4. Resultados e Discussões

Ao longo de 3 anos do projeto de extensão, 18 alunas do 1º ano do ensino médio participaram efetivamente das oficinas de programação. O projeto não foi idêntico nos 3 anos, especialmente da atividade 4 onde foram utilizadas ferramentas diferentes em 2016 (Scratch) e 2017 (MIT App Inventor). A equipe do projeto é formada por duas professoras e alunas de graduação (bolsistas e voluntárias) dos cursos de Ciência da Computação e Engenharia de Computação que sempre conduziram as oficinas.

O questionário aplicado na atividade 1 nos permitiu conhecer previamente as turmas e consequentemente organizar e planejar melhor as atividades de acordo com os dados levantados. Das respostas obtidas apenas uma aluna já tinha algum conhecimento prévio sobre programação e o grau de dificuldade desta atividade foi considerado 4 (numa escala de 1 a 5, sendo 5 muito difícil). 72% delas possuem computador em casa e o utilizam diariamente, o acesso às redes sociais e os jogos online são as principais atividades realizadas por elas. Com relação as questões de lógica, a média de acertos ficou em 3 questões de um total de 8, sendo a questão 2 a que aparece com mais erros. As alunas consideraram as questões difíceis e relataram não entender o enunciado ou não conseguirem identificar o padrão lógico pedido na questão.

O Lightbot foi utilizado em dois momentos distintos: no primeiro momento as alunas receberam a descrição das oito tarefas do primeiro nível do jogo, o conjunto de instruções disponíveis e uma folha de respostas onde deveriam anotar as instruções necessárias para a solução das tarefas; o segundo momento consistiu em jogar o jogo completo em até no máximo 1 hora. A média de acertos das tarefas ficou em 45%, sendo a maior dificuldade na interpretação das instruções (virar a direita, virar a esqueda e utilização da seta para frente ao invés da instrução de pular). O tempo médio para a conclusão do jogo ficou em 54 minutos e ao término da atividade o próprio jogo gera um “Certificado de Conclusão *Hour of Code*”.

Na terceira atividade foram trabalhados os conceitos básicos de algoritmos e programação com o ambiente Code Studio. Foram utilizados partes dos cursos 3 e 4 de acordo com os conteúdos abordados (sequência, laços, condicional, repetição, depuração, variáveis e funções) e o acompanhamento individual das alunas feito no próprio ambiente. A figura 1 mostra a aba “Estatísticas” selecionada e as informações de cada aluna com relação aos níveis concluídos, dentro das atividades propostas, e a quantidade de linhas de código escritas pela aluna.

Nome	Níveis concluídos	Linhas de código
Giovanna Bvb	31	194
Anna	103	1112
Yasmin Carvalho	93	608
Annielly Nunes	38	278
Dudão	47	327
Glenda Hellen	26	179

Figura 1. Acompanhamento das estatísticas individuais de cada aluna considerando os níveis concluídos e linhas de código escritas.

A figura 2 mostra a aba "Progresso" selecionada e o desempenho de cada aluna em cada uma das fases. É interessante destacar a importância desse acompanhamento pois através dele é possível identificar se as tarefas estão sendo realizadas com êxito ou não, pois a ferramenta verifica e analisa a quantidade de blocos/instruções que cada aluna está usando em cada fase.

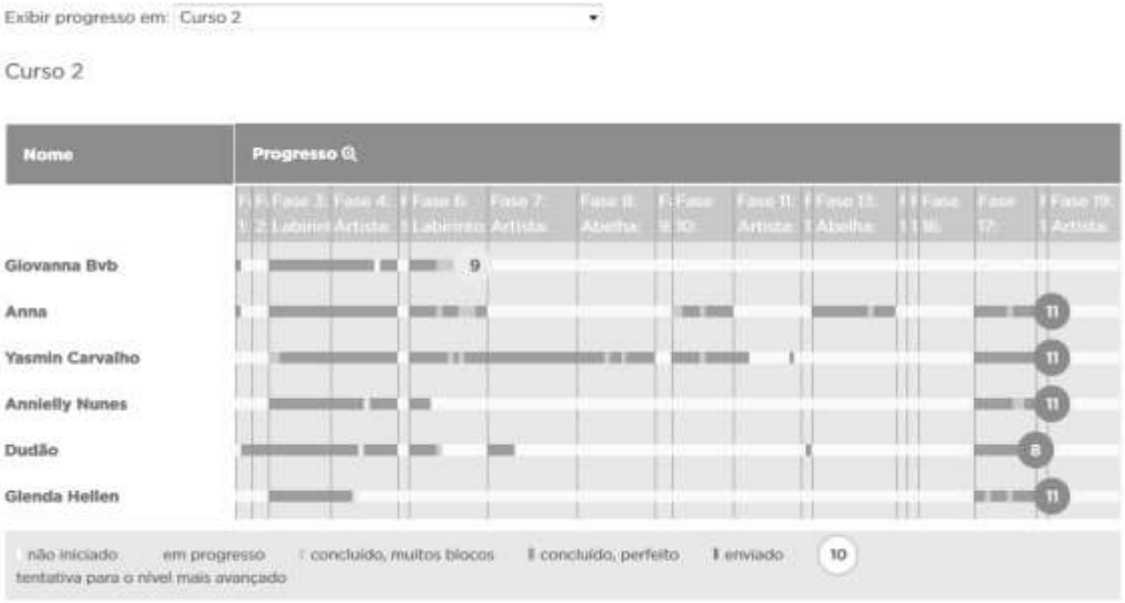


Figura 2. Progresso individual de cada aluna em cada uma das fases propostas na atividade.

A avaliação desta fase é feita utilizando os recursos disponibilizados pela ferramenta Studio Code, pela observação das instrutoras em sala de aula e pelas alunas. Algumas alunas consideram a atividade interessante porém cansativa devido à

quantidade de tarefas selecionadas e muitas vezes repetidas dentro da mesma fase. As instrutoras observam a mesma situação, as alunas iniciam a atividade empolgadas mas ao longo dos encontros previstos elas reclamam da repetição.

Na última fase do projeto as alunas tem certa liberdade de criação pois colocam em prática os assuntos abordados no Studio Code. Para tal, já foram utilizadas as ferramentas Scratch e MIT App Inventor. Na ferramenta Scratch foram desenvolvidos pequenos jogos animados e no decorrer dos encontros as alunas foram aprimorando o jogo e inserindo novas funcionalidades à ele de acordo com os assuntos previstos. Alguns exemplos de tarefas desenvolvidas foram as simulações de movimentos de dança e dança ao sons de um tambor.

A atividade com o MIT App Inventor foi considerada bastante estimulante pelas instrutoras e alunas pois o desenvolvimento de aplicativos para dispositivos móveis é um tema atual e que desperta o interesse dos alunas. As atividades realizadas incluíram o desenvolvimento de uma Calculadora, uma Agenda e uma aplicação chamada de Papagaio (TalkToMe) onde a fala era convertida em texto e reproduzida em forma de som pelo aplicativo. De acordo com os relatos das instrutoras, as atividade foram realizadas satisfatoriamente pelas alunas com todas conseguindo realizar a sua aplicação e motivadas a incluir novas funcionalidades ao seu aplicativo. A figura 3 mostra as telas de dois aplicativos, Calculadora e Papagaio, desenvolvidos pelas alunas.

Figura 3. Tela dos aplicativos desenvolvidos.

Na avaliação realizada ao término do projeto anual foi aplicado um teste, que serviu para selecionar uma aluna bolsista PIBIC-EM (Ensino Médio), contendo 6 questões de lógica, nos mesmos moldes daquele aplicado na primeira atividade, e mais 4 questões retiradas de provas da Olimpíada Brasileira de Informática – Modalidade Iniciação. Cinco alunas participaram desta avaliação e a quantidade de acertos foi,

respectivamente, 9, 8, 6, 5 e 3 questões. Quando perguntado a respeito das oficinas e a sua importância, todas elas relataram que se surpreenderam com as aulas, os conteúdos estudados e as ferramentas utilizadas bem como a sua satisfação pessoal em conseguir, elas mesmas, criar algo que possa ser útil a alguém.

5. Considerações Finais

A participação feminina em carreiras ou cursos de graduação ligados à Ciência e Tecnologia é baixa e inúmeras iniciativas vem sendo desenvolvidas, especialmente no Brasil, com o objetivo de mostrar às alunas do ensino fundamental e médio as reais possibilidades de atuação do profissional e o mercado de trabalho. Este relato apresentou as atividades que vem desenvolvendo há 3 anos no Colégio da Polícia Militar em João Pessoa-PB, junto as alunas do 1º ano do ensino médio com o intenção de trabalhar conceitos básicos de programação de computadores e que estão presentes no dia a dia de todo profissional da área de Computação.

Os dados obtidos em pesquisas com as alunas mostram que poucas tem algum conhecimento prévio sobre programação e consideram a atividade de programar difícil. Muitas delas tem computador em casa e celular e as principais atividades realizadas são o acesso às redes sociais e jogos online. De maneira geral as alunas se sentem motivadas, interessadas e desafiadas com as atividades propostas, embora algumas destas atividades sejam consideradas massantes e repetitivas. O desenvolvimento de aplicativos com o MIT App Inventor foi a atividade mais desafiadora e estimulante pra elas pois puderam por em prática todos os conceitos estudados anteriormente com maior liberdade e criando as suas próprias aplicações. Embora seja um projeto de médio prazo, os primeiros resultados começam a aparecer. No vestibular de 2018 da UFPB tivemos uma aluna participante da primeira turma aprovada no concurso para um curso da área de Computação. Ainda não tivemos a oportunidade de fazer um acompanhamento e levantamento das escolhas profissionais das alunas participantes das oficinas até o ano de 2017. Os trabalhos futuros incluem este acompanhamento das escolhas profissionais, o impacto das oficinas de programação no rendimento escolar das alunas participantes bem como a revisão dos conteúdos abordados, das ferramentas utilizadas e a possibilidade de replicar o modelo apresentado para outras escolas da cidade.

A experiência das Oficinas de Programação junto ao público feminino vem sendo extremamente satisfatória sob dois aspectos principais: a participação das alunas do ensino médio que tem a oportunidade de aprender programação imersas em ambientes lúdicos e próprios para quem nunca programou; e a participação das alunas graduandas dos cursos da área de Computação que tem a possibilidade de compartilhar com a sociedade aquilo que vivenciam no ambiente acadêmico e o seu fortalecimento como profissional feminino da área.

Referências

Beaubouef, T. and Zhang, W. (2011) “Where are the women computer science students?” In: *Journal of Computing Sciences in Colleges*, Consortium for Computing Sciences in Colleges, USA, v. 26, n. 4, pages 14-20.

Burge, J. D. and Suarez, T. L. (2005) "Preliminary Analysis of Factors Affecting Women and African Americans in the Computing Sciences" In: *Proceedings of the 2005 Conference on Diversity in Computing*, ACM, USA, pages 53-56.

Cohoon, J. M. (2002). Recruiting and retaining women in undergraduate computing majors. ACM SIGCSE Bulletin, 34(2), ACM, USA, pages 48-52.

DuBow, W. M. (2013) "Diversity in Computing: Why It Matters and How Organizations Can Archive It" Computer, IEEE Publisher, v.46, n.3, pages 24-29.

Eney, C., Lazowska, E., Martin, H. and Reges, S. (2013) "Broadening Participation: The Why and the How" Journal Gender Diversity in Computer, IEEE Publisher, v.46, n.3, pages 48-51.

INEP. (2016) "Censo da Educação Superior 2016 – Principais Resultados", http://download.inep.gov.br/educacao_superior/censo_superior/documentos/2016/censo_superior_tabelas.pdf.

Klawe, M. (2013) "Increasing Female participation in Computing: The Harvey Mudd College Story" Computer, IEEE Publisher, v. 46, n. 3, pages 56-58.

Maciel, C. and Bim, S. A. (2016) Programa Meninas Digitais – Ações Para Divulgar a Computação para Meninas do Ensino Médio. Computer on the beach 2016, https://siaiap32.univali.br/seer/index.php/acotb/article/view/10742, pages 327-336.

Mattos, G. O., Silva, D. R. D. and Moreira , J. A. (2015) "A Utilização de Kits de Robótica como Ferramenta para o Ensino de Programação à Meninas do Ensino Médio" In: *XXIII Workshop sobre Educação em Computação*, Publicado pela SBC, pages 2085-2093.

Ramos, N., Freitas, C., Avila, S., Costa, P. D. P., Testoni, V. and Borin, J. F. (2015) "Ensino de Programação para Alunas de Ensino Médio: Relato de uma Experiência" In: *XXIII Workshop sobre Educação em Computação*, Publicado pela SBC, pages 2089-2098.

Robins, A., Rountree, J. and Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), pages 137–172.

Sales, A., Calado, B., Silva, D. R. D., Mattos, G. O. and Moreira, J. A. (2014) "Dificuldades para o Ingresso e Permanência na Ciência e Engenharia da computação: Um Olhar Feminino" 18º REDOR, Editora da UFPB, pages 3468-3482.

Souza, S. M., Rios, M. S., Rodrigues, C. A., Santos, D. M. B. and Bittencourt, R. A. (2013) "Oficinas de Programação com Ambientes Lúdicos Para Meninas do Ensino Fundamental", In: *XXI Workshop sobre Educação em Computação*, Publicado pela SBC, pages 959-968.

Oliveira, A., Moro, M. M. and Prates, R. O. (2014) "Perfil Feminino em Computação: Análise Inicial", In: *XXII Workshop sobre Educação em Computação*, Publicado pela SBC, pages 1465-1474.

SBC. (2016) "Educação Superior em Computação – Estatísticas 2016" Disponível em http://sbc.org.br/documentos-da-sbc/send/133-estatisticas/1167-educacao-superior-em-computacao-estatisticas-2016.

Desenvolvimento e Avaliação de uma Metodologia de Aprendizagem Ativa Apoiada pelo uso de QR Code para Ensino de Banco de Dados

Ronney Moreira de Castro, Sean Wolfgand Matsui Siqueira
Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)
Av. Pasteur, 456 - Urca - Rio de Janeiro - RJ - Brasil
ronney.castro@uniriotec.br, sean@uniriotec.br

Abstract. *Considering the development of information and communication technologies and the reality of the students, the traditional lectures tend to be monotonous and uninteresting. Active Learning is an option to this type of class, bringing techniques closer to the students, resulting in greater motivation and involvement on their part. This paper presents a technique based on QR Code as a support to teaching. The authors applied the technique in a Database course of Information Systems. The goal is to provide students with a more practical and engaging activity encompassing knowledge explored in the course and also help teachers in choosing a different methodology to be adopted in their classes.*

Resumo. *Diante do desenvolvimento das tecnologias da Informação e comunicação e a realidade dos alunos, a aula tradicional expositiva tende a ser monótona e desinteressante. A Aprendizagem Ativa (AA) é uma opção a este tipo de aula, trazendo técnicas mais próximas dos alunos, resultando em uma maior motivação e envolvimento por parte deles. Este artigo apresenta uma técnica baseada em QR Code como apoio ao ensino. Os autores aplicaram a técnica em uma disciplina de Banco de Dados do curso de Sistemas de Informação. O objetivo é levar aos alunos uma atividade mais prática e envolvente englobando conhecimentos vistos na disciplina e também auxiliar professores na escolha de uma metodologia diferente a ser adotada em suas aulas.*

1. Introdução

A sala de aula tradicional, que apoia a sociedade industrial, baseia-se na figura central do docente, que é o detentor e disseminador do conhecimento e os alunos, por sua vez, têm pouco ou nenhum espaço para suas opiniões [Araújo et al. 2015]. O tipo de aula mais utilizado é a tradicional ou mais conhecida como expositiva. Nela, o professor é aquele que possui o conhecimento e passa o mesmo aos alunos, que devem dominar informações no lugar de realmente aprenderem a ter suas próprias opiniões [Thomas e Brown 2011].

Os currículos da sala de aula tradicional fundamentam-se, principalmente, no material elaborado que irá definir o que e como ensinar, acomodando as práticas escolares e estabelecendo uma fronteira pedagógica que distancia professores de alunos, restringindo a forma de pensar [Carvalho; Nevado; Menezes 2005]. Observa-se, assim, que os métodos de ensino adotados mostram-se ineficazes para a formação dos alunos e,

um dos maiores desafios da área da Educação está relacionado a quais práticas devem ser implantadas em sala de aula que possam favorecer os processos de aprendizagem [Acosta; Reategui; Behar 2016].

Na sociedade contemporânea, o pensamento crítico, a colaboração, a capacidade de solucionar problemas e a tomada de decisões são habilidades fundamentais [Acosta; Reategui; Behar 2016]. A aprendizagem está diretamente associada a um processo que ocorre em etapas de construção e reconstrução e é fundamental que os indivíduos possam interagir com outros sujeitos [Piaget 1977]. O aprendizado, então, necessita ter relação com o concreto, ou seja, o aluno deve vivenciar situações reais apoiado por um mediador/coordenador (professor) e construir seu conhecimento através dessas experiências [Behar 2011].

A Aprendizagem Ativa (AA) é uma alternativa em relação às aulas tradicionais podendo complementar as mesmas e motivar os discentes para um estudo mais detalhado, além de proporcionar uma maior retenção do conhecimento [Bonwell e Eison 1991] [Meyers e Jones 1993]. Segundo Acharya et al. (2015) várias áreas fazem uso da AA para o ensino de conteúdos e a Computação recentemente também tem feito uso de tal técnica [Mitchell; Petter; Harris 2017].

Este artigo apresenta uma forma de usar o QR Code como técnica de AA na disciplina de Banco de Dados (BD). O objetivo é levar os alunos a uma busca por informações contidas nos códigos e, juntar as mesmas para resolver uma atividade proposta. Os códigos contêm informações importantes tais como tabelas, atributos, relacionamentos, códigos SQL entre outros. Com isso, os alunos poderão fixar conceitos importantes aprendidos na disciplina de BD.

A técnica foi utilizada em duas turmas do 4º período de um Curso de Bacharelado em Sistemas de Informação, nos anos 2016 e 2017 e avaliada através da aplicação de um questionário e também da percepção dos autores durante a aplicação no laboratório. A dinâmica da técnica e os resultados da avaliação são apresentados no texto.

O restante do artigo está organizado em mais cinco seções. A seção 2 apresenta o conceito de Aprendizagem Ativa e sua utilização em cursos na área de Computação. A seção 3 descreve a técnica de QR Code propriamente dita, sua preparação, assim como a forma utilizada para aplicação em sala de aula. Os resultados alcançados estão na seção 4 e, por fim, a conclusão do artigo na seção 5.

2. Aprendizagem Ativa

A Aprendizagem Ativa (AA) surgiu em 1990 e provou ser uma excelente alternativa para auxiliar os alunos em um maior envolvimento com o aprendizado [Bonwell e Eison 1991] [Meyers e Jones 1993]. Diversas áreas têm feito uso da AA tais como, por exemplo, Biologia, Computação, Direito, Medicina, entre outras [Acharya et al. 2015].

Allen e Tanner (2005) descrevem a AA como a aquisição de novos conteúdos de forma a construir seu significado e com a oportunidade de reiterá-lo a outros, além disso, enfatiza a interação entre os discentes e o docente. No lugar de simplesmente ficarem ouvindo as aulas expositivas, os alunos têm inúmeras oportunidades de participar de atividades e, principalmente, receber o feedback imediato sobre elas [Gleason et al. 2011].

Prince (2004) define a AA como uma "atividade de sala de aula que exige que os alunos façam algo diferente de ouvir e tomar notas". Morgan et al. (2005) relatam que não há uma definição clara na literatura para AA, mas existem algumas comumente utilizadas: i) "Experiência de aprendizagem multidirecional na qual a aprendizagem ocorre de professor para aluno, aluno para professor e aluno para aluno"; ii) "Qualquer coisa que envolve os alunos em fazer coisas e pensar sobre as coisas que estão fazendo"; iii) "É definido em contraste com o pior do ensino tradicional, no qual os professores ativamente apresentam a informação e os alunos passivamente a recebem"; iv) "Qualquer método instrucional que envolva os alunos no processo de aprendizagem"; v) "Qualquer estilo de ensino que maximize a participação dos alunos no processo de aprendizagem". Mitchell; Petter; Harris (2017), afirmam que AA "são exercícios introduzidos na sala de aula para incentivar o pensamento e a participação dos alunos em um esforço para envolvê-los no processo de aprendizagem".

Para que uma atividade possa ser classificada como "ativa" é necessário que o aluno esteja ativamente envolvido na mesma e aplique o conhecimento construído de forma a fazer sentido para sua aprendizagem [Linnenbrink e Pintrich 2004]. As técnicas de AA incrementam as aulas e motivam os discentes, além de proporcionar uma maior assimilação do conhecimento. Alunos motivados são mais propensos a participar das aulas e, portanto, isso auxilia a sustentar o interesse e o prazer dos discentes pela aprendizagem [Barkley 2010].

Atividades como animações, dinâmicas de grupo, estudos de caso, jogos, quizzes, simuladores, vídeos são exemplos de AA. Segundo Thongmak (2017), a aprendizagem no século XXI tem-se baseado muito na AA. Para o autor, a AA é uma mudança na forma de aprendizagem permitindo aos docentes basear suas aulas em tarefas ou perguntas de forma a auxiliar os discentes a compreender melhor os conceitos envolvidos. Pesquisas recentes apoiam que os alunos valorizam a AA e a interação entre pares e que, estratégias como salas de aula invertidas, aprendizagem baseada em equipe, aprendizagem baseada em problemas e aprendizagem baseada em casos, podem ter excelentes resultados entre os alunos [Thongmak 2017].

Como qualquer estratégia, a AA não é a solução de todos os problemas de ensino-aprendizagem. Suas técnicas podem ser aplicadas levando em consideração os resultados de forma a tornar o processo de aprendizagem mais expressivo para o aluno [Drake 2012].

2.1. Aprendizagem Ativa em Sistemas de Informação/Computação

Mitchell; Petter; Harris (2017) elaboraram uma revisão de artigos publicados relacionados ao uso de AA em Sistemas de Informação. Os autores conduziram uma revisão sistemática da literatura usando a string de busca "*active learning*" E "*information systems*", no período de janeiro de 2000 a julho de 2016 em revistas acadêmicas da ABI/INFORM Collection, Informing Sciences Institute library e a eLibrary da Association for Information Systems (AIS). Foram selecionados 49 artigos e identificadas vinte tipos diferentes de técnicas agrupadas em 5 categorias: (1) apresentações visuais; (2) projetos colaborativos de estudantes; (3) interação tecnológica; (4) avaliação; e (5) jogos.

Acharya et al. (2015) utilizaram ferramentas de AA tais como exercícios em classe, estudos de caso e vídeo desenvolvidos em parceria com empresas de software

com o objetivo de compreender os tópicos relacionados à Engenharia de Software (validação e verificação de software, engenharia de requisitos, revisões, inspeções, gerenciamento de configuração e testes). Massey; Brown; Johnston (2005) fazem uso de jogos como técnica de AA para incentivar os alunos a revisar os conteúdos, além de envolvê-los melhor nas aulas. São propostos um jogo web de perguntas e respostas sobre diversos assuntos aplicado em sala de aula, e palavras cruzadas como tarefas para casa. Ramiller (2002) apresenta o Virtual Interactive Project (VIP), que faz um intercâmbio entre projetos de campo e projetos baseados em texto. Nele os alunos atuam em equipes como se fossem empresas de desenvolvimento e o professor como um "cliente virtual", disposto a responder aos questionamentos para o desenvolvimento de uma aplicação.

Dong et al. (2017) relata que é muito difícil para os docentes saber se todos os alunos estão envolvidos ou não na aula. Os discentes recebem informações do professor, porém ocorre pouca ou nenhuma interação, o que os leva a enfrentar dificuldade em manter a concentração. Uma forma de envolver o aluno e professor é através do uso de uma tecnologia projetada para isso. O trabalho apresenta uma ferramenta que faz uso de "Sistemas de Resposta Instantânea" (Instant Response Systems - IRSs) ou "clickers" que possibilita aos alunos solicitar uma pausa na aula quando necessário. Além disso, possibilita ao docente saber se o material de aprendizagem utilizado está sendo passado de forma rápida ou com dificuldade para os alunos compreenderem.

Castro e Siqueira (2017) apresentam algumas técnicas de AA utilizadas em um curso de Bacharelado em Sistemas de Informação. As técnicas foram: i) Uso de Quizzes, uma ferramenta desenvolvida pelos próprios alunos contendo um tabuleiro virtual que fica projetado usando um equipamento de Datashow em sala de aula. Os alunos devem se dividir em equipes e responder as perguntas sobre o assunto Engenharia de Software. Existem casas "coringas" que correspondem a prendas (ex: "imitar o professor X"; "cantar uma canção", "imitar o bicho Y" etc.) que servem para dar um caráter mais descontraído ao jogo; ii) Boneco de Jornal, utilizado para ensino do conceito de processos [Castro e Souza 2016]. Os alunos têm que se dividir em equipes e construir um boneco utilizando folhas de jornal e tesoura. Porém, existem fases a serem cumpridas. Em uma primeira fase, os grupos não têm comunicação com os outros, o que leva o boneco ficar como um "monstro", levando os alunos a entender, analogamente, que é importante a comunicação dentro de uma empresa de desenvolvimento de software, por exemplo. São aplicadas mais outras fases nas quais o boneco vai melhorando aos poucos, mostrando também que o processo não é estático, mas sim deve estar em melhoria contínua; iii) Jogo Agility Scrum para ensino do Scrum [Castro et al. 2017] no qual os alunos também se dividem em equipes e determinam os papéis dentro das mesmas: Scrum Master ou desenvolvedor. O professor faz o papel de Product Owner. As equipes têm que montar um circuito eletrônico que possui como componentes leds, baterias, suporte de bateria, resistores, fios, botões de acionamento e protoboard. Os Sprints são fixados no quadro da sala adaptado a um quadro de Kanbam e são sequenciais, ou seja, a equipe deve terminar o primeiro sprint, entregar o mesmo, logo depois construir o sprint seguinte. Os sprints são uma sequência de funcionalidades do circuito ficando mais complexo na medida que forem implementados. Ganha o jogo o grupo que terminar todo o circuito e entregar o mesmo em pleno funcionamento. A técnica também faz uso práticas do Movimento Maker, mais especificamente de circuitos eletrônicos. Este movimento tem como conceito fundamental, aprender com a

prática. No trabalho também foi citado o uso de QR Codes, porém de forma sucinta. Aqui detalhamos a técnica com o objetivo de divulgar a mesma para docentes que queiram adotá-la em suas aulas.

3. Uso de QR Code na Disciplina de Banco de Dados

A disciplina Banco de Dados está presente em praticamente todos os cursos na área de Computação. Muitas vezes divide-se a mesma em duas partes: uma que envolve somente a modelagem em si, usada para descrição de dados no nível conceitual (Modelo Conceitual) e outra sobre a implementação propriamente dita (Modelo Físico), que envolve o aprendizado da linguagem SQL para que o Banco de Dados seja criado em um Gerenciador de Banco de Dados (SGBD).

A técnica aqui proposta envolve o conhecimento tanto do modelo conceitual quanto do modelo físico, permitindo ao professor trabalhar com os alunos uma atividade para fixar a matéria lecionada. Para isso é utilizado QR Code, uma evolução dos códigos de barras no qual as informações são ordenadas em uma matriz de duas dimensões, conforme figura 1. Os QR Codes são utilizados largamente nos dias de hoje. Algumas de suas aplicações mais conhecidas são revistas, campanhas publicitárias, aplicativos, bilhetes de passagem etc.

Figura 1. Exemplo de um QR Code [Elaborado pelos autores]

Em uma busca na literatura foram encontradas algumas aplicações usando QR Code para fins acadêmicos. Porém, não foi encontrado nenhum trabalho relacionado ao uso de QR Code com Banco de Dados. Os detalhes da técnica elaborada usando os códigos, assim como sua preparação são mostrados na Tabela 1.

Tabela 1. Detalhes e preparação para aplicação da técnica

Objetivo	Mostrar de forma divertida conceitos aprendidos na disciplina Banco de Dados: Diagrama entidade-relacionamento (DER), Diagrama de tabelas relacionais (DTR), relacionamento entre tabelas, cardinalidade, tabelas, atributos, tipo de atributos (integer, char, varchar etc.).
Participantes	Todos os alunos da sala de forma individual.
Tempo	De 90 a 100 minutos. Em geral duas aulas germinadas.
Local	Usar um Laboratório que contenha PCs para a turma toda. Pode-se usar também dois alunos por PC. Além disso, é necessário ter instalado nas máquinas um SGBD (SQL Server, MYSQL, por exemplo) e um editor de textos.
Material Utilizado	20 folhas de papel A4 que deverão conter os QR Codes da tarefa. Celulares que possuam câmera (Smartphones). Um aplicativo de leitura de QR Codes, preferencialmente gratuito. Por conveniência, para o professor, é importante também deixar impresso o que significa cada QR Code. Fita adesiva para fixar as folhas nas paredes ou outro local. O material para aplicação da técnica pode ser solicitado por e-mail aos autores ou acessado nesse link: <https://goo.gl/LgnbtH>

Preparação	Solicitar aos alunos, em uma aula anterior, que baixem em seus celulares, um aplicativo para leitura de QR Code. No dia da aplicação, cerca de uma hora antes, imprimir as folhas contendo os QR Codes, embaralhar as mesmas e colar nas paredes, porta, móveis etc. do laboratório escolhido para aplicação da técnica usando fita adesiva. A figura 2 ilustra o laboratório com os QR Codes. Sugere-se fechar o laboratório e levar a turma inteira de forma conjunta para o mesmo. É importante que todos os alunos estejam presentes no início da dinâmica.

Figura 2. Distribuição das folhas com QR Code no laboratório

3.1. Descrição

Ao chegar no laboratório os alunos verão os códigos espalhados, o que já os deixará curiosos sobre o que farão naquela aula, além de gerar uma curiosidade sobre o que significa cada um dos códigos das folhas. O professor deve então explicar que existe um exercício a ser feito e que o texto e script do mesmo deverão ser entregues até o final da aula. Para isso será necessário fazer a leitura aleatória dos QR Codes localizados nas folhas. Logo após é dado o início por parte do professor e todos podem fazer a leitura dos códigos.

Existem folhas com trechos do enunciado do exercício (Exemplo: “Uma firma vende produtos de limpeza, e deseja melhor controlar os produtos que vende, seus clientes e os pedidos.”), informações de cardinalidades (Exemplo: “Guarda-se igualmente a informação dos pedidos feitos pelos clientes. Cada pedido possui um número e guarda-se a data de elaboração do pedido. Cada pedido pode envolver de um a vários produtos, e para cada produto, indica-se a quantidade deste pedida.”), partes de tabelas (Exemplo: “CodCliente char(3) Not Null, Nome varchar(50), Endereco varchar(50)”), tabela inteiras (Exemplo: “CodCliente char(3) Not Null, Nome varchar(50), Endereco varchar(50), Telefone varchar(15), Status char(5), LimiteCredito float(10,2)”), folhas coringa, que introduzem obstáculos a serem ser tratados pelos alunos (Exemplo: “Esse QR Code não tem nenhuma informação. Vá para outro QR Code”), dicas (Exemplo: “É necessário juntar as partes de texto para saber o enunciado da tarefa”).

O objetivo final não é ter um vencedor, mas que todos montem o exercício com seu enunciado e também façam o script de criação do Banco de Dados.

3.2. Aplicação

A técnica foi aplicada em duas turmas do 4º período de um curso de Bacharelado em Sistemas de Informação da cidade de Juiz de Fora - MG nos anos de 2016 e 2017. Na

instituição de ensino superior escolhida, a disciplina de Banco de Dados é dividida em duas partes: uma referente à Modelagem de Banco de Dados, no 3º período e outra referente a Implementação de Banco de Dados no 4º período. Todos os alunos já haviam cursado a parte de modelagem conceitual de Banco de Dados e já sabiam como criar as tabelas usando um script SQL. A Figura 3 mostra os participantes em 2016 e a Figura 4 os de 2017.

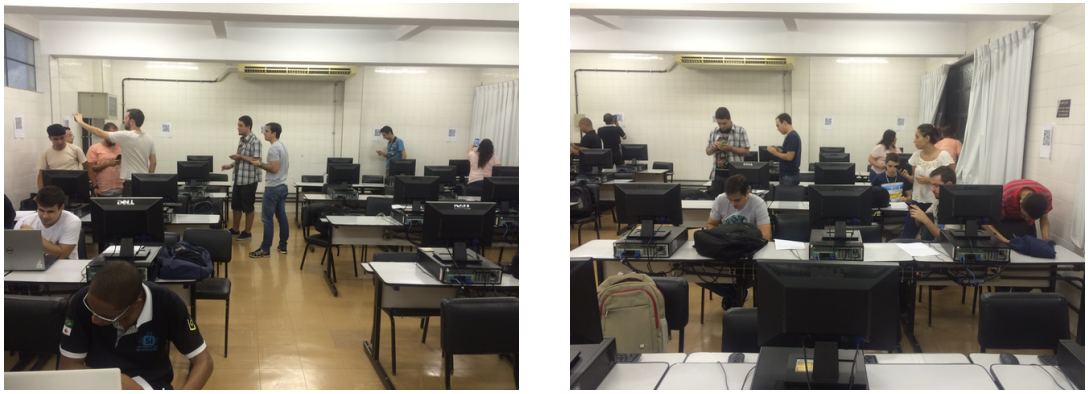


Figura 3. Alunos da turma de 2016 fazendo leitura dos QR Codes no laboratório

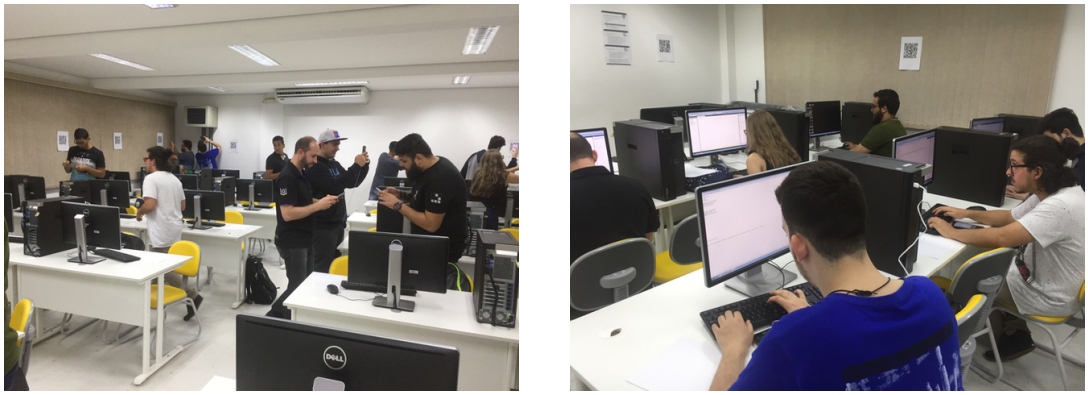


Figura 4. Alunos da turma de 2017 fazendo leitura dos QR Codes no laboratório

4. Resultados

Foi possível perceber através da observação dos participantes durante a tarefa, uma grande motivação principalmente por eles estarem utilizando seus celulares durante a atividade, para a coleta das informações necessárias e não para uso em redes sociais, por exemplo. Percebeu-se também um grande envolvimento durante todo o tempo de aplicação da atividade. Isso mostra que uma técnica de AA proporciona dinamismo, curiosidade envolvimento e satisfação ao final de sua aplicação.

O questionário utilizado no trabalho de Castro e Souza [2016], com uma pequena modificação na questão 1, foi aplicado com o objetivo de medir o nível de aceitação da técnica. O mesmo foi composto de 25 questões que obedeciam a escala: 1 (ruim), 2 (razoável), 3 (bom), 4 (muito bom) e 5 (excelente). Na turma de 2016, 8 alunos responderam o questionário e na turma de 2017, 7 alunos. Por conta de limitação de espaço, questões do questionário foram omitidas assim como as respostas. Todas as questões e respostas podem ser acessadas no endereço: <https://goo.gl/LgnbtH>

Algumas questões foram destacadas.

i) “Estou satisfeito com a atividade apresentada porque foi uma prática diferente das aulas convencionais” mostrou que tanto os alunos de 2016 quanto os de 2017 ficaram satisfeitos com a técnica e concordam que é uma atividade diferente das atividades tradicionais usadas em sala de aula. A figura 5 ilustra os resultados, sendo à esquerda os alunos de 2016 e à direita os alunos de 2017.

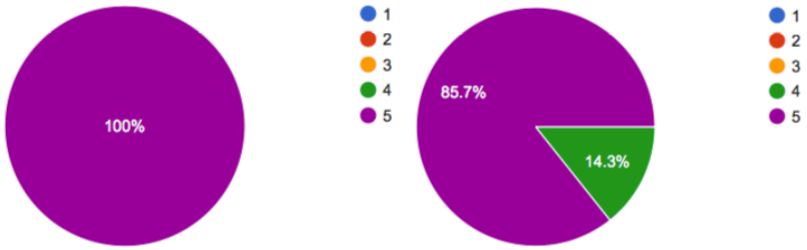


Figura 5. Respostas dos alunos da turma de 2016 e 2017, respectivamente, quanto a satisfação com a atividade diferente das usadas em aulas tradicionais

ii) “A utilização da atividade me fez ficar atento à aula ministrada”. A figura 6, esquerda mostra os alunos de 2016 com cerca de 87,5% de aceitação e, à direita os alunos de 2017 com 100% de aceitação. Percebe-se que foi detectado um *outlier* na turma de 2016 referente a 1(um) aluno que marcou a opção 1. Em análise posterior se observou tratar de um caso de questões particulares e não rejeição específica à técnica. Pode-se concluir que a técnica elaborada auxilia os alunos a ficarem mais atentos durante a aula.

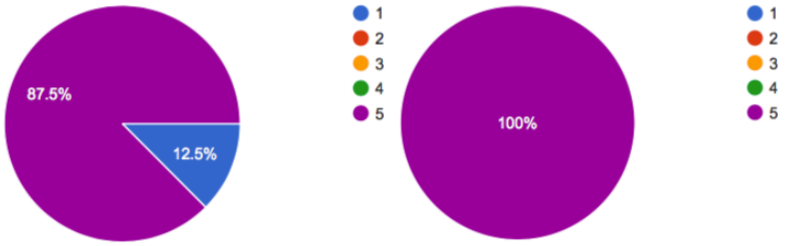


Figura 6. Respostas dos alunos da turma de 2016 e 2017 respectivamente quanto a atenção à aula ministrada

iii) “A atividade contribuiu para a minha aprendizagem na disciplina” mostra que os alunos também concordam que a atividade auxiliou sua aprendizagem na disciplina, ou seja, o uso de uma técnica AA pode contribuir para o aprendizado. A figura 7 ilustra os resultados obtidos (a direita os alunos de 2016 e a esquerda os alunos de 2017). Aqui também pode ser visualizado novamente o *outlier* na turma de 2016.

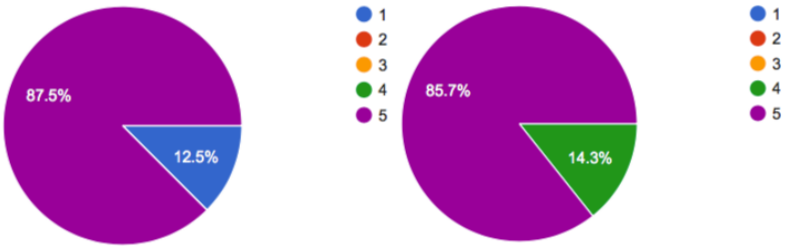


Figura 7. Respostas dos alunos da turma de 2016 e 2017 respectivamente quanto a contribuição da técnica para sua aprendizagem

5. Conclusão

As aulas tradicionais (expositivas) tendem a não prender a atenção dos alunos. O uso da Aprendizagem Ativa (AA) pode mudar este cenário permitindo um melhor aprendizado do conteúdo lecionado nas aulas.

Este trabalho apresentou uma técnica de AA que pode ser utilizada por docentes da disciplina Banco de Dados (BD), de forma a completar o conteúdo abordado em sala de aula. Essa técnica mostrou-se eficaz em uma experiência com alunos de 4º período de um curso de Bacharelado em Sistemas de Informação nos anos 2016 e 2017. A técnica não é limitada somente ao uso na disciplina de BD, podendo também ser aplicada em outras disciplinas como, por exemplo, Análise Orientada a Objetos.

Deste modo, conclui-se que uma atividade didático-pedagógica diferenciada, ou mais especificamente uma atividade de AA, pode apoiar ensino-aprendizagem dos alunos, além de mantê-los motivados durante as aulas.

Referências

Acharya, S., Manohar, P. A., Wu, P., Schilling, W., Ansari, A. (2015). Integrated Active Learning Tools for Enhanced Pedagogy in a Software Engineering Course. Computers in Education Journal, 7(2), 17-28.

Acosta, O., Reategui, E., Behar, P. A. (2016). Recomendação de Conteúdo em um Ambiente Colaborativo de Aprendizagem Baseada em Projetos. Congresso Brasileiro de Informática e Educação – CBIE. Uberlândia.

Allen, D., Tanner, K. (2005). Infusing active learning into the large-enrollment Biology class: seven strategies, from the simple to complex. Cell Biol. Educ. 4(4), 262–268

Araújo, R. A., Santos, R.A.A., Farias, R. S., Franca, R. S., Silva, T. S. C., Vasconcelos, R. L., Tedesco, P., Padilha, M. A., Belian, R. B. (2015) Investigação sobre Inovações Pedagógicas Protagonizadas por Docentes em uma Instituição e Ensino Universitário no Brasil. In: CINDU2015 – IV Congresso Internacional de Docência Universitária. Vigo. Anais do CINDU 2015.

Barkley, E.: Student Engagement Techniques. Jossey-Bass, San Francisco (2010)

Behar, P. A. (2011). Constructing Pedagogical Models for E-Learning. International Journal of Advanced Corporate Learning, 4(3), 16–22.

Bonwell, C. C., Eison, J. A. (1991). Active learning: Creating excitement in the classroom. Association for the Study of Higher Education, Washington, DC.

Carvalho, M. J. S., Nevado, R. A., Menezes, C. S.. (2005). Arquiteturas Pedagógicas para Educação a Distância: Concepções e Suporte Telemático. SBIE - Simpósio Brasileiro de Informática na Educação, 351–360.

Castro, R. M., Souza, G. S. (2016). O Uso de Recursos Lúdicos para o Ensino de Processos em Engenharia de Software. 24º WEI - Workshop sobre Educação em Computação, XXXVI Congresso da Sociedade Brasileira de Computação, Porto Alegre.

Castro, R. M. et al. (2016). AGILITY SCRUM – Um Jogo para Ensino da Metodologia SCRUM. 25º WEI - Workshop sobre Educação em Computação, XXXVI Congresso da Sociedade Brasileira de Computação, São Paulo.

Castro, R. M., Siqueira, S. W. M. (2017). Aprendizagem Ativa em Sistemas de Informação: Novas Técnicas Propostas e Reflexões sobre as Experiências. 13º SBSI - Simpósio Brasileiro de Sistemas de Informação, Lavras - MG.

Dong, J. J., Hwang, W. Y., Shadiev, R., Chen, G. Y. (2017). Pausing the classroom lecture: The use of clickers to facilitate student engagement. Active Learning in Higher Education, 18(2), 157-172.

Drake, J. R. (2012). A critical analysis of active learning and an alternative pedagogical framework for introductory information systems courses. Journal of Information Technology Education: Innovations in Practice, 11, 39-52.

Gleason, B. L. et al. (2011). An active-learning strategies primer for achieving ability-based educational outcomes. Am. J. Pharm. Educ. 75(9), 186.

Linnenbrink, E.A., Pintrich, P.R. (2004). Role of affect in cognitive processing in academic contexts. In: Dai, D., Sternburg, R. (eds.)Motivation, Emotion, and Cognition, pp. 57–87. Lawrence Erlbaum Associates, Mahwah.

Massey, A. P., Brown, S. A., Johnston, J. D. (2005). It's All Fun and Games. Until Students Learn. Journal of Information Systems Education, 16 (1).

Meyers, C., Jones, T. (1993). Promoting active learning: Strategies for the college classroom. San Francisco, CA: Jossey- Bass Publishers.

Mitchell, A., Petter, S., Harris, A. (2017). Learning By Doing: Twenty Successful Active Learning Exercises for Information Systems Courses. Journal of Information Technology Eduaction: Innovations in Practice, 16(3), 21–46.

Piaget, J. (1977). The Role of Action in the Development of Thinking. In W. F. Overton, & J. M. Gallagher, Advances in Research and Theory. New York: Plenum Press.

Ramiller, N. C. (2002). The Virtual Interactive Project: Teaching Analysis and Design Through Narrative and Drama. Communications of the Association for Information Systems, 9(1).

Thomas, D., Brown, J. S. (2011). A new culture of learning: Cultivating the imagination for a world of constant change: CreateSpace Independent Publishing.

Thongmak, M. (2017). Flipping MIS Classroom by Peers: Gateway to Student's Engagement Intention. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 387-396). International World Wide Web Conferences Steering Committee.

Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores

Carolina Moreira[1], **Andrey Ricardo Pimentel**[1], **Eleandro Maschio**[2]

[1]Programa de Pós-Graduação em Informática
Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

[2]Coordenação do Curso de Tecnologia em Sistemas para Internet
Universidade Tecnológica Federal do Paraná (UTFPR)
Guarapuava – PR – Brasil

caroll-m@hotmail.com, andrey@inf.ufpr.br, eleandrom@utfpr.edu.br

Resumo. *A pesquisa detalha uma abordagem para tratar do sequenciamento inteligente e adaptativo de enunciados, considerando a modelagem dinâmica do aprendiz, no domínio de Programação de Computadores. Propõe-se um processo de ordenação heurística que indica capacidades a serem desenvolvidas pelo aprendiz, bem como o sequenciamento adaptativo de enunciados que contribuam nessa evolução. Com isso, aspectos de sobreposição destacam o progresso do aprendiz frente ao conhecimento do domínio, além da contribuição individual de cada enunciado. A análise dos resultados observou a aderência didática da abordagem, tendo em vista a comparação das sugestões de enunciados fornecidas pelo processo de sequenciamento com outras dadas por tutores humanos.*

Abstract. *This research details an approach to deal with intelligent and adaptive exercise ordering, considering the student dynamic modeling in Computer Programming domain. Then is proposed a heuristic ordering process that indicates capabilities to be developed by the student, as also an adaptive exercise ordering that contributes in this evolution. Thus, overlaying aspects highlight the progress of the learner compared to the domain knowledge, and to the individual contribution of each exercise. The analysis of the results observed the didactic approach adherence, when comparing the exercises suggestions provided by the ordering process with others given by human tutors.*

1. Introdução

O ensino em Programação de Computadores tem sido um desafio por décadas [Norvig 2001]. Embora se trate de uma disciplina fundamental para qualquer curso da área de Ciência da Computação, o aprendizado dela mostra-se difícil para muitos iniciantes. Logo, é importante encontrar meios de diminuir o impacto nesses níveis, bem como atenuar os altos índices de reprovações e de desistências nos respectivos cursos.

Dentre as várias abordagens para apoiar o processo de ensino, destacaram-se os Sistemas Tutores inteligentes (STIs). Tratam-se de sistemas projetados para associar técnicas de Inteligência Artificial, de maneira que saibam o que ensinar, a quem ensinar e

como ensinar [Nwana 1990]. Assim, objetivam maior aproximação das aulas individuais, considerando características particulares do aprendiz para adaptar a interação.

Nesse sentido, a modelagem do aprendiz responsabiliza-se por armazenar o estágio atual de conhecimento e o desempenho do indivíduo sobre os tópicos tutorados. A partir dela, o sistema consegue definir o nível de dificuldade relativo de um conteúdo frente ao conhecimento do aprendiz, além de propor estratégias de tutoria personalizadas.

Paralelamente, qualquer ambiente de ensino que utilize enunciados precisa, de alguma maneira, determinar a sequência em que eles serão oferecidos ao aprendiz. Alguns sistemas apresentam os enunciados em ordem predefinida. Outros, permitem que os aprendizes selecionem o próximo enunciado a partir de uma lista estática de alternativas [Weber and Brusilovsky 2001]. Em ambas as situações, a escolha do enunciado não considera as capacidades atuais do aprendiz.

Considerando a modelagem do aprendiz, um STI tem a capacidade de sugerir enunciados próprios ao nível de conhecimento do indivíduo. Isso se mostra positivo, pois enunciados com dificuldade muito abaixo desse nível podem causar entediamento, e do contrário, sendo muito acima, provocar desmotivação [Pimentel and Direne 1998]. Ambos os casos são passíveis do abandono da atividade proposta.

Dessa forma, existe espaço de contribuição para abordagens de sequenciamento adaptativas e inteligentes de enunciados, buscando auxiliar diferentes perfis de aprendizes, independentemente dos níveis de conhecimento. Com isso, pode-se diversificar a ordem e mesmo a quantidade de enunciados e potencializar o processo de ensino-aprendizagem para cada indivíduo em particular.

2. Objetivo e Contribuições

O objetivo geral desta pesquisa foi propor uma abordagem para o sequenciamento adaptativo de enunciados, frente a uma modelagem dinâmica do aprendiz, no contexto de Programação de Computadores. A metodologia de sequenciamento proposta foi implementada no protótipo de ferramenta denominado NextStep, que adota grafos genéticos [Maschio and Direne 2015, Goldstein 1979] como base para representação interna e externa (ou seja, armazenamento e apresentação) dos modelos. Assim, aspectos de sobreposição no grafo destacam o progresso do aprendiz frente ao conhecimento do domínio, além da contribuição individual de cada enunciado. Logo, no contexto do grafo genético, a sobreposição enfatiza que a estrutura correspondente ao Modelo do Aprendiz deriva do conhecimento do domínio, sendo um subgrafo (ou subconjunto) dele.

Partindo dos modelos do aprendiz e do domínio, proveu-se um processo de ordenação heurística que orienta conteúdos a serem explorados pelo aprendiz. Indicam-se perícias (vértices do grafo) que deveriam ser prioritariamente desenvolvidas (compreendidas e exercitadas) pelo indivíduo, conforme o seu nível atual de conhecimento. Depois, são sugeridos enunciados que contemplem o desenvolvimento dessas perícias.

A abordagem denota como **enunciados**, de maneira genérica, as contribuições dentro de um determinado domínio de conhecimento, sejam elas conteúdos, explicações, exemplos, exercícios, entre outras. O protótipo, por sua vez, considera especificamente exercícios que devam ser implementados em uma linguagem de programação, tais como os problemas utilizados nas maratonas da área.

Alguns passos metodológicos, nesse sentido, foram antecipados pela pesquisa de [Maschio and Direne 2015]. O mesmo estudo desenvolveu duas ferramentas que atuam nas primeiras etapas do processo. A primeira ferramenta se concentra na descrição do conhecimento do domínio por meio do detalhamento das perícias componentes. A segunda destina-se à catalogação de enunciados descritos como subgrafos do conhecimento de domínio formalizado pela primeira. A metodologia proposta pela presente pesquisa utiliza os recursos providos por essas duas ferramentas, para realizar o sequenciamento adaptativo de enunciados, então implementado em uma terceira [Moreira 2016].

3. Resenha Literária

O referencial teórico da pesquisa alicerça-se em três temas principais, detalhados nas seções seguintes: modelo de sobreposição e grafos genéticos (3.1), aquisição de conhecimento em Programação de Computadores (3.2) e sequenciamento de enunciados (3.3).

3.1. Modelo de Sobreposição e Grafos Genéticos

O Modelo do Aprendiz, no contexto de um STI, remete à competência do aprendiz no conteúdo tutorado, podendo incluir suas preferências relativas ao processo de aquisição de conhecimento. Esse modelo, portanto, contém uma representação do estado atual de conhecimento e do desempenho do aprendiz sobre o domínio ensinado [Giraffa 2003]. Ele é atualizado de forma dinâmica pelo sistema e representa informações de aprendizes em particular. Todavia, o modelo pode ser adaptado a necessidades específicas, representando informações coletivas ao invés de individuais, ou ainda por tempo fixado em detrimento de um histórico completo.

Além disso, o Modelo do Aprendiz pode variar na maneira com que representa internamente as informações obtidas, sendo o Modelo de Sobreposição (*Overlay*) uma delas. Ele representa o conhecimento do aprendiz como um subconjunto do conhecimento do domínio. A sobreposição desses conjuntos evidencia os conhecimentos que devem ser apropriados pelo aprendiz. Essa abordagem implica que a representação de ambos os modelos (do domínio e do aprendiz) sejam comparáveis.

A pesquisa de [Maschio and Direne 2015] buscou uma técnica de representação que pudesse descrever as capacidades do domínio de Programação de Computadores e que, frente ao Modelo de Sobreposição, denotasse a incidência do conhecimento do aprendiz sobre o conhecimento do domínio. O citado estudo optou pelo Grafo Genético [Goldstein 1979], que se trata de um tipo específico de grafo cujo conhecimento do domínio (ou de um experto) é descrito por um conjunto de capacidades (vértices) interconectados por relações (arestas) que evoluem: da simplificação à elaboração, do desvio à correção, da abstração ao refinamento, da especialização à generalização e do pré-requisito ao pós-requisito.

3.2. Aquisição de Conhecimento em Programação de Computadores

O processo de aprendizagem em domínios de natureza prática e complexa fundamenta-se na aquisição de conhecimentos sobre **princípios** e consequente desenvolvimento deles até que constituam **perícias** na área [Lesgold et al. 1988]. A **aquisição de princípios** corresponde à assimilação, pelo aprendiz, dos fundamentos de um domínio específico, ou seja, do conhecimento formal inicialmente repassado. O **desenvolvimento de perícias**

acaba sendo o processo de construção de conhecimento por meio da experiência, levando ao crescimento da aptidão pela prática. Esse processo envolve a integração dos princípios aprendidos com a experiência que se alcançou até o momento.

Diante da mesma perspectiva de [Lesgold et al. 1988], conforme [Maschio 2013], a aprendizagem em Programação de Computadores pode ser entendida como a aquisição de conhecimentos sobre os princípios de lógica de programação e consequente desenvolvimento deles até consolidarem perícias na área. Os princípios envolvem a compreensão das instruções isoladas, considerando a rigidez léxica, sintática e semântica da linguagem utilizada. As perícias se tratam do encadeamento e aninhamento dessas instruções na construção de um algoritmo. Portanto, a aplicação de ambas as categorias de conhecimento são exigidas no ato de programar.

Nesse sentido, [Pimentel and Direne 1998] identificou o seguinte conjunto de capacidades presentes no estereótipo de um programador perito: **(1)** precisão sintática; **(2)** precisão semântica; **(3)** identificação de estruturas principais no programa fonte (busca por palavra chave); **(4)** simulação mental dos estados do computador durante a execução; **(5)** catálogo de erros; **(6)** mapeamento mental das estruturas do programa; **(7)** checagem de pré-condições; **(8)** análise do problema; **(9)** integração dos subproblemas; **(10)** generalização da solução; **(11)** reutilização de soluções já conhecidas; e **(12)** catálogo de soluções. Em continuidade, a pesquisa de [Maschio 2013] atuou na revisão desse conjunto de capacidades, com o objetivo de utilizá-lo para catalogar enunciados de programação.

O estudo constatou a necessidade de definir perícias de mais baixo nível de abstração que, por sua vez, correspondessem à prática em Programação de Computadores. Isso foi justificado pela dificuldade de relacionar enunciados com as perícias isoladas do conjunto definido por [Pimentel and Direne 1998]. Consequentemente, [Maschio 2013] reconheceu que esse conjunto de capacidades se caracterizava pelo alto nível de abstração e pela sobrejacência a outras perícias mais próximas dos elementos instrucionais da programação. Entretanto, eram justamente essas últimas perícias que se faziam relevantes à modelagem e evolução do aprendiz frente ao conhecimento do domínio, bem como à catalogação e ao sequenciamento de enunciados.

A abordagem para identificar esse segundo conjunto de perícias foi reconhecer frações mais restritas de habilidades em correspondência às instruções e princípios de programação. Assim, uma primeira tentativa identificou 335 perícias, hierarquizadas em cinco níveis, considerando apenas um subconjunto de conhecimento do domínio que chegava até o conteúdo de Estruturas de Repetição. Com o objetivo de facilitar o processo de catalogação dos enunciados, o conjunto de perícias foi sintetizado em termos daquelas pertencentes aos primeiros níveis hierárquicos. Esse conjunto foi modelado por meio de um grafo genético composto por 41 perícias e 60 relações, sendo apontado pelo pesquisador como uma versão experimental e sujeita a alterações pela comunidade científica.

3.3. Sequenciamento de Enunciados

Contribuições anteriores evidenciaram a complexidade dos enunciados propostos como um dos principais componentes de motivação do aprendiz [Soldado and du Boulay 1995]. Equivalentemente, [Pimentel and Direne 1998] sustenta que "a repetição sistemática de enunciados completamente diferentes, porém de graus de complexidade aproximadamente iguais pode elevar consideravelmente a autoconfiança do aprendiz, mantendo-o

motivado e produtivo". Em contrapartida, [Maschio 2013] apontou que a extrema rigidez na ordenação das galerias de enunciados é um fator adverso em ambientes de ensino.

No ensino de Programação de Computadores, [Pimentel and Direne 1998] descreve **medidas cognitivas**, diante da perspectiva do desenvolvimento de perícias, que se mostram úteis no sequenciamento automático e adaptativo de enunciados. Elas possuem a função de quantificar cognitivamente um enunciado, sendo responsáveis por estimar quanto ele exige de um aprendiz em termos de conhecimentos adquiridos e de capacidades desenvolvidas na progressão do aprendizado. As medidas cognitivas consideram subcomponentes relacionados à complexidade do software (e.g. linhas de código, complexidade estrutural e detalhes de implementação), somados ao conjunto de capacidades identificadas no estereótipo de um programador perito (Seção 3.2).

Nesse mesmo sentido, a **carga cognitiva** de um enunciado é definida como a capacidade que ele tem de exercitar o aprendiz na construção de um programa de computador, contribuindo para o desenvolvimento de perícia na área. Ela consiste na ponderação das medidas cognitivas presentes em um enunciado. Essa ponderação é necessária porque diferentes medidas possuem níveis distintos de contribuição para cada enunciado. Os pesos são determinados por meio de informações obtidas de especialistas no domínio.

Assim, as medidas cognitivas conseguem amparar a escolha do próximo enunciado em uma sessão de ensino. Elas auxiliam na classificação de enunciados por grau de exigência/contribuição, podendo adaptar a sequência deles ao nível atual do aprendiz. A mesma pesquisa implementou isso por meio da ferramenta **Sequence**, que avalia os enunciados quanto à complexidade do software e considera as perícias de precisão sintática e semântica, análise do problema, reutilização de soluções e simulação mental.

Conforme antedito, os estudos de [Pimentel and Direne 1998] foram retomados por [Maschio 2013]. A pesquisa revisou as capacidades do estereótipo de um programador experto e modelou essas perícias em um grafo genético (seções 3.2 e 3.1). Na abordagem, tanto o conhecimento do aprendiz quanto a contribuição de cada enunciado constituem subgrafos do conhecimento do domínio. A referida pesquisa implementou duas ferramentas, uma destinada à **descrição do conhecimento do domínio** e outra que se concentra na **catalogação e elicitação de enunciados**. Aspectos de modelagem do aprendiz não atingiram o estágio de implementação.

Por fim, em outros sistemas recentemente divulgados, se observa que são escassas as publicações que forneçam detalhes dos mecanismos que apoiam a escolha do próximo enunciado. Abaixo são relacionados outros três sistemas tutores que promovem o sequenciamento de enunciados e suas respectivas contribuições:

- **JV^{2}M:** destina-se ao ensino do processo de compilação em linguagens orientadas a objetos por meio da metáfora de uma aventura em 3D. Ele determina que o melhor momento para intervir é quando o aprendiz comete um erro, pois se consegue utilizar a própria falha como oportunidade focada de explicação. O próximo exercício é apresentado quando se detecta que o aprendiz tem conhecimento suficiente para avançar ao próximo nível [Gómez-Martín et al. 2005];
- **SQL-Tutor:** aborda o ensino da linguagem de pesquisa SQL. Ele comparou duas estratégias para a seleção de enunciados, sendo que a primeira utiliza a complexidade estática especificada no momento da autoria, enquanto a outra considera

medidas dinâmicas de dificuldade que são calculadas individualmente para cada aprendiz. O estudo indicou que a segunda estratégia trouxe maiores benefícios ao desempenho dos aprendizes no processo de ensino [Mitrovic and Martin 2004];

- **PHP ITS:** promove o ensino de desenvolvimento web básico para iniciantes. Ele considera que o exercício mais adequado ao indivíduo, em um determinado momento, é aquele que abrange o menor número de conteúdos ainda não aprendidos [Weragama 2013].

4. Fundamentos da Solução Proposta

Primeiramente, são discutidos os conceitos de automaticidade, adaptatividade e inteligência no escopo do problema (4.1). Na sequência, propõe-se uma metodologia inteligente e adaptativa para sequenciamento de enunciados, baseada em grafos genéticos (4.2).

4.1. Sequenciamento Automático, Adaptativo e Inteligente de Enunciados

A seguir, são prescritos os conceitos de automaticidade, adaptatividade e inteligência assumidos pelos autores da pesquisa no escopo do sequenciamento de enunciados. Foram consideradas as perspectivas das áreas de Inteligência Artificial [Silva 2015, Machado et al. 2015] e Interação Humano-Computador [Ahmad et al. 2004], ambas aplicadas à Educação. Por meio de exemplos, destacam-se os contrastes entre esses conceitos, indicando que possuem peculiaridades e que não devem ser considerados sinônimos.

Sequenciamento Automático Simples

Automatiza a geração de enunciados sem implicar, invariavelmente, que esse processo seja adaptativo ou inteligente (isto é, usar técnicas da IA). Como exemplo, pode-se gerar uma sequência de 10 enunciados: **(1)** de maneira aleatória; **(2)** sendo 4 fáceis, 4 médios e 2 difíceis, conforme categorização prévia pelo instrutor; ou **(3)** diferente de outra anteriormente proposta.

Sequenciamento Adaptativo

Adapta a sequência de enunciados segundo um conjunto de critérios, não necessariamente inteligentes. São exemplos: **(1)** excluir da lista de enunciados candidatos aqueles que abrangem o mesmo tópico e têm dificuldade muito próxima de exercícios recém-resolvidos pelo aprendiz; **(2)** propor um enunciado de dificuldade levemente menor, após a tentativa malsucedida de resolução de determinado exercício pelo aprendiz; e **(3)** reduzir ou aumentar a quantidade de enunciados da sequência, em resposta ao desempenho do aprendiz.

Sequenciamento Inteligente

Promove o sequenciamento de enunciados baseado em critérios inteligentes. Muito dessa inteligência se estabelece sobre a precisão da modelagem do aprendiz frente ao conhecimento do domínio. Com isso, pode-se variar radicalmente tanto a ordem e quanto a quantidade de enunciados fornecidos, considerando, por exemplo: **(1)** a dificuldade relativa de um enunciado frente à modelagem do aprendiz; **(2)** características individuais de aprendizagem capturadas na interação; ou **(3)** fatores coletivos dos demais aprendizes do grupo, que seriam ponderados com os individuais.

Nesses termos, uma estratégia de sequenciamento pode ser automática, mas não adaptativa. Ou ainda, ser adaptativa sem que seja inteligente.

4.2. Metodologia para Sequenciamento de Enunciados

Os seguintes passos metodológicos foram estabelecidos, cada qual detalhado na continuidade: **(1)** Modelagem do aprendiz; **(2)** Sugestão do tópico tutorado/exercitado; **(3)** Sequenciamento de enunciados; **(4)** Atualização do Modelo do Aprendiz; **(5)** Acompanhamento do processo de aprendizagem; **(6)** Implementação de um protótipo de ferramenta.

Basicamente, por meio da modelagem dinâmica do aprendiz, uma busca heurística recomenda conhecimentos que devem ser desenvolvidos e enunciados que contemplem essa demanda. Depois, incide-se sobre a alimentação do Modelo do Aprendiz, conforme seu desempenho na resolução dos enunciados propostos. O processo, como um todo, é cíclico e permite o acompanhamento pela representação do grafo genético na interface.

O processo depende de que o conhecimento do domínio esteja formalizado em um grafo genético e que os enunciados tenham sido elicitados nesses termos. Tais aspectos metodológicos anteriores foram cumpridos pela pesquisa de [Maschio and Direne 2015]. Em se tratando de implementação, o mesmo estudo desenvolveu duas ferramentas.

A primeira ferramenta se concentra na **descrição do conhecimento do domínio** por meio do detalhamento das perícias componentes. A própria interação com a ferramenta ocorre por meio da representação visual do grafo genético e *menus* de contexto. Permite-se: **(a)** a inserção, edição e remoção de perícias; **(b)** o estabelecimento das relações evolucionárias entre perícias; **(c)** a definição de uma perícia inicial; e **(d)** a validação do modelo descrito quanto à alcançabilidade das demais perícias a partir daquela inicial.

A segunda ferramenta, usando a mesma representação visual, destina-se à **catalogação e elicitação de enunciados** descritos como subgrafos do conhecimento de domínio formalizado pela primeira. Ela possibilita que sejam indicadas as perícias que cada enunciado contempla. O aspecto de sobreposição do enunciado, no conhecimento do domínio, destaca a região abrangida do grafo. A ferramenta também permite inspecionar a quantidade de enunciados que exercitam determinada perícia e a existência de perícias não abordadas pelo conjunto de enunciados. Além disso, ela provê uma análise mais profunda sobre a coesão do catálogo e consequente requisição da autoria de enunciados adicionais.

4.2.1. Modelagem do Aprendiz

Admitindo a modelagem em grafo genético das perícias no domínio de Programação de Computadores, o conhecimento de um aprendiz é considerado como o subconjunto (ou subgrafo) do conhecimento de um experto. Assim, as perícias (vértices) faltantes ao aprendiz são destacadas pela sobreposição do segundo grafo no primeiro. A Figura 1 exemplifica o estágio atual do aprendiz frente ao conhecimento do domínio.

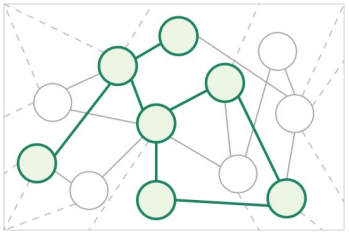


Figura 1. Sobreposição do Modelo do Aprendiz frente ao conhecimento do domínio

A sobreposição propriamente dita é evidenciada. Assume-se a modelagem não apenas das perícias desenvolvidas pelo aprendiz, mas também da estrutura complementar que fornece indícios de como ele aprendeu (arestas). Apesar do armazenamento dessas informações ser trivial, existe dificuldade em inferir tais elementos por meio da interação com aprendiz.

4.2.2. Sugestão do Tópico Tutorado/Exercitado

Partindo dos modelos do aprendiz e do domínio, provê-se um processo de busca heurística que orienta conteúdos a serem explorados pelo aprendiz. Ele indica perícias que deveriam ser prioritariamente desenvolvidas (compreendidas e exercitadas) pelo indivíduo, conforme o seu nível atual de conhecimento (Figura 2). A abordagem considera como critérios: **(a)** priorização absoluta da perícia inicial; **(b)** priorização de perícias insuficientemente desenvolvidas; **(c)** perícias localizadas na fronteira entre o subgrafo do Modelo do Aprendiz e o restante do conhecimento do domínio; **(d)** distância das perícias desse subconjunto até aquela definida como inicial para o conhecimento do domínio; **(e)** estratégias de tutoria bem-sucedidas para aquele aprendiz; e **(f)** navegabilidade proporcionada para que o aprendiz escolha entre o conjunto de perícias sugeridas.

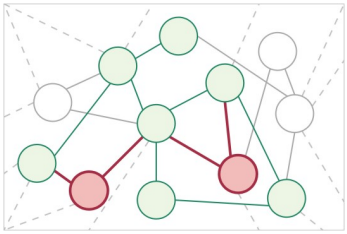


Figura 2. Indicação de regiões do grafo a serem exploradas

4.2.3. Sequenciamento de Enunciados

Escolhida a perícia (ou conjunto delas) como tópico a ser exercitado, encontram-se enunciados que correspondam a essa demanda. Consegue-se isso porque o mesmo grafo genético (que representa o conhecimento do domínio) é empregado como gabarito para os enunciados a serem propostos. Nessa perspectiva, cada enunciado contribui para que se desenvolvam perícias específicas do aprendiz, sendo também um subgrafo do conhecimento do domínio (Figura 3).

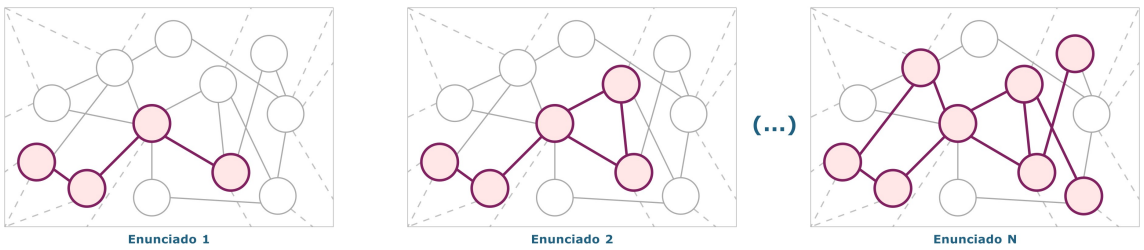


Figura 3. Catálogo de enunciados em sobreposição ao conhecimento do domínio

Diante dessa possibilidade, o conjunto de enunciados selecionados consegue ser catalogado de acordo com o referido gabarito. Ocorre, então, a adaptação inteligente desses enunciados de acordo com o perfil atual do aprendiz. Com isso, diferentes exercícios

podem ser sugeridos para a tutoria de uma mesma perícia, considerando aprendizes distintos. Adotou-se a abordagem de priorizar os enunciados que abranjam o menor número de perícias não desenvolvidas, isto é, que sejam mais próximos do conhecimento do aprendiz naquela ocasião. Caso dois enunciados se igualem nesse quesito, prioriza-se aquele que exercita o menor número de perícias ao todo (mesmo que já tenham sido desenvolvidas). A abordagem se baseia na Zona de Desenvolvimento Proximal de Vygotsky.

4.2.4. Atualização do Modelo do Aprendiz

Após o cumprimento do enunciado, ocorre uma avaliação para verificar se houve o desenvolvimento das perícias abordadas. A alimentação do Modelo do Aprendiz é indispensável para que se passe a refletir o nível atual de conhecimento do indivíduo, considerando o possível progresso. Entretanto, apenas a sugestão do enunciado para o aprendiz não é condição suficiente para se inferir que as perícias foram desenvolvidas. Foram consideradas alternativas que forneçam parâmetros para que o Modelo do Aprendiz seja atualizado, permitindo que a próxima sugestão de enunciado já considere essas mudanças.

A primeira consiste na avaliação das resoluções por um especialista externo (humano), levando em conta o gabarito de perícias fornecido pelo enunciado. A Figura 4 registra o progresso do aprendiz e consequente evolução na estrutura do grafo genético.

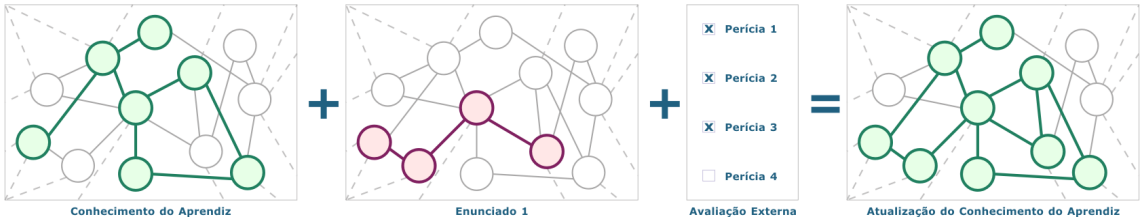


Figura 4. Atualização do Modelo do Aprendiz

Outra alternativa resume-se em substituir o especialista externo por um diagnóstico automático simples, a exemplo do que ocorre nas maratonas de programação. Contudo, para a atualização do Modelo do Aprendiz, avaliar uma solução vai muito além de observar se ela fornece saídas corretas. Interessa saber quais foram as capacidades desenvolvidas, ou seja, o aprendizado que se obteve com o enunciado sugerido. Mesmo uma solução com comportamento incorreto pode ter trazido benefícios ao aprendizado, pois é possível que, ainda assim, parte das perícias abordadas tenham sido desenvolvidas.

4.2.5. Acompanhamento do Processo de Aprendizagem

A representação em grafo abre oportunidades de extrair dos modelos informações úteis à didática do instrutor. A pesquisa suporta um conjunto dessas possibilidades, considerando: **(a)** a sintetização do Modelo do Aprendiz de um conjunto, como uma sala de aula, para que se observe como as capacidades são ensinadas pelo instrutor; **(b)** a apresentação de um histórico do desenvolvimento de perícias do aprendiz durante um intervalo de tempo específico (Figura 5), exibindo o grafo sendo conquistado (coloração dos vértices) à proporção desse tempo; e **(c)** a aplicação desse mesmo recurso para sintetizar o Modelo do Aprendiz de um conjunto, auxiliando a visualizar a progressão do aprendizado de uma sala inteira, ou de um curso inteiro.

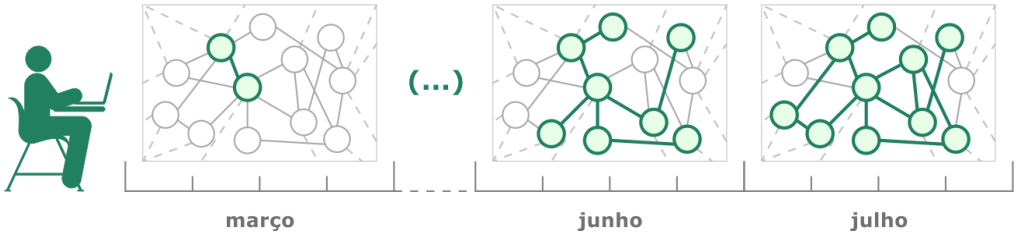


Figura 5. Progresso do aprendiz ao longo do tempo

5. Dinâmica de Funcionamento da Solução Proposta

O cenário de utilização simula e acompanha o processo de sequenciamento de enunciados para um aprendiz. O processo é divido nas etapas de: **(1)** situar o aprendiz; **(2)** sugerir perícia; **(3)** sugerir enunciado; e **(4)** avaliar solução. No exemplo, foi suposta a aplicação do processo para um aprendiz que teve apenas uma introdução teórica aos primeiros tópicos da área. A explicação tem caráter resumido, enfocando a fluência desse processo.

O Modelo do Aprendiz é um subconjunto nulo do conhecimento do domínio, pois nenhuma perícia do grafo genético foi desenvolvida ainda. A ferramenta, em primeiro momento, situa o aprendiz frente ao conhecimento do domínio. Faz-se isso como amparo metacognitivo, por meio de representações externas, de caráter visual (sobreposição no grafo genético) e textual (painel de estatísticas), como pode ser observado na Figura 6.

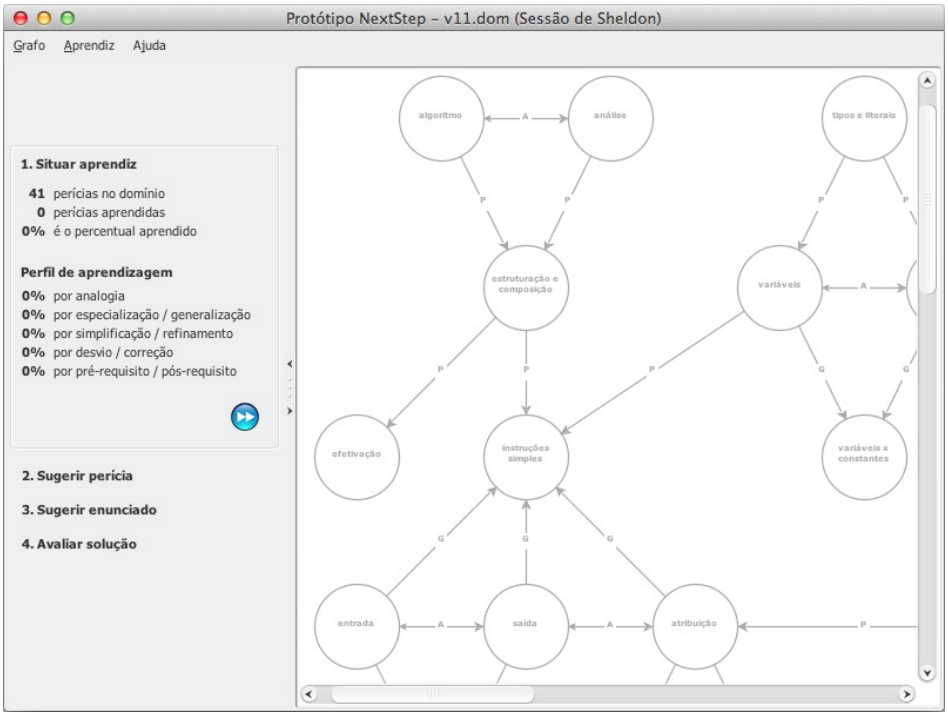


Figura 6. Protótipo NextStep: Situar aprendiz

A ferramenta sugere exclusivamente a perícia inicial do domínio, que possui o identificador *algoritmo*, para ser desenvolvida na sessão de sequenciamento. Admitindo a perícia como prioritária a ser exercitada, cabe à ferramenta encontrar enunciados que contribuam para esse desenvolvimento. De todo o catálogo, apenas os enunciados *Olá*

mundo, *Olá usuário* e *Metade de um real* contribuem para a apropriação da perícia *algoritmo*. Convém destacar que, embora todos os enunciados de um catálogo envolvam, por exemplo, instruções de saída, nem todos se propõem a desenvolver esse conhecimento.

Como se tem mais de um enunciado que contempla a perícia sugerida, oferece-se navegabilidade ao aprendiz, para que ele possa decidir entre as opções apresentadas. A escolha é orientada pela ordem das sugestões, priorizando os enunciados que abranjam o menor número de perícias não desenvolvidas e, depois, aqueles que exercitam o menor número de perícias ao todo (mesmo que já tenham sido desenvolvidas). Essas duas informações são apresentadas ao lado do identificador de cada enunciado. Adicionalmente, quando se seleciona um enunciado, as perícias que ele objetiva desenvolver (regiões) são destacadas no conhecimento do domínio (Figura 7).

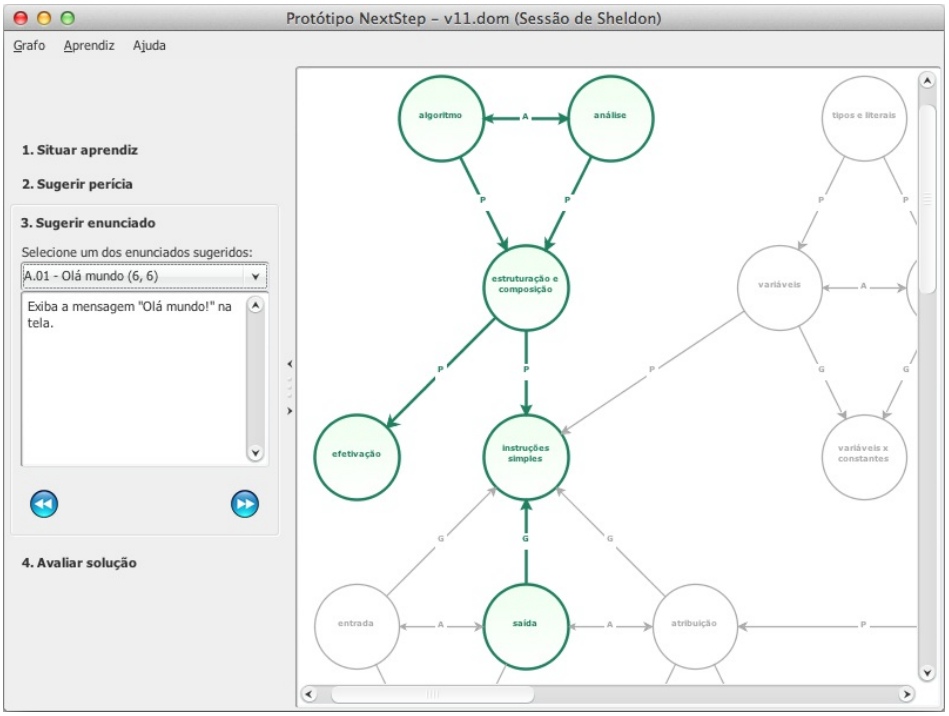


Figura 7. Protótipo NextStep: Sugerir enunciado

Então, supondo-se que o primeiro enunciado (*Olá mundo*) tenha sido a escolha do aprendiz para exercitar a perícia *algoritmo* e que a resolução seja avaliada como correta na ferramenta, o Modelo do Aprendiz é atualizado de modo que passe a refletir que as perícias exercitadas foram desenvolvidas. Exibe-se um sumário dessa atualização, indicando quais outras perícias foram desenvolvidas no processo e que o ciclo de sequenciamento foi completado.

Considerando o Modelo do Aprendiz atualizado, cabe à ordenação heurística da ferramenta sugerir a próxima perícia a ser desenvolvida e, depois, um enunciado que a atenda. A ferramenta prossegue avaliando a resolução do aprendiz, atualizando o seu modelo, sugerindo perícias a serem desenvolvidas e enunciados correspondentes. Cada ciclo de sequenciamento sempre começa situando o aprendiz sobre o estado atual dele frente ao conhecimento do domínio.

Diante do exposto, a ferramenta consiste em um **simulador** para acompanhar o

processo de sequenciamento de enunciados, fundamentado na metodologia inteligente e adaptativa proposta. O foco primário incide em observar a solução e analisá-la em funcionamento, não na fluência da interação com possíveis atores do cenário.

Atualmente, a ferramenta admite que todas as perícias exercitadas foram desenvolvidas, desde que a resolução do aprendiz seja determinada como correta no processo de sequenciamento. Aspectos de granularidade, ou seja, o desenvolvimento parcial e progressivo de perícias é amparado pela abordagem mas não foi contemplado pela versão descrita do protótipo. O detalhamento de toda metodologia, além de outros ciclos que exemplificam dinâmica de funcionamento, pode ser encontrado em [Moreira 2016].

6. Estudo Empírico

Foi realizado um estudo empírico que observou a aderência didática da solução proposta, tendo em vista a comparação das sugestões fornecidas pelo processo de sequenciamento com outras dadas por tutores humanos, assumindo enunciados de conteúdos iniciais. Considerou-se, especialmente, a proximidade entre os arranjos e a ocorrência de inversões severas, na ordem dos tópicos e dos enunciados sugeridos.

6.1. Sujeitos

Foram sujeitos três docentes de Ciência da Computação, ministrantes de disciplinas introdutórias de programação e que não tiveram envolvimento anterior com a pesquisa. Os sujeitos receberam orientações e expressaram concordância por meio de um Termo de Consentimento Livre e Esclarecido (TCLE).

6.2. Instrumentos

Um questionário com três questões foi utilizado como instrumento para coleta de dados dos docentes, apresentando situações para a sugestão de enunciados. Adicionalmente, os seguintes materiais foram fornecidos impressos:

1. Um texto de apresentação resumindo a pesquisa;
2. O conhecimento do domínio representado em grafo genético, plotado;
3. O conhecimento de um aprendiz, em determinado estágio, também representado em grafo genético; e
4. O catálogo de enunciados, diagramado no formato de cartões em tamanho A5. Cada cartão continha um enunciado e a região abrangida do grafo genético. O catálogo abrangia os conteúdos: instruções de entrada, de saída e de atribuição; estruturas de condicionais; e estruturas de repetição.

O questionário entregue aos docentes foi reproduzido na sequência:

Questão 1

Ordene os enunciados do catálogo na sequência didática que você considere mais conveniente. Entregue os cartões nessa ordem.

Questão 2

Considere que o aprendiz domina o conceito de algoritmo, conhece os tipos primitivos de dados, variáveis, constantes, bem como as instruções de entrada, de saída e de atribuição. Diante do catálogo de enunciados representado pelos cartões, qual seria o próximo enunciado que você recomendaria? Justifique.

Questão 3

Considere o estágio de desenvolvimento do aprendiz ilustrado pela figura anexa [grafo genético]. Diante do catálogo de enunciados representado pelos cartões, qual seria o próximo enunciado que você recomendaria? Justifique.

6.3. Procedimentos

Detalha-se, a seguir, as orientações fornecidas e o propósito de cada uma das questões componentes. Para comparação, foram realizadas simulações na ferramenta em resposta às mesmas questões perguntadas aos docentes. Como procedimento, assumiu-se sempre a primeira sugestão de perícia e de enunciado oferecida pela ferramenta.

Questão 1

Procura saber como os docentes ordenariam didaticamente o mesmo catálogo de enunciados utilizado pela pesquisa. Os cartões foram entregues embaralhados aos docentes, que ficaram livres para considerar a representação em grafo de cada enunciado.

Questão 2

Constrói um cenário do conhecimento atual do aprendiz para que o docente recomende o próximo enunciado, dentre aqueles do catálogo. Não faz referência à representação em grafo.

Questão 3

Em contraste com a questão anterior, remete-se diretamente ao grafo genético e induz que o docente decida com base nas mesmas informações assumidas pelo protótipo, dado um cenário para que um enunciado seja sugerido.

6.4. Resultados

Na observação dos resultados, foram contrastadas as sugestões dos docentes com aquelas fornecidas pela ferramenta, sendo:

Questão 1

Diante da ordenação do catálogo de enunciados pelos docentes, foi examinado se houve antecipação de algum enunciado que contemplasse conteúdo posterior. Ou ainda, se ocorreu o contrário, de um enunciado que requeresse conteúdo mais básico ser prorrogado. Esses casos não aconteceram e tanto a ordenação da ferramenta quanto a ordenação de cada docente puderam ser seccionadas em três partes, em correspondência respectiva aos conteúdos anteriormente citados. Constatou-se também que o sequenciamento dos enunciados dentro de um mesmo conteúdo é bastante sensível à didática do docente, pois há assuntos que um professor prefere postergar enquanto outro decide antecipá-los (e.g. seleção múltipla dentro de estruturas condicionais).

Questão 2

Os três docentes sugeriram um enunciado que introduzisse estruturas condicionais ao aprendiz. Dois docentes sugeriram o mesmo enunciado. As escolhas dos docentes corroboraram as ordenações de enunciados que eles próprios recomendaram na questão anterior, uma vez que a sugestão de cada um coincidiu com o primeiro enunciado que trata desse conteúdo na ordenação que eles propuseram antes. A ferramenta alcançou o mesmo comportamento dos dois docentes citados.

Questão 3

> Considerando o estágio do aprendiz frente ao conhecimento do domínio, os docentes tinham três opções de perícias a serem desenvolvidas para sugerir: constantes; o contraste entre variáveis e constantes; ou expressões aritméticas. Entretanto, o catálogo não possui enunciado que contemple perícias relacionadas a constantes. Essa situação foi propositalmente avaliada para elaborar o questionário. Dois docentes sugeriram enunciados que contemplassem expressões aritméticas, sendo que o outro professor indicou a autoria de um enunciado que introduzisse constantes. Por sua vez, a ferramenta sugere que constantes sejam aprendidas e, no próximo passo, não recomenda enunciados, indicando a necessidade de autoria. No entanto, a ferramenta permite retroceder e optar por expressões aritméticas, para então sugerir o mesmo enunciado de um dos docentes.

Em síntese, foi observado que a ferramenta expressou comportamento semelhante aos dos docentes nas situações avaliadas, obtendo resultados bastante próximos quando comparados. Destaca-se que não houve inversões de ordem significativas nas sugestões de perícias ou de enunciados feitas pela ferramenta. Além disso, foi notado que a ordem dos enunciados de um mesmo conteúdo diferiu mesmo entre os docentes sujeitos da pesquisa, entendo-se ser sensível, em certo grau, à didática de cada tutor humano em particular.

Algumas possibilidades, ainda na comparação com tutores humanos, residem na condução de dois outros experimentos distintos, um excluindo os detalhes da metodologia (tais como terminologia e representação) e outro imergindo os docentes nessas informações. Os resultados podem ser úteis para adequação dos parâmetros e aprimoramento da abordagem proposta, bem como para uma investigação mais profunda do problema.

7. Considerações Finais e Trabalhos Futuros

A presente pesquisa delineou uma abordagem para tratar do sequenciamento adaptativo de enunciados, considerando a modelagem dinâmica do aprendiz, no domínio de Programação de Computadores. A metodologia apresentada considera que o modelo do domínio, do aprendiz e o catálogo de enunciados sejam representados por grafos genéticos. Com isso, aspectos de sobreposição evidenciam o progresso do aprendiz frente ao conhecimento do domínio, além da contribuição individual de cada enunciado para que isso ocorra. Propôs-se, então, um processo de busca heurística que indica capacidades a serem prioritariamente desenvolvidas pelo aprendiz, bem como o sequenciamento adaptativo de enunciados que auxiliem nessa evolução. A atualização do Modelo do Aprendiz, conforme seu desempenho ao resolver os enunciados propostos, e a extração de informações úteis sobre a aprendizagem também são analisadas.

Esforços imediatos se concentraram na implementação desse formalismo em uma ferramenta de software, denominada NextStep. Então, se procedeu com a coleta de dados a partir do uso inicial dessa ferramenta, que indicou a eficiência da abordagem. A solução proposta obteve resultados bastante próximos daqueles fornecidos por tutores humanos.

Como principal contribuição, a pesquisa dedicou-se a dinamizar o aprendizado na área, por meio da adaptação de sequências de enunciados às particularidades dos aprendizes. Por fim, considera-se que parte das contribuições alcançadas possam se estender à aquisição de conhecimento em outros domínios de natureza prática e complexa.

Por fim, são indicados possíveis trabalhos futuros que podem ampliar a pesquisa:

1. Condução de experimentos adicionais para avaliar os benefícios da solução proposta, bem como a necessidade de possíveis adequações;
2. Avaliação e ajuste de parâmetros do processo a fim de garantir que a adaptatividade não seja tendenciosa;
3. Extensão dos critérios assumidos para a sugestão de perícias a serem tutoradas. Dois critérios que podem ser considerados são a necessidade de reforço [Leffa 2014] e o cometimento de erros pelo aprendiz [Kutzke 2015]. O primeiro observaria a demanda de reforço dos conteúdos à proporção do tempo, evitando o esquecimento do que foi aprendido. O outro contemplaria os erros cometidos pelo aprendiz, tentando inferir se provieram da ausência de conceitos, de concepções errôneas, de esquecimento, ou ainda se foram apenas deslizes pontuais;
4. Consideração mais aprofundada da natureza de cada uma das relações do conhecimento do domínio, na sugestão de perícias a serem tutoradas. Como passo subsequente, pode-se explorar padrões de relacionamento entre duplas ou triplas de perícias. Ou ainda, critérios como penalizar a sugestão de tópicos que possuam pré-requisitos indiretos ainda não desenvolvidos;
5. Estudo e melhoria do processo de avaliação das resoluções do aprendiz, objetivando a detecção de aspectos como o desenvolvimento de apenas algumas perícias do enunciado;
6. Implementação da granularidade para a atualização do Modelo do Aprendiz, pois a resolução de um único enunciado que contemple determinada perícia pode não significar que ela foi desenvolvida. Deve-se admitir uma progressão mais gradual que denote esse crescimento;
7. Tratamento das perícias iniciais do conhecimento do domínio, considerando que podem envolver enunciados de raciocínio lógico anteriores à utilização de uma linguagem formal (construção de algoritmos ou implementação de programas);
8. Admissão da proficiência do aprendiz, para que não seja necessário sujeitá-lo a enunciados que contribuiriam para o desenvolvimento de perícias já adquiridas;
9. Embarcamento da solução em um ambiente de ensino para promover a tutoria inteligente (e.g. FARMA-ALG [Kutzke 2015]);
10. Investimento na representação externa de mecanismos adicionais, nesse ambiente, que permitam extrair e apresentar informações úteis ao processo de ensino-aprendizagem (mineração de dados).

Referências

Ahmad, A.-R., Basir, O., and Hassanein, K. (2004). Adaptive user interfaces for intelligent e-learning: Issues and trends. *The Fourth International Conference on Electronic Business (ICEB)*, pages 925–934.

Giraffa, L. M. M. (2003). Fundamentos de sistemas tutores inteligentes. Relatório técnico, Pontifícia Universidade Católica do Rio Grande do Sul.

Goldstein, I. P. (1979). The genetic graph: A representation for the evolution of procedural knowledge. *International Journal of Man-Machine Studies*, 11(1):51–77.

Gómez-Martín, M. A., Gómez-Martín, P. P., and González-Calero, P. A. (2005). Game-based learning as a new domain for case-based reasoning. *1st Workshop on Computer Gaming and Simulation Environments*.

Kutzke, A. R. (2015). *Informática Educacional e a Mediação do Erro na Educação: Um Estudo Teórico-Crítico e Uma Proposta de Instrumento Computacional*. Tese de doutorado, Universidade Federal do Paraná, Curitiba.

Leffa, V. J. (2014). Gamificação adaptativa para o ensino de línguas. *Congreso Iberoamericano de Ciencia, Tecnología, Innovación y Educación*, pages 1–12.

Lesgold, A., Rubinson, H., Feltovich, P., Glaser, R., Klopfer, D., and Wang., Y. (1988). *The nature of expertise*, chapter Expertise in a Complex Skill: Diagnosing X-ray Pictures, pages 311–342. Lawrence Erlbaum Associates, Inc, Hillsdale, NJ, England.

Machado, G. M., Maran, V., de Oliveira, J. P. M., Gasparini, I., and Pernas, A. (2015). Uma revisão sistemática sobre as abordagens ubíquas para recomendação educacional: Estariam elas se tornando adaptativas? *Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 170–179.

Maschio, E. (2013). *Modelagem do Processo de Aquisição de Conhecimento Apoiado por Ambientes Inteligentes*. Tese de doutorado, Universidade Federal do Paraná, Curitiba.

Maschio, E. and Direne, A. I. (2015). Múltiplas representações externas no suporte à aquisição de conhecimento em programação de computadores. *Revista Brasileira de Informática na Educação (RBIE)*, 23(3):81–96.

Mitrovic, A. and Martin, B. (2004). Evaluating adaptive problem selection. In *Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference*, volume 3137, pages 185–194, Eindhoven, The Netherlands. Springer Berlin Heidelberg.

Moreira, C. (2016). *Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores*. Dissertação de mestrado, Universidade Federal do Paraná, Curitiba.

Norvig, P. (2001). Teach yourself programming in ten years. norvig.com/21-days.html.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, (4):251–277.

Pimentel, A. R. and Direne, A. I. (1998). Medidas cognitivas no ensino de programação de computadores com sistemas tutores inteligentes. *Revista Brasileira de Informática na Educação (RBIE)*, 3:17–24.

Silva, R. C. (2015). *Sequenciamento Adaptativo de Exercícios Baseados na Correspondência entre a Dificuldade da Solução e o Desempenho Dinâmico do Aprendiz*. Tese de doutorado, Universidade Federal do Paraná, Curitiba.

Soldado, T. D. and du Boulay, B. (1995). Implementation of motivational tactics in tutoring systems. *Journal of Artificial Intelligence in Education*, 6(4):337–378.

Weber, G. and Brusilovsky, P. (2001). ELM-ART: An adaptive versatile system for web-based instruction. In *International Journal of Artificial Intelligence in Education*, volume 12, pages 351–384.

Weragama, D. S. (2013). *Intelligent Tutoring System for Learning PHP*. Tese de doutorado, Queensland University Of Technology, Brisbane, Austrália.

Construção Colaborativa de Signos Específicos da Língua Brasileira de Sinais para Termos da Subárea de Engenharia de Software

José Augusto Fabris[1], Soraia Silva Prietch[2], Kefferson Ricardi[1]

[1]Curso de Ciência da Computação. Faculdade Anhanguera. Av. Ary Coelho, 829 - Cidade Salmen, 78705-050, Rondonópolis-MT, Brasil

[2]Curso de Sistemas de Informação. Universidade Federal de Mato Grosso. Câmpus de Rondonópolis. Av. dos Estudantes, nº 5055. Bairro Cidade Universitária. 78735-901, Rondonópolis-MT, Brasil

joseaugustoo2011@hotmail.com, soraia@ufmt.br, keffersonricardi@gmail.com

Abstract. *Deaf students are losing interest in the search for knowledge because of the absence of content, books, vocabularies, dictionaries or glossaries with specific academic jargon in Libras. Therefore, this paper presents related researches highlighting the precariousness of terms referring to Software Engineering contents in Libras. The signs constructed and documented in the technical files were submitted to opinion research with people who are deaf, Libras users with completed higher education or in progress. As a result, it was evident from participants' opinion that it is important to construct signs of the Libras for areas of knowledge that still have few signs, such as Computing area.*

Resumo. *Estudantes surdos estão perdendo o interesse na busca do conhecimento por não ter acesso a livros, vocabulários, dicionários ou glossários com jargões acadêmicos específicos em Libras. Portanto, esse artigo apresenta trabalhos correlatos ressaltando a precariedade de termos referentes a conteúdos de Engenharia de Software em Libras. Os signos construídos, e documentados nas fichas técnicas, foram submetidos à pesquisa de opinião de pessoas surdas usuárias da Libras com ensino superior concluído ou em andamento. Como resultado ficou evidente, pela opinião dos participantes, a importância de construir sinais da Libras para áreas de conhecimento que ainda possuem poucos sinais, como a área de Computação.*

1. Introdução

O ponto de partida para a realização do presente trabalho foi a experiência e o desejo pessoal do primeiro autor, desse artigo, enquanto estudante surdo usuário da Língua Brasileira de Sinais (Libras) de um curso de graduação de Ciência da Computação. O referido estudante vem conquistando seu espaço, tanto no mercado de trabalho como em sua educação formal, usufruindo de seus direitos como sujeito participativo e crítico na sociedade. Contudo, um dos maiores desafios para sua inclusão plena é a barreira da comunicação. Como um sujeito que percebe o mundo através de experiências visuais,

torna-se evidente a necessidade de uma comunicação mais aberta em relação às duas Línguas Brasileiras: Português e Libras, em especial, no que se refere aos conteúdos e aos jargões específicos de uma área de conhecimento.

A Libras foi reconhecida no Brasil como segunda língua oficial [Lei nº. 10.436 (2002); Decreto nº 5.626 (2005)], apesar de ser utilizada pela comunidade surda muito antes desta data. A Libras é um idioma relativamente novo, que sofre mudanças ao longo do tempo, como qualquer outro idioma, e se torna cada vez mais rico. Como indivíduo, o estudante surdo, vê a necessidade de construção de novos signos linguísticos específicos para garantir aprendizagem significativa, enriquecer o vocabulário e proporcionar comunicação clara e aberta entre aluno e professor.

Além disso, no município em que a pesquisa foi realizada, não existem intérpretes da Libras com conhecimento profundo ou formação na área de Computação. Antes das aulas existe uma preparação por parte dos profissionais intérpretes; no entanto, durante as aulas, sinais são gerados para tentar suprir a falta de representações coerentes para os termos usados na língua portuguesa para a área específica. Boscarioli *et al*. (2015) traz um exemplo dessa situação em que o termo "computação em nuvem" foi utilizado em uma aula e, assim como esse, por falta de conhecimento específico na área, diversos sinais são criados sem a compreensão sólida dos conceitos. Ao refletir sobre o assunto, se em cada sala de aula de cursos de computação diferentes sinais estão sendo gerados isoladamente, então torna-se muito difícil garantir igualdade de oportunidades ao público usuário da Libras, tais como, realizar mobilidade acadêmica, participar do Exame Nacional para Ingresso na Pós-Graduação em Computação (POSCOMP), dentre outras.

Com o objetivo de auxiliar no processo de ensino-aprendizagem, surgiu a necessidade da construção colaborativa de signos linguísticos em Libras para termos técnicos para a disciplina de Engenharia de Software presente em cursos da área de Computação, podendo promover melhorias na comunicação, na interação entre os presentes em sala de aula e na compreensão de conteúdos específicos.

2. Metodologia de pesquisa

Os procedimentos metodológicos foram realizados conforme a sequência:

- Buscar e assistir aos vídeos disponíveis na Internet a respeito de dicionários, vocabulários, glossários e relação de termos da área de computação/informática em Libras, com foco de atenção para os conteúdos de Engenharia de Software;
- Pesquisar sobre trabalhos correlatos, ou seja, localizar trabalhos na literatura que tenham realizado pesquisa semelhante a esta proposta;
- Efetuar contatos com pessoas surdas que frequentam ou frequentaram cursos na área de computação/informática e com pessoas surdas que trabalham na área de Tecnologia da Informação (TI), a fim de convidá-los posteriormente para participar como voluntários na pesquisa;
- Definir relação de termos selecionados, e seus significados, a partir de livros usados na disciplina de Engenharia de Software para que sinais da Libras sejam propostos;
- Construir os sinais da Libras para o domínio específico, de acordo com os significados/conceitos dos termos selecionados. A construção dos signos linguísticos

foi realizada pelo estudante surdo, usuário nativo da Língua Brasileira de Sinais, em colaboração com profissional intérprete da Libras (ambos autores desse artigo);

- Elaborar questionário para efetuar pesquisa de opinião sobre os sinais construídos;
- Aplicar questionário em pesquisa de opinião com participantes voluntários, que sejam pessoas surdas ou com deficiência auditiva, de forma presencial, e que, de alguma forma, tiveram contato com a área de computação/informática.

3. Trabalhos Correlatos

Para localizar os trabalhos disponíveis na literatura, uma busca exploratória, no Google Acadêmico, foi realizada utilizando as seguintes palavras-chave combinadas: {{Glossário} OR {Vocabulário} OR {Dicionário}} AND {{Libras} OR {Língua Brasileira de Sinais}} OR/AND {{Computação} OR {Informática} OR {Engenharia de Software}}, somente em língua portuguesa, sem se preocupar com o ano de publicação. A partir dessa busca, alguns trabalhos foram encontrados, sendo que dentre eles 06 (seis) foram considerados mais próximos ao que desejava-se propor na presente investigação. Os trabalhos são os seguintes, os quais são brevemente descritos na sequência, em ordem cronológica de publicação, destacando os pontos mais interessantes: Faqueti, Grandi, Fantini e Lorenzetti (2005); Oliveira e Stumpf (2013); Souza, Lima e Pádua (2014); Kuhn (2015); Borges, Obara, Leite e Rocha (2015); Brochado, Lacerda e Rocha (2016).

O projeto de Faqueti, Grandi, Fantini e Lorenzetti (2005) informa que a pesquisa se iniciou com entrevistas com alunos surdos do ensino médio para se definir os componentes básicos (levantar requisitos) do InfoLibras. Posteriormente, “Foram criados 320 sinais relacionados com palavras específicas da informática. Os sinais foram criados a partir de uma necessidade enquanto acadêmico surdo do Curso de Ciência da Computação [...]” (*ibid.*, p. 2865).

Oliveira e Stumpf (2013) relatam, em seu artigo, sobre o processo de desenvolvimento do Glossário Letras-Libras online, da Universidade Federal de Santa Catarina (UFSC), para sinais acadêmicos. Conforme as autoras (*ibid.*, p. 2017), “o número de estudantes surdos nas universidades, particulares e públicas tem aumentado no Brasil, notadamente na última década”, especialmente devido ao Decreto nº 5.626/2005. Portanto, a relevância da construção de sinais para domínios específicos ainda está em sua fase inicial e muito precisa ser realizado a esse respeito. Na equipe de criação do Glossário Letras-Livras, Oliveira e Stump (2013, p. 218) mencionam que “A tradução destes termos técnicos demanda discussão permanente da equipe, pesquisa e criação de neologismos a fim de minimizar a dificuldade dos estudantes na compreensão dos textos acadêmicos”. Isso porque a Libras é uma língua dinâmica e porque possui diferenças culturais (regionalismos). Inclusive, a equipe trabalha com grupos de outros estados brasileiros para que essas diferenças sejam representadas e registradas. Para a produção dos sinais para o Glossário Letras-Libras, Oliveira e Stumpf (2013) informam que os conteúdos dos vídeos são padronizados seguindo a mesma estrutura: (1) Soletração da palavra; (2) Sinal a ser utilizado pelos tradutores do curso; (3) Explicação do termo; (4) Exemplos; e, (5) Variações regionais. Contudo, “Caso a equipe não conheça um ‘sinal’ em Libras que corresponda ao termo em português e, ainda não seja possível apresentar uma proposta de neologismopara o termo, o Glossário tem a seguinte apresentação: *Soletração da palavra; *Explicação do termo; *Exemplos” (*ibid.*, p.

223). Além disso, Tennant e Brown (1998) *apud* Oliveira e Stumpf (2013, p. 223-224) mencionam que "[…] uma combinação de sinais pode ser suficiente para representar o conceito até que um sinal mais eficiente seja criado pela comunidade surda".

O trabalho de Souza, Lima e Pádua (2014, p. 76) foi motivado por "Dois problemas […]: 1) o da escassez do léxico terminológico em Libras nos campos da Ciência, Tecnologia, Cultura e Arte e 2) o do ônus da escassez do léxico que recai sobre o estudante surdo tornando seu acesso e permanência nas esferas acadêmicas superiores extremamente dificultado". Os autores realizaram estudo de caso para a proposta de termos em Libras da área de Desenho Arquitetônico.

Complementando o problema da escassez de sinais em Libras para diversas áreas, Souza, Lima e Pádua (2014, p. 81) questionam "Por que não temos dicionários terminológicos bilíngues bimodais, em Libras? De quais maneiras pode-se criar, recolher e/ou validar sinais em Libras? Por que os sinais criados todos os dias nas salas de aulas do País não são registrados e compartilhados?". Uma demonstração disso, segundo os autores (*ibid.*), é a comparação entre a edição revisada e aumentada do Novo Deit-Libras - Dicionário Enciclopédico Ilustrado Trilíngue da Língua de Sinais Brasileira, de 2013, que contém 9.828 sinais de Libras, e o Dicionário Houaiss em Língua Portuguesa, que contém em torno de 228.500 verbetes.

Souza, Lima e Pádua (2014) proõem uma ficha léxico-terminográfica, adaptada de Lima (2014, p. 122), a qual abarca diversas informações a respeito da construção de um sinal em Libras de determinada área de conhecimento e serve como modelo para a criação da ficha elaborada para uso nesse trabalho. Souza, Lima e Pádua (2014) recomendam que para a criação de um glossário é preciso reunir um grupo de especialistas, a fim de definir e validar os termos que constarão nesse documento ou aplicação. Nesse processo, os autores mencionam a importância de constar "representantes da comunidade surda, surdos que já tenham conhecimento sobre a disciplina, alunos que estão aprendendo a disciplina, e intérpretes" (*ibid.*, p. 86). Isso mostra a importância de se conduzir a construção de signos seguindo o rigor metodológico e considerando as pessoas certas no processo de construção.

A dissertação de Kuhn (2015) propõe a criação de material didático para o curso de Engenharia de Produção em Libras, usando como referência 10 áreas e 55 subáreas regulamentadas pela Associação Brasileira de Engenharia de Produção (ABEPRO). Sendo assim, o material produzido por Kuhn (2015) conta com o total de 65 novos sinais da Libras para a área específica. Foi criado através do estudo dos conteúdos em livros do domínio de interesse para ter sólido embasamento teórico sobre os termos definidos, da produção de vídeo amador e de desenhos técnicos. Para isso, a autora utilizou os 05 (cinco) parâmetros da Libras para a composição dos sinais e das informações contidos nos desenhos técnicos.

Borges, Obara, Leite e Rocha (2015) propõem, em seu resumo, um "Glossário interativo de Libras para a área de Computação", cuja motivação foi a ampliação do acesso e da permanência da pessoa surda em ambiente educacional. Os autores mencionam que existem materiais didáticos adaptados para surdos, porém ainda não é suficiente; pois, muitos termos de áreas profissionais ainda não possuem representação em Libras. A motivação dos autores reforça o que os demais trabalhos correlatos encontrados na literatura também evidenciaram, que ainda não existem sinais suficientes

em Libras que abranjam diversas áreas de conhecimento específicas. Borges, Obara, Leite e Rocha (2015) informam que, em levantamento preliminar, reuniram 310 termos técnicos da área de Computação/ Informática, e conta com um avatar 3D animado, chamada Stayce. Não ficou claro se os termos levantados se encontram em Libras ou se são termos em língua portuguesa a serem criados.

O resumo de Brochado, Lacerda e Rocha (2016) relata sobre a proposta de projeto de pesquisa em andamento intitulado como "Desenvolvimento de Software Glossário de Informática com Aplicação de Libras e de Tecnologia de Captura de Movimento 3D". O projeto conta com a parceria entre a Universidade Estadual do Norte do Paraná (UENP) e a Universidade Tecnológica Federal do Paraná (UTFPR)/ Câmpus de Cornélio Procópio. Uma das etapas desse projeto de pesquisa consistiu na definição dos sinais em Libras, realizada em diferentes meios de pesquisa (sendo 120 verbetes analisados), agregando aqueles sinais já utilizados por surdos atuantes na área. Após a seleção, realizada por estudantes de Engenharia da Computação, a relação de sinais resultante foi analisada por uma professora surda, tradutora/ intérprete da Libras, em que descobriu-se variantes de sinais já utilizados por outros grupos.

A partir da leitura dos trabalhos supracitados é possível verificar que ainda existem poucas pessoas surdas em cursos de ensino superior, por esse motivo sinais devem ser criados por pessoas nativas de sua língua, sendo o caso, pois a língua natural dos surdos brasileiros é a Libras. Dentre os trabalhos selecionados, 03 (três) se referem à Dicionários ou Glossários para a área de Computação/ Informática em Libras [Faqueti, Grandi, Fantini e Lorenzetti (2005); Borges, Obara, Leite e Rocha (2015); Brochado, Lacerda e Rocha (2016)]. No entanto, nenhum dos três recursos propostos foram encontrados na Web para uso ou consulta. Portanto, além de construir sinais novos para conteúdos e jargões específicos, ainda se faz necessária a disponibilização e a divulgação desses sinais, de modo que se tornem acessíveis para um grande número de pessoas surdas interessadas pela área de Computação.

4. Construção de Signos da Libras para a Subárea da Computação - Engenharia de Software

4.1 Preparação para a pesquisa de opinião

Nesta etapa, foi definida a relação de termos, e seus significados, da subárea da Engenharia de Software para que sinais da Libras sejam propostos, a saber: (1) Engenharia de Software; (2) Processos; (3) Modelo de processo/ Ciclo de vida; (4) Modelos de Processos Prescritivos; (5) Modelo Cascata; (6) Modelo Incremental; (7) Modelo Evolucionário; e, (8) Modelo Espiral. A escolha desses termos ocorreu pela afinidade do estudante surdo (primeiro autor desse artigo) com o assunto durante as aulas que teve na disciplina de Engenharia de Software no Curso de Ciência da Computação. Os termos foram definidos utilizando as seguintes referências: Sommerville (2003); Pressman (2011); Wazlawick (2013); e, Paula Filho (2015).

4.2 Desenvolvimento

A etapa de preparação para pesquisa foi subdividida em três atividades: construção dos signos da Libras para a subárea Engenharia de Software, preparação e execução da pesquisa de opinião, as quais são descritas na sequência.

A construção dos sinais da Libras para o domínio específico, de acordo com os significados dos termos escolhidos foi proveniente da interação aluno-intérprete em sala de aula, e de estudo realizado por intérprete junto com o autor (estudante surdo). Para documentar os sinais construídos, levou-se em consideração o conhecimento e os exemplos extraídos do embasamento teórico e dos trabalhos correlatos. Um modelo de ficha foi elaborado para uso nesse trabalho, tomando como base a ficha léxico-terminográfica de Lima (2014, p. 122) *apud* Souza, Lima e Pádua (2014, p. 85). Para cada um dos 08 (oito) termos foram construídos 03 (três) sinais em Libras, totalizando em 24 fichas preenchidas. Esses sinais foram filmados e os dados referentes aos sinais foram documentados usando o modelo. No Quadro 1, uma ficha preenchida é apresentada a fim de exemplificar a forma de documentação dos sinais.

Quadro 1. Ficha da Opção 1 de sinal para o termo Engenharia de Software.

Nº: 01/A
(a) Termo em língua portuguesa: Engenharia de Software
1)(b) Definição do termo com base na literatura especializada (acepção), em língua portuguesa: "A engenharia de software abrange um processo, um conjunto de métodos (práticas) e um leque de ferramentas que possibilitam aos profissionais desenvolverem software de altíssima qualidade".
(c) Ilustração/ Representação, conforme referências:
(d) Quantidade de palavras/sinais do termo: 03
(e) Descrição de cada parâmetro da Libras, considerando o **início** da comunicação do sinal
(i) Configuração de mãos (Palavra/Sinal 1): Conforme tabela número 44 - configuração da mão em "Pinça"; Configuração de mãos (Palavra/Sinal 2): Configuração número 7, forma "S"; Configuração de mãos (Palavra/Sinal 3): Configuração da mão em 52, forma de "W".
(ii) Ponto de articulação (Palavra/Sinal 1): Posterior de braço iniciando do punho primeiro toque e finalizando no cotovelo segundo toque; Ponto de articulação (Palavra/Sinal 2): Articulação no espaço neutro diante do corpo; Ponto de articulação (Palavra/Sinal 3): Articulação no espaço neutro diante do corpo.
(iii) Movimento (Palavra/Sinal 1): Percorre posterior do braço; Movimento (Palavra/Sinal 2): No espaço neutro formando um ângulo de 90°; Movimento (Palavra/Sinal 3): Sinal estático (sem movimento inicial e/ou final).
(iv) Orientação/direção (Palavra/Sinal 1): Da direita para esquerda; Orientação/direção (Palavra/Sinal 2): "S" De baixo para cima; Orientação/direção (Palavra/Sinal 3): "W" De baixo para cima.
(v) Expressão facial e corporal (Palavra/Sinal 1): Expressão facial neutra, braço inclinado a um ângulo de 90°; Expressão facial e corporal (Palavra/Sinal 2): Expressão facial neutra, leve inclino do corpo; Expressão facial e corporal (Palavra/Sinal 2): Expressão facial neutra, ombros levemente elevados.
(f) Fotos/Desenhos do sinal: 
(g) Link para o vídeo: <A ser disponibilizado>
(h) Situação em que o signo (sinal) para o termo foi construído: Estudante em conjunto com o intérprete.
(i) Referências: Pressman (2011)

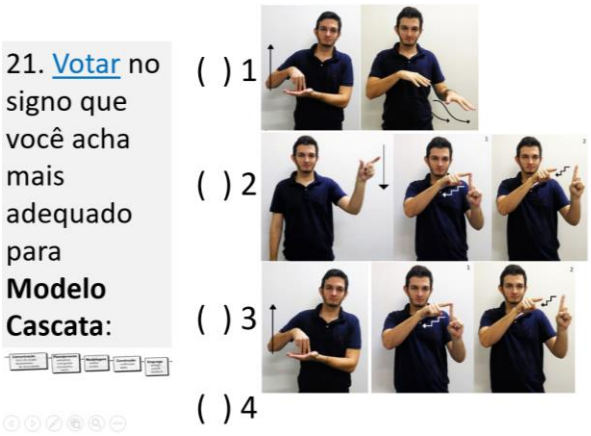
Fonte: Ficha adaptada de Lima (2014, p. 122) *apud* Souza, Lima e Pádua (2014, p. 85); (Fotos do primeiro autor).

4.3 Pesquisa de opinião

Após a elaboração das fichas, iniciou-se a fase de preparação, que consistiu na elaboração de um questionário para verificar se os participantes conheciam sinais em Libras para a relação de termos escolhidos, para identificar o perfil das pessoas surdas

que conhecem ou que estudaram esses termos previamente e para efetuar votação dentre os sinais construídos. Uma das perguntas do questionário consta ilustrado na Figura 1, com três opções de respostas fechadas (1, 2 ou 3) e uma aberta (4) caso o participante quisesse sugerir um sinal novo ou um sinal aprendido previamente.

Figura 1. Formato de uma das questões do segundo modelo de questionário utilizado na pesquisa de opinião.



Fonte própria (Fotos do primeiro autor).

Duas pessoas surdas participaram voluntariamente, sendo um egresso do curso de Ciência da Computação e um estudante do curso de Pedagogia. Ambas as participações foram realizadas de forma individual, presencial e em Libras. O pesquisador principal (estudante surdo) conduziu a pesquisa de opinião sozinho, apenas sanando algumas raras dúvidas com o intérprete. Vale comentar que o convite aos participantes ocorreu por conveniência, pois os voluntários estudavam na mesma instituição de ensino que o pesquisador principal. Por esse motivo, ocorreu a participação de um estudante de um curso de Pedagogia, que não era da área de Computação. A intenção era a de recrutar participantes surdos, usuários da Libras, que tivessem contato com a área de Computação, porém, considerando todas instituições de ensino superior da cidade, no período de realização da pesquisa (ano de 2017), os únicos dois estudantes surdos eram o pesquisador principal e um dos participantes voluntários. Houve algumas tentativas de realizar a pesquisa de opinião de forma remota, com estudantes surdos de outras localidades, porém, não obteve-se sucesso.

5. Resultados

Primeiramente, é importante destacar que o conteúdo do questionário foi o mesmo aplicado na Pesquisa de Opinião 1 e na Pesquisa de Opinião 2, o que mudou foi somente o formato como as perguntas estavam dispostas na tela do computador. Levando isso em consideração, as respostas ao questionário são passíveis de comparação. O questionário foi composto por 25 perguntas. A primeira pergunta para conhecer o nome completo, que não será transcrito aqui devido ao respeito ao anonimato dos participantes. As respostas para as questões de número 2 a 16 constam no Quadro 2, as de número 17 a 24 constam no Quadro 3, e as respostas à questão 25 constam na forma textual, as quais foram traduzidas por um intérprete.

Quadro 2. Respostas às perguntas de 2 a 16 do questionário.

Perguntas	Participante 1	Participante 2
2. Idade	17	26
3. Sexo	Masculino	Masculino
4. Nível de conhecimento em Libras (auto avaliação)	8	9
5. Nível de escolaridade	Ensino superior incompleto	Ensino superior completo
6. Curso de graduação	Pedagogia	Ciência da Computação
7. Instituição de formação, cidade e estado	Unic (antiga Faculdade Anhanguera), Rondonópolis-MT	
8. Classificação sobre a percepção de existência de sinais da Libras para termos específicos da área	Não se aplica	7
Conhece algum sinal para se referir ao termo:		
9. Engenharia de software?	Sim (vídeo)	Sim (vídeo)
10. Processos?	Não	Sim (vídeo)
11. Modelo de processo/ Ciclo de vida?	Não	Sim (vídeo)
12. Modelos de Processos Prescritivos?	Não	Sim (vídeo)
13. Modelo Cascata?	Não	Sim (vídeo)
14. Modelo Incremental?	Não	Não
15. Modelo Evolucionário?	Não	Não
16. Modelo Espiral?	Não	Não

No Quadro 2, dentre as respostas afirmativas para as questões de 9 a 13, consta entre parênteses a palavra 'vídeo' para destacar que houve a filmagem do sinal informado pelo participante durante a entrevista.

Quadro 3. Respostas às perguntas de 17 a 24 do questionário.

Voto para sinal proposto para:	Participante 1	Participante 2	Parecer
17. Engenharia de Software	1	1	1
18. Processos	2	1	0
19. Modelo de processo/ Ciclo de vida	2	1	0
20. Modelos de Processos Prescritivos	1	1	1
21. Modelo Cascata	1	1	1
22. Modelo Incremental	3	3	1
23. Modelo Evolucionário	3	1	0
24. Modelo Espiral	3	3	1
	Quantidade de votos nas mesmas opções		5

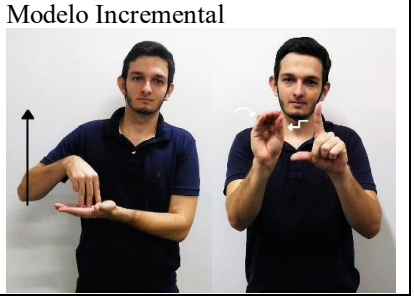
Fonte própria. Legenda (Parecer): Votos na mesma opção = 1; Votos em diferentes opções = 0.

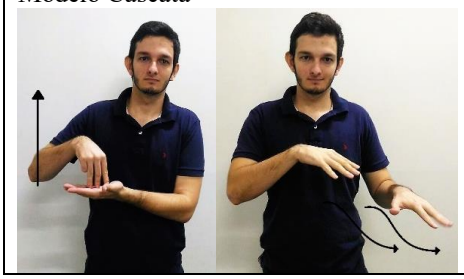
Como pode ser visto no Quadro 3, 05 (cinco) dentre 08 (oito) termos foram votados por ambos os participantes. Um desses sinais é aquele que consta no Quadro 1 e os demais constam ilustrados na Figura 2.

Figura 2. Quatro sinais votados por ambos os participantes.

Modelos de Processos Prescritivos

Modelo Incremental



Modelo Cascata	Modelo Espiral
	

Fonte própria (Fotos do primeiro autor).

Com relação à pergunta 25, se os entrevistados consideravam importante criar sinais da Libras para áreas de conhecimento que ainda possuem poucos sinais, como a computação, por exemplo. Ambos responderam que é importante.

6. Considerações finais

O objetivo geral era auxiliar no processo de ensino-aprendizagem, a partir da construção colaborativa de signos linguísticos em Libras para termos técnicos para a disciplina de Engenharia de Software presente em cursos da área de Computação. Transcorrido o desenvolvimento do trabalho, foi possível alcançar parcialmente o objetivo apresentado. A contribuição principal foi a colaboração entre o estudante surdo (autor principal desse artigo) com o professional intérprete da Libras na construção dos signos linguísticos para para termos selecionados. Essa construção contou com esforço, estudo e dedicação de ambos. O ponto que apresentou limitações foi a quantidade de participantes voluntários na pesquisa de opinião.

Além disso, também foi possível o desenvolvimento de novas termologias derivadas de conceitos pré-existentes fundamentados e suas definições conforme previsto no projeto. A elaboração de fichas catalogadas com cada termo-acadêmico-cientifica foi essencial no decorrer de cada entrevista o que proporcionou a aplicação prática dos conceitos assimilados. A realização de entrevistas se mostrou favorável à aceitação destes novos termos pelos participantes voluntários.

A intenção é mostrar que a construção de termos linguísticos significativos e suas termologias, contribuem muito a um conhecimento inteligível e sua assimilação. Além disso, pode ser um incentivo para que novos trabalhos acadêmicos, projetos e software voltados aos estudantes surdos possam ser produzidos, alcançando toda a comunidade surda. Portanto, esse trabalho poderá promover atração de mais estudantes surdos para cursos da área de computação, ampliar as condições para compreensão dos conteúdos por estudantes matriculados em cursos da área, colaborar com a ampliação de vocabulário para a Libras, e tentar padronizar sinais de jargão específico da engenharia de software, independente de regionalismos da língua.

Uma possibilidade de continuidade desta pesquisa é levar os signos linguísticos construídos para discussão no Fórum de Estudos dos Surdos na Área da Informática (FESAI), cujo objetivo “é desenvolver e incentivar pesquisas para criação de novos sinais e existentes de sinais de informática para divulgar na educação básica e no ensino

superior" (FESAI, 2017). Os sinais são propostos e votados de forma colaborativa e democrática pelos participantes, com pessoas que atuam na área de informática.

Referências

Brasil. Lei nº. 10.436, de 24 de abril de 2002. Dispõe sobre a Língua Brasileira de Sinais – Libras e dá outras providências.

Brasil. Decreto nº 5.626, de 22 de dezembro de 2005. Regulamenta a Lei nº 10.436, de 24 de abril de 2002, que dispõe sobre a Língua Brasileira de Sinais – Libras.

Borges, Lucas C.; Obara, Eduardo N.; Leite, Claudia Z.; Rocha, Fabiano G. (2015). Glossário interativo de Libras para a área de Computação. Computer on the Beach, Florianópolis/SC.

Boscarioli, Clodis; Galante, Guilherme; Oyamada, Marcio Seiji; Zara, Reginaldo A., Villwock, Rosangela. (2015). Aluno Surdo na Ciência da Computação: Discutindo os Desafios da Inclusão. XXIII Workshop sobre Educação em Informática, Recife, PE.

Brochado, Sonia Maria D.; Lacerda, Cristina Broglia de F.; Rocha, Luiz Renato M. da. (2016). Projeto de Pesquisa: Software Glossário de Informática com Aplicação de Libras e de Tecnologia de Captura de Movimento 3D. Journal of Research in Special Educational Needs, Volume 16, Number s1, p. 905–908.

Faqueti, Charlles Giovany; Grandi, Gilberto; Fantini, Liliane S.; Lorenzetti, Maria Lúcia. (2005). InfoLIBRAS – O Uso da Web para o Aprendizado da Língua de Sinais com Termos da Informática. XXV Congresso da Sociedade Brasileira de Computação, XXII Workshop de Informática na Escola (WIE), São Leopoldo/RS.

FESAI. Fórum de Estudos dos Surdos na Área da Informática. Disponível em: https://fesaiblog.wordpress.com/, acesso dia: 10/04/2017.

Kuhn, Talícia do Camo G. (2015). Processo de Criação de Termos Técnicos em Libras para Engenharia de Produção. Dissertação de Mestrado em Ensino de Ciência e Tecnologia, da Universidade Tecnológica Federal do Paraná. Ponta Grossa/PR.

Oliveira, Janine S.; Stumpf, Marianne R. (2013). Desenvolvimento de glossário de Sinais Acadêmicos em ambiente virtual de aprendizagem do curso Letras-Libras. INFORMÁTICA NA EDUCAÇÃO: teoria & prática, Porto Alegre, v. 16, n. 2.

Paula Filho, Wilson de Pádua. (2015). Engenharia de Software: fundamentos, métodos e padrões. 3.ed. - [Reimpr.]. - Rio de Janeiro: LTC.

Pressman, Roger S. (2011). Engenharia de Software: Uma Abordagem Profissional. 7.ed. – Porto Alegre: AMGH.

Sommerville, Ian. Software Engineering. São Paulo: Pearson Addison Weslwy, 2003.

Souza, Celso Luiz de; Lima, Vera Lucia de S. e; Pádua, Flávio Luís Cardeal. Abordagem Interdisciplinar para a Criação e Preservação de Novos Sinais para Dicionários Terminológicos em Libras. Acta Semiótica et Lingvistica, Universidade Federal da Paraíba, v. 19, n. 1, 2014. Disponível em: goo.gl/hFb8Pe, acesso dia: 06/04/2017.

Wazlawick, Raul Sidnei. Engenharia de Software: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

Ponteiros em Papel: O ensino e a aprendizagem de ponteiros em linguagem de programação C com base em envelopes coloridos

Gustavo Kira[1], **Luiz Ernesto Merkle**[2]

[1]Departamento de Ciência da Computação – Universidade do Estado de Santa Catarina (UDESC) Joinville, SC.

[2]Departamento de Informática – Universidade Tecnológica Federal do Paraná (UTFPR) Curitiba, PR.

gustavokira@gmail.com, merkle@utfpr.edu.br

Abstract. *This paper presents a teaching and learning dynamic with colored envelopes as a way to work with pointers in the C language. Pointer mastery is an important step to understand C at same time it is one of it's most difficult topics. This activity was planned assuming that learning pointers, without a concrete base, privileges those who already have a good algebra understanding or comprehends well the computer's abstract architecture model. Using envelopes enables a concrete analogy and offers an intermediate step for those who still developing their abilities to handle symbolic manipulation or still learning how to deal with the abstract machine model.*

Resumo. *Este artigo apresenta uma dinâmica com envelopes coloridos como um meio para trabalhar o conceito de ponteiro no ensino e aprendizagem da linguagem de programação C. A mestria de ponteiros é um passo importante para o domínio desta linguagem, ao mesmo tempo que é, talvez, um de seus tópicos mais difíceis. Esta atividade foi pensada com o pressuposto de que o a aprendizagem de ponteiros, sem um apoio concreto, privilegia quem já tem um bom domínio de álgebra ou uma boa compreensão abstrata da arquitetura de um computador. O uso de envelopes permite desenvolver uma analogia concreta e oferece um passo intermediário para aqueles que ainda estão por desenvolver suas habilidades ligadas a manipulação simbólica ou a compreensão abstrata da máquina.*

1. Introdução

As dificuldades no ensino e aprendizagem de uma linguagem de programação não são questões novas como já identificou [Aureliano et al. 2016]. Trabalhos como os de [Raiol et al. 2015], [França e Tedesco 2015], [Anido 2015], [Raeder et al. 2016] e [Da Silva et al. 2016] localizam, em especial, o problema na questão da abstração associada à própria ciência da computação. Mesmo que a abstração seja um conceito central tanto para a computação quanto para o campo de seu ensino e aprendizagem, é díficil encontrar uma reflexão um pouco mais profunda sobre o termo. É comum a referência a [Wing 2006] sobre pensamento computacional, mas em nenhum momento é feita uma reflexão sobre o que é a abstração.

Estrutura de dados é uma disciplina consolidada como parte da ciência da computação no Brasil e tem como conteúdos conceitos considerados abstratos e historicamente associados ao desenvolvimento de programas, tais como pilhas: listas e árvores [Nunes et al. 2015]. Nos Estados Unidos este conteúdo é associado aos dois primeiros anos dos cursos de computação (CS1, CS2) [Bucci et al. 2001, Simon et al. 2010]. Mesmo não sendo uma disciplina sobre linguagens de programação, é comum o uso de uma para a implementação dos conceitos associados a disciplina.

No curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) da Universidade do Estado de Santa Catarina (UDESC), a disciplina de estrutura de dados usa a linguagem C. O conceito de ponteiros desta linguagem é um dos fundamentais para um bom entendimento dos conteúdos da disciplina. Entretanto, também é um dos que discentes têm mais dificuldade [Milne e Rowe 2002, Craig e Petersen 2016].

[Milne e Rowe 2002] também lançam a hipótese de que a dificuldade quanto aos ponteiros e outros conceitos relacionados pode vir da falta de conhecimento, por parte de quem aprende, de como a memória principal do computador funciona, algo também colocado por [Lippert 2018] e por [Craig e Petersen 2016]. Esta hipótese abre espaço para pensar que a habilidade em lidar com conceitos abstratos na computação não é uma característica nata, mas dependente de uma série de outros conceitos do próprio campo.

A observação de [Milne e Rowe 2002] também torna a alta variância de conhecimentos ligados a computação apresentado por quem ingressa em cursos de gradução da área uma variável importante para o ensino e aprendizagem daquilo que é considerado abstrato, pois existe uma certa dependência destes conceitos de outros conceitos da própria computação. O perfil discente das matérias iniciais de um curso de graduação inclui estudantes que estão a mais tempo no curso refazendo a disciplina, egressos de cursos de ensino médio técnico em informática e pessoas que tiveram um contato mais a fundo com a informática pela primeira no ensino superior.

Pensando nessa assimetria de perfis, buscou-se na filosofia e na educação um conceito de abstração que permita lidar um pouco melhor com este problema e que sirva de base para pensar em instrumentos e formas de trabalhar com conceitos abstratos. Com isso, este trabalho propõe enxergar o conceito de abstração como uma forma de mediação ao invés de uma categoria estanque e antagônica ao conceito de concreto. O entendimento como mediação implica também no entendimento do abstrato ou da abstração como sendo sempre um processo que pode ser aprendido e ensinado. Este último desdobramento traz consigo uma perspectiva diferente para a produção de materiais de ensino voltados à computação, uma vez que exercícios, exemplos e exposições devem, de alguma maneira, ajudar os e as estudantes a compreender e obter uma maestria deste processo.

Para iniciar uma discussão sobre esta forma de encarar a abstração, este trabalho está dividido em mais três seções: Primeiro, uma breve introdução das questões que envolvem entender o abstrato como uma mediação entre concretudes; segundo, uma atividade piloto sobre um tema que é tido como abstrato (ponteiros em linguagem C) dentro da computação pensada com o apoio teórico descrito na seção anterior e; por fim, uma pequena discussão sobre o desdobramentos possíveis para a atividade descrita.

2. Abstrato como mediação entre duas concretudes

O minidicionário Aurélio define Abstração como "resultante de abstração", "que opera com qualidades e relações, e não com a realidade" e "que expressa qualidade ou característica separada do objeto a que pertence ou está ligada". Já o concreto, o mesmo dicionário define como "que existe em forma material" e "de consistência mais ou menos sólida" [Holanda 2010].

Para [Meksenas 1992] e [Lefebvre 1991, p.49-50], essa aparente oposição entre os conceitos de concreto e de abstrato é uma espécie de senso comum com raízes na filosofia. A origem do entendimento de ambos como uma oposição binária pode ser localizada na Grécia Antiga, mais precisamente com Aristóteles e a sua distinção entre contemplação, entendida com abstração; e ação, entendida como concretude. A questão torna-se mais complexa do que uma simples associação de ideias ao lembrar que, na Grécia de Aristóteles "todo trabalho prático e produtivo cabia aos escravos, enquanto o pensamento metafísico era uma atividade própria da aristocracia, livre para pensar" [Meksenas 1992, p.93].

Esta perspectiva sobre os conceitos de concreto e de abstrato pode levar a uma visão elitista de conhecimento a qual associa o abstrato a algo somente acessível a uma minoria preparada para tal [Meksenas 1992]. Complementamos que esta mesma perspectiva também leva a entender que existe uma naturalidade para o conhecimento abstrato, algo que a priori não deveria pautar uma agenda de ensino e aprendizagem.

[Silva 2000] também tenta explicar uma outra problemática que surge do uso de oposições binárias como forma de marcação social e linguística: "em uma oposição binária, um dos termos é sempre privilegiado, recebendo um valor positivo, enquanto o outro recebe uma carga negativa" [Silva 2000, p.83]. O positivo é tratado como a "norma" enquanto o negativo é colocado como uma espécie de desvio do que é normal. O problema surge ao entender que esta relação não é simétrica: a norma define o que é correto, muitas vezes percebida como a única forma, delegando ao polo negativo a função de ser "readequado" a norma.

Como pares de conceitos antagônicos e estáticos, tem-se um problema de dualidade: tudo que é concreto não é abstrato, assim como tudo que é abstrato não é concreto. Indo além, se abstração é aquilo que [Silva 2000] chama de "norma", cabe ao concreto a conotação negativa.

[Miliszewska e Tan 2007] mostram uma das problemáticas que deriva deste modelo: "outra dificuldade enfrentada por estudantes de programação está na necessidade de imaginar e compreender vários termos abstratos que não têm equivalentes na vida real[1]". Essa passagem mostra uma percepção que existe um corte claro entre vida real (concreto) e programação de computadores (abstrato), esquecendo das diversas relações entre ambos, desde da presença destes no mundo real até mesmo as analogias possíveis entre os conceitos da vida e do computador.

Para fugir desta dicotomia, [Meksenas 1992] propõe entender o concreto e o abstrato de uma outra forma e apoia-se no entendimento do abstrato como mediação, esta última definida por [Machado 2011, p.55] como algo que situa-se "sempre no meio do

[1] *Another difficulty faced by programming students is the need to imagine and comprehend many abstract terms that do not have equivalents in real life [Miliszewska e Tan 2007]*

processo, constituindo em condição de possibilidade do conhecimento em qualquer área, em vez de ponto de partida ou ponto de chegada. São um degrau necessário que conduz de um patamar de concretude a outro" [Machado 2011, p.55].

[Meksenas 1992, p.95] resume o processo ao colocar que "as diferentes práticas de ensino partem sempre de certos níveis de concretude, chegando, pela abstração, a outros níveis de concretude". Isto faz com que o concreto não seja o contrário do abstrato, nem mesmo podendo ser consideradas entidades de mesmo tipo. Pode-se dizer que o concreto é um dado estado de percepção de um objeto enquanto o abstrato é um processo de passagem para um outro tipo de estado de percepção, este também concreto, mas com outra configuração. A figura 1 ilustra graficamente a proposta de [Meksenas 1992] e torna mais claro qual o papel do abstrato dentro desta perspectiva.

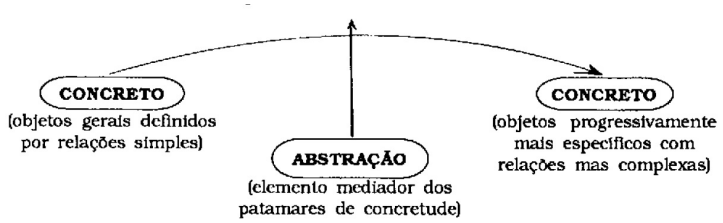


Figura 1. Esquema proposto por [Meksenas 1992]. fonte:[Meksenas 1992, p.95]

[Meksenas 1992] ressalta que o processo não acaba no segundo estágio de "concretude" mas é um processo sem fim. No caso da computação esta perspectiva se alinha muito bem com o tipo de conhecimento exigido na área, dada a grande diversidade e complexidade de temas sob o nome de computação ou informática. Além disso, uma outra implicação mais importante também surge: programação e/ou temas ligados a computação considerados abstratos, na verdade são outros tipos de concretude, mais específicos e mais complexos que os habituais. Ou seja, ensinar e aprender o conceito de árvore apresentado como uma estrutura de dados é fazer uma passagem do conceito de árvore (ser que existe no mundo) para o de árvore (estrutura análoga que possui relações e propriedades ligadas ao computador, a matemática e a planta árvore). Ambos são concretos, mas estão sob óticas diferentes de conhecimento.

Com isso, materiais de ensino para temas tidos como abstratos dentro da computação devem, de alguma maneira, trabalhar com este processo de transformação. Ou seja, dentro desta perspectiva, pode-se lançar a hipótese de que o problema quanto ao aprendizado de abstração é na verdade a falta de medição entre a concretude do corpo discente com a concretude necessária para o domínio dos conceitos a serem trabalhados. Outro entendimento interessante seria o da distância entre as concretudes, uma vez que para um bom entendimento de alguns conceitos, talvez seja necessário diversas transformações de entendimento.

É importante citar que mesmo não usando diretamente o mesmo referencial teórico descrito aqui, existem iniciativas que parecem ter uma mesma premissa, como por exemplo uso de jogos para ensino e aprendizagem de estrutura de dados [Barbosa et al. 2015] e o uso de robótica como apoio ao ensino e aprendizagem de programação [Paparidis e Franco 2016].

Assim, na próxima seção, será apresentada uma atividade que usa envelopes de papel coloridos como uma forma de explicar o funcionamento de ponteiros na linguagem C que não se apoie somente em conceitos da computação (como memória principal ou acesso randômico), mas que use conceitos análogos que fazem parte do cotidiano do corpo discente. Espera-se que este tipo de exercício seja um instrumento de auxílio para que as e os estudantes tenham subsídios para chegar mais perto de um entendimento que inclua os conceitos da computação.

3. Envelopes Coloridos

Ponteiros são um tipo de variável existente na linguagem C e, como colocado anteriormente, são considerados de difícil aprendizado [Milne e Rowe 2002, Adcock et al. 2007, Craig e Petersen 2016]. A ideia de usar envelopes teve como inspiração o experimento dos quatro cartões feito pelo psicólogo inglês Peter Watson em 1966. Para [Machado 2011, 50-53], este experimento mostra que existem duas possibilidades de concretude: uma com relação ao fato do material ser manipulável, uma visão mais próxima do entendimento comum e outra quanto aos significados do material, mais próxima do entendimento de concreto e de abstrato apresentado na última seção. Ao usar envelopes, assim como fez Watson, optou-se por tentar maximizar ambas as dimensões.

Foram desenvolvidos os seguintes materiais: um conjunto de envelopes verdes, um conjunto de envelopes amarelos e um conjunto de envelopes laranjas. Além disso, também foram cortados papéis das mesmas cores para serem colocados dentro dos envelopes. Cada envelope tem duas informações: um "nome" composto por números e letras e um "endereço" composto por apenas números. Na figura 2 temos um envelope de cada cor, sendo o texto grande ("p10", "pp11", "a") o nome do envelope e o número (1005, 2004, 0014) no canto direito inferior o seu endereço.

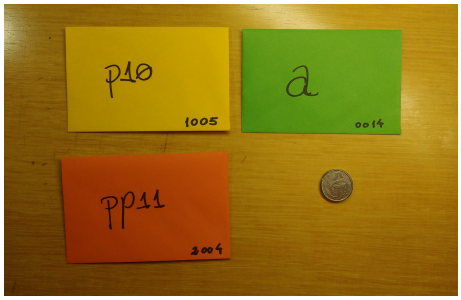


Figura 2. Exemplo dos envelopes usados nesta atividade. fonte: próprio autor.

Já os papéis, dependendo da sua cor, podem ter valores diferentes. Os verdes sempre têm valores numéricos, os amarelos têm somente números equivalentes aos possíveis endereços dos envelopes verdes e os laranjas têm somente números equivalentes aos possíveis endereços dos envelopes amarelos. A figura 3 mostra um exemplo no qual o envelope de nome "pp11" tem um papel com valor 1005 que é o mesmo valor do endereço do envelope amarelo "p10" que por sua vez tem um papel com valor 0014, endereço do envelope verde.

A relação com o conceito de ponteiros é dada da seguinte maneira: Os envelopes verdes representam as variáveis de um programa escrito em C. Tem um nome, tem um endereço de memória e podem ter um valor. O valor é o papel verde guardado dentro

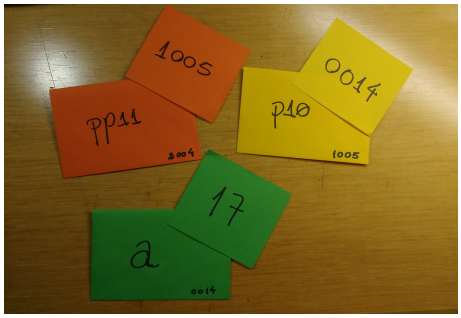


Figura 3. Exemplo dos envelopes e papéis correspondentes. fonte: próprio autor.

do envelope. Os envelopes amarelos são ponteiros de envelope verde, já que guardam somente valores que podem ser endereços de envelopes verdes. E por fim, os envelopes laranjas são ponteiros de envelope amarelo, mantendo a mesma relação vista entre os amarelos e verdes. A figura 4 ilustra um exemplo no qual o envelope amarelo é um ponteiro para o envelope verde, pois tem dentro de si o valor do endereço do envelope verde de nome "a". Ou seja, os envelopes verdes são similares as variáveis inteiras, os amarelos a ponteiros para inteiros e os laranjas são ponteiros para ponteiros de inteiros.

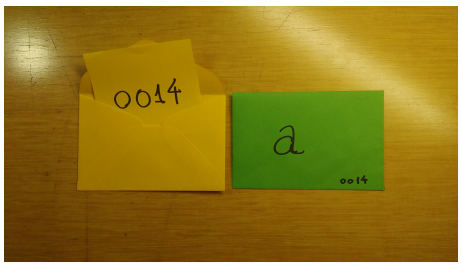


Figura 4. Exemplo de como funciona a analogia com o conceito de ponteiro. fonte: próprio autor.

Foram apresentadas também três regras de manipulação dos envelopes. Primeiro, é possível abrir um envelope chamando-o pelo seu nome. Segundo, é possível pedir o endereço de um envelope usando o sinal de (&) + (o nome do envelope). Terceiro, é possível usar o sinal (*) + (nome do envelope) para ler o valor de um envelope cujo endereço está dentro daquele no qual foi aplicado o sinal de (*). É importante notar que estas são operações associadas ao uso de ponteiros na linguagem C.

Foram pensadas três possibilidades de uso para este conjunto de materiais. (a) Como apoio a uma explicação teórica, (b) como uma atividade interativa, (c) como uma espécie de jogo. Somente as duas primeiras foram executadas, pois durante o planejamento do jogo, este foi desmembrado em um outro projeto com seus próprios materiais. Entretanto, isto não impede de que um jogo usando os envelopes seja feito, apenas que, para esta iteração ele foi descartado.

3.1. Explicação Teórica Concreta

A técnica descrita neste artigo foi aplicada na UDESC, em uma disciplina de estrutura de dados do curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS). Na

grade atual, esta disciplina encontra-se no terceiro semestre do curso. Dado o contexto, todos os e as estudantes que participaram da atividade têm o ensino médio completo e algum tipo de experiência com a linguagem C.

A atividade foi feita com o objetivo de revisar o conceito de ponteiros, pois é um conceito importante para a aprendizagem da implementação das estruturas de dados em C. Ela também foi uma resposta a dificuldade de uma parte considerável da turma em resolver alguns exercícios com ponteiros passados uma semana antes. Ainda cabe mencionar que junto dos discentes do TADS, também frequentam a disciplina estudantes do bacharelado em Ciência da Computação da mesma instituição.

O primeiro uso dos envelopes e cartões foi como material de apoio para uma revisão expositiva sobre ponteiros em C. Neste momento foi feita a apresentação do que são, e como funcionaria a lógica dos envelopes descrita anteriormente. Como o corpo discente já teve pelo menos uma disciplina de linguagem C, as associações entre os envelopes e ponteiros foram também apontadas neste momento.

3.2. Ponteiros em Papel

Explicado como funciona os envelopes, foi pedido para que cada um da turma pegasse um envelope e um valor de mesma cor. De posse deste material, cada estudante deveria cumprir instruções como as seguintes: 1) Escreva uma instrução que execute uma soma com 2 envelopes verdes e 1 amarelo. 2) Escreva uma instrução que execute uma soma usando 3 envelopes amarelos. 3) Escreva uma instrução com apenas envelopes amarelos que execute uma troca do valor de um envelope verde. 4) É possível guardar um papel amarelo em um envelope laranja ou vice-versa?

Uma vez apresentada as questões, os e as estudantes deveriam procurar colegas que tenham outros envelopes e usar as regras dos operadores (*) e (&) para tentar achar envelopes que os ajudariam a responder às questões. A figura 5 mostra a forma como um estudante respondeu as instruções. Neste caso, em especial, é interessante notar que foi feita uma substituição de termos equivalentes a cada passo.

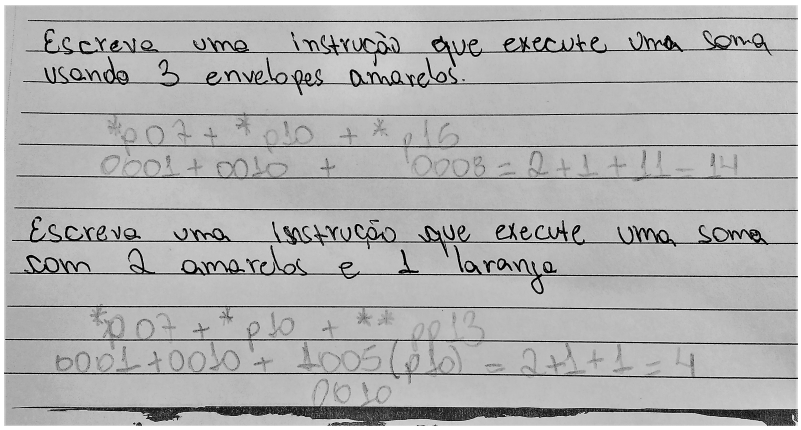


Figura 5. Exemplo de resposta do exercício. fonte: próprio autor.

Vale a pena citar que nem todos prestaram atenção quando foi dito para que pegassem somente os papéis e envelopes separados para a atividade. Alguns pegaram materiais que estavam em uma outra pilha que não deveria ser usada. Para resolver este problema,

alguns estudantes pegaram vários envelopes e organizaram de um modo que fosse possível responder as instruções. Este desvio também se mostrou uma forma contingente para explicar o que acontece ao tentar usar uma variável não declarada.

3.3. Da Concretude do Papel para a Concretude da Tela

Além das atividades com os envelopes, também foi feita uma outra atividade que consistia em transpor a lógica dos envelopes para um código na linguagem C. Foi dado um código inicial com três variáveis: a primeira chamada envelopeVerde e do tipo inteiro(int) com um valor qualquer, a segunda chamada envelopeAmarelo do tipo ponteiro para inteiro com o endereço da variável envelopeVerde com seu valor e a terceira chamada envelopeLaranja do tipo ponteiro para ponteiro de inteiro com o endereço da variável envelopeAmarelo como seu valor. Com isso, o exercício pedia para que as e os estudantes completassem o trecho de código para que ao ser executado fosse exibido na tela: o valor e o endereço de cada uma das variáveis o que obriga os estudantes a aplicarem os operadores (*) e (&), também alvos da atividade com envelopes coloridos.

Esta passagem é importante para que fique clara a relação entre a atividade prática e o conteúdo da disciplina. No código 1, temos um exemplo de trecho de código produzido na atividade. Ele foi alterado para poder ser reproduzido, entretanto as mudanças não afetam a essência do código.

```
1
2     int envelopeVerde = 17;
3     int* envelopeAmarelo = &envelopeVerde;
4     int** envelopeLaranja = &envelopeAmarelo;
5
6     //Parte 1
7     printf("O valor do envelope verde: %i\n", envelopeVerde);
8     printf("O valor do endereco do envelope verde: %i\n", &envelopeVerde);
9
10    //Parte 2
11    printf("O valor do envelope amarelo: %p\n",*envelopeAmarelo);
12    printf("O valor do endereco do envelope Amarelo: %i\n", &envelopeAmarelo);
13    printf("O valor contido no endereco do envelope Amarelo:\n", envelopeAmarelo);
```

Código 1. Exemplo de trecho de código produzido na atividade.

4. Considerações Finais

Mesmo que executada somente uma vez, os códigos entregues e as respostas dos estudantes durante as atividades foram boas o suficiente para dar continuidade ao trabalho sobre esta técnica. Neste momento, são imaginadas quatro frentes: 1) Refinamento do material (envelopes, papéis e disposição das informações). 2) Aplicação em grupos com perfis diferentes 3) Desenvolvimento de mais atividades que usem os mesmos materiais e 4) Novas aplicações desta mesma atividade com apoio de instrumentos de pesquisa.

Quanto ao primeiro tópico, é possível ir em duas direções: Para mais perto ou mais longe do computador. Mais perto, seria usar os endereços de uma forma mais parecida com os endereços da memória principal de um computador. Por exemplo, separar uma quantidade de memória de acordo com o tipo da variável e fazer com que os endereços nos envelopes respeitem estes espaços. Por outro lado, é também seria possível usar endereços de ruas ou casas para trabalhar os conceitos de ponteiros de uma forma semelhante a forma como trabalha a computação desplugada.

Ainda sobre os materiais, também é possível criar arquivos digitais para que o material seja reproduzido por qualquer um que tenha interesse. Para concretizar este passo será preciso estudar uma licença aberta adequada e preparar o material para ser distribuído como um recurso educacional aberto (REA).

Sobre a aplicação em grupos com perfis diferentes, uma primeira possibilidade está em estudantes de ensino médio frequentadores de cursos técnicos integrados ligados as disciplinas de informática e/ou computação. Esta restrição a um grupo específico de secundarista se dá pelo próprio conceito de ponteiros, muito mais útil e proveitoso para quem estuda computação ou informática.

Como a atividade descrita foi aplicada com o objetivo de revisão, também seria interessante ver o seu desempenho ao ser usada como introdução ao conceito de ponteiro da linguagem C.

Por fim, também seria muito rico aplicar novamente esta mesma atividade, mas com apoio de questionários ou outras ferramentas de medição a fim de ter resultados controlados e passíveis de comparação com outras abordagens.

Referências

Adcock, B., Bucci, P., Heym, W. D., Hollingsworth, J. E., Long, T., e Weide, B. W. (2007). Which pointer errors do students make? In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 9–13, New York, NY, USA. ACM.

Anido, R. (2015). Saci - ainda outro ambiente para o ensino de programação. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Aureliano, V. C. O., Tedesco, P. C. d. A. R., e Giraffa, L. M. M. (2016). Desafios e oportunidades aos processos de ensino e de aprendizagem de programação para iniciantes. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Barbosa, W. A., Nunes, I. d. F., Inocêncio, A. C. G., Oliveira, T. B. d., e Júnior, P. A. P. (2015). Deg4trees:um jogo educacional digital de apoio ao ensino de estrutura de dados. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Bucci, P., Long, T. J., e Weide, B. W. (2001). Do we really teach abstraction? *SIGCSE Bull.*, 33(1):26–30.

Craig, M. e Petersen, A. (2016). Student difficulties with pointer concepts in c. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '16, pages 8:1–8:10, New York, NY, USA. ACM.

Da Silva, L. M. P., Bonfim, B. C., Silva, R. C., e da Silva, J. B. (2016). Poogame: Um jogo sériopar ao ensino de programação orientada a objetos. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

França, R. S. d. e Tedesco, P. (2015). Explorando o pensamento computacional no ensino médio: do design à avaliação de jogos digitais. In *Anais do 35º Congresso da*

Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15).

Holanda, A. B. d. (2010). *Mini Dicionario Aurelio Lingua Portuguesa*. Positivo livros.

Lefebvre, H. (1991). *Lógica formal, lógica dialéctica*. Civilização Brasileira, Rio de Janeiro.

Lippert, E. (2018). References are not addresses. disponível em <https://blogs.msdn.microsoft.com/ericlippert/2009/02/17/references-are-not-addresses>. Último acesso em 31 de mar. de 2018.

Machado, N. J. (2011). *Matemática e língua materna*. Cortez: Autores Associados, São Paulo.

Meksenas, P. (1992). As noções de concreto e abstrato: sua relação com as práticas de ensino. *Revista da Faculdade de Educação*, 18(1):92–98.

Miliszewska, I. e Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. In *Issues in Informing Science and Information Technology*.

Milne, I. e Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information Technologies*, 7(1):55–66.

Nunes, M. M., Costa, E. B. B., Feitosa, F. A., e Brancher, J. D. (2015). Análise das ementas de estruturas de dados das universidades brasileiras. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Paparidis, O. S. e Franco, M. E. (2016). Plataforma arduino como apoio ao ensino de programação no curso de técnico em informática integrado. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Raeder, M., Py, M., Rigo, S., e Pinheiro, J. (2016). L2pm: relato de uma experiência sobre o ensino integrado de lógica, programação e matemática para computação. In *Anais do 36º Congresso da Sociedade Brasileira da Computação (CSBC'16), 24º Workshop sobre Educação em Computação (WEI'16)*.

Raiol, A. A. C., Sarger, J., Souza, A., Sivaldo, S., e Fábio, B. (2015). Resgatando a linguagem de programação logo: Uma experiência com calouros no ensino superior. In *Anais do 35º Congresso da Sociedade Brasileira da Computação (CSBC'15), 23º Workshop sobre Educação em Computação (WEI'15)*.

Silva, T. T. d. (2000). A produção social da identidade e da diferença. In da Silva, T. T., editor, *Identidade e diferença: a perspectiva dos estudos culturais*, chapter 2, pages 73–102. Editora Vozes, Rio de Janeiro.

Simon, B., Clancy, M., McCartney, R., Morrison, B., Richards, B., e Sanders, K. (2010). Making sense of data structures exams. In *Proceedings of the Sixth International Workshop on Computing Education Research*, ICER '10, pages 97–106, New York, NY, USA. ACM.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, 49(3):33–35.

Kahoot!: um relato de experiência no contexto acadêmico

Luciana Mara Freitas Diniz[1], **Fischer Ferreira**[2]

[1]Universidade de Itaúna (UIT)
Itaúna – MG – Brazil

[2]Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

lucianamfd@gmail.com,
fischerjf@dcc.ufmg.br

Abstract. *The application of digital media in the educational context presents a different and interactive approach, it may assist the student assessment and learning processes. This review describes an experience that happened in academic environment and had as representative samples brazilians Computer Scienceś students. Its objective was to validate the use of the Kahoot! application as an educational technology. On the occasion, questions about logical and computing reasoning were employed, at graduation level. As a result, the students understood that is an interesting aplication and simulates learning in a different and fun way.*

Resumo. *A utilização de mídias digitais no contexto educacional apresenta uma abordagem diferenciada e interativa, podendo auxiliar nos processos de avaliação e aprendizagem discente. Este estudo descreve uma experiência que se deu em meio acadêmico e teve como amostragem alunos brasileiros de Ciência da Computação. Seu objetivo foi validar o uso da ferramenta Kahoot! enquanto tecnologia educacional. Na ocasião, foram utilizadas questões sobre raciocínio lógico e computacional, a nível de graduação. Como resultados, os alunos perceberam que a ferramenta é interessante e estimula a aprendizagem de forma diferente e divertida.*

1. Introdução

O jogo didático é uma atividade que tem despertado o interesse de alunos e professores para apoiar as atividades realizadas em sala de aula. Os jogos possibilitam a realização de atividades práticas diferenciadas que mantêm um equilíbrio entre o caráter educacional e lúdico das mesmas. Por meio delas é possível relacionar o ensino de conteúdos a uma atividade interativa e enriquecedora de conhecimento, que motiva, revisa conteúdo e possibilita maior interação na sala de aula. Muitos estudos propõem a utilização de mídias digitais, incluindo jogos didáticos, para apoiar o processo de aprendizagem e avaliação em cursos superiores de computação [Petri et al. 2016, Mauricio et al. 2017].

Como exemplo de tais práticas aplicadas à educação, a ferramenta Kahoot! tem sido bastante referenciada em trabalhos dessa natureza [Tobias et al. 2014, Pintor Holguín et al. 2014, Chaiyo and Nokham 2017]. Existem trabalhos que avaliam a eficácia do uso Kahoot! em cursos das áreas tecnológicas em países tais Noruega e

Malásia [Wang 2015, Abidin and Zaman 2017]. Considerando que alguns fatores sociais e geográficos podem afetar a aplicação e a eficiência do uso desta ferramenta, faz-se necessário avaliá-la no contexto dos estudantes brasileiros.

Este relato de experiência foi realizado com alunos do curso de bacharelado em Ciência da Computação da Universidade de Itaúna, localizada no interior do estado de Minas Gerais, Brasil. Buscou-se identificar fatores sobre a percepção que os alunos tiveram ao utilizarem o Kahoot! em contexto acadêmico. Como resultado foi considerado que a ferramenta, para fins de aprendizagem e avaliação em contexto acadêmico tem bom índice de aceitação.

2. Fudamentação Téorica

Na era digital a qual estamos inseridos, os recursos tecnológicos disponíveis em ambientes educacionais proporcionam novas formas de aprendizado, as quais se configuram por meio de softwares, jogos, dentre outros [Tajra 2011].

Segundo [Bates 2016], o que distingue a era digital de todas as anteriores, é o desenvolvimento da tecnologia em ritmo acelerado e a imersão das pessoas nas atividades de cunho tecnológico no cotidiano. Sendo assim é correto pontuar o impacto da internet sobre a educação como uma mudança de paradigma, principalmente em termos de tecnologia educacional.

Ainda de acordo com [Bates 2016], a tecnologia permite que um conjunto de atividades sejam disponibilizadas aos alunos, o que amplia a probabilidade de que uma série de preferências dos alunos esteja sendo atendida e os incentiva a envolverem-se em novas atividades e abordagens de aprendizado. Em torno do ano de 2005, uma nova gama de ferramentas de web começaram a serem usadas de forma geral, e gradativamente uso educacional também. Como exemplo, o autor cita diversas ferramentas, dentre elas, os jogos multiusuários, tal qual se configura a ferramenta Kahoot!.

Neste contexto, a adoção desta ferramenta como tecnologia educacional, apresenta-se como um novo recurso no qual o professor e os alunos, em sala de aula, interagem por intermédio de um jogo *(quiz) online*, usando uma infraestrutura já existente e disponibilizada em ambiente acadêmico [Wang 2015].

2.1. Trabalhos relacionados

Alguns trabalhos com abordagem acadêmica apresentam o uso do Kahoot! para revisão de conteúdos disciplinares e para motivar alunos em certas disciplinas.

Em [Petri et al. 2016], o Kahoot! foi utilizado como um *quiz* com perguntas sobre Gerenciamento de Projetos, disponibilizado para alunos dos cursos de Sistemas de Informação e Ciência da Computação como forma de revisão de conteúdo. Os autores identificaram *feedbacks* positivos nos resultados encontrados. Afirmaram também que o jogo educacional cumpriu com o seu objetivo de aprendizagem por proporcionar a revisão de conhecimentos de forma divertida e motivadora.

Para os autores [Abidin and Zaman 2017], a utilização do Kahoot! foi uma estratégia de motivação para os alunos de uma faculdade de Engenharia Elétrica ao cursarem uma disciplina de programação de computadores. O conteúdo apresentado no game

era relacionado à disciplina em questão e o objetivo era possibilitar aos alunos uma experiência atrativa para a apredizagem, pois muitos deles perdiam o entusiamo ao estudarem programação de computadores. O resultado mostrou que a grande maioria dos alunos conseguiu melhorar sua compreensão sobre a disciplina e também se mostraram mais engajados e motivados ao final do experimento.

Outro trabalho similar [Correia and Santos 2017] apresenta as opiniões de alunos de um programa de formação de professores (licenciatura e mestrado) sobre as vantagens e desvantagens da utilização do Kahoot! em sala de aula. Como vantagens percebidas estão a correção automática das questões e o *feedback* em tempo real para os alunos e professores. Como desvantagens, foram identificados os seguintes fatores: número limitado de caracteres tanto para o enunciado das questões quanto para as opções de respostas, além do tempo limitado de resposta.

3. Configuração do relato de experiência

3.1. Objetivos e questões de pesquisa

O principal objetivo do estudo foi identificar percepções dos alunos sobre o uso da ferramenta Kahoot! como suporte educacional em contexto acadêmico. Objetivos mais específicos consistiram em verificar a aceitação da mesma com base em algumas perspectivas além de elucidar as vantagens e desvantagens da sua utilização.

Com base no método Objetivo-Questão-Métrica (GQM) [Wohlin et al. 2012], esperou-se obter percepções dos alunos a respeito da ferramenta Kahoot! sob quatro perspectivas específicas, a saber: avaliação da aprendizagem, fixação do conhecimento, motivação e conteúdo disciplinar apresentado em formato de jogo.

Para apoiar o estudo proposto, foram elaboradas três questões de pesquisa (QP), com o propósito de sumarizar quantitativa e qualitativamente as respostas dos alunos.

QP1 *- A ferramenta Kahoot! tem boa aceitação para uma atividade de aprendizagem e avaliação em contexto acadêmico?*

QP2 *- Quais são as vantagens e desvantagens da utilização do Kahoot! para apoiar atividades acadêmicas em computação?*

QP3 *- A ferramenta Kahoot! apresenta efeito colateral na percepção dos alunos ao responderem o quiz online?*

3.2. Amostragem e caracterização da experiência

O jogo criado especificamente para este estudo utilizando-se a ferramenta Kahoot! teve o intuito de revisar conteúdos vistos no início do curso, de acordo com a matriz curricular em vigor, relacionados às disciplinas de raciocínio lógico e computacional. Foram selecionados 30 alunos, matriculados nas disciplinas de Algoritmos e Estrutura de Dados III e Programação Orientada a Objetos II da Universidade de Itaúna, ofertadas no 3º e 5º períodos do curso, respectivamente, no primeiro semestre de 2018. A decisão por mesclar participantes de dois períodos distintos teve como finalidade a heterogeneidade do público da amostra.

3.3. Artefatos da experiência

Alguns artefatos foram elaborados para a realização do estudo, a saber:

Jogo sobre raciocínio lógico e computacional: o jogo elaborado [1] continha 20 questões de escolha única, das quais a maioria delas tinha quatro opções de resposta e somente três dessas, duas opções de resposta. As questões criadas no Kahoot! tiveram abordagens genéricas sobre os conteúdos supracitados e não abordaram aspectos específicos de nenhuma linguagem de programação. Os tempos de respostas foram configurados de forma individual para cada questão e variaram de 20 a 120 segundos, de acordo com a complexidade das mesmas. As questões foram elaboradas e classificadas pelos próprios autores em três níveis, a saber: fácil, médio e difícil.

Termo de consentimento: este documento impresso, teve por finalidade explicar aos alunos os objetivos do estudo, garantir a voluntariedade dos mesmos na participação, bem como coletar assinatura deles como forma de autorização para utilizar e publicar os dados obtidos, sem qualquer tipo de identificação, ou seja, de forma sigilosa.

Formulário de retorno (ou feedback): utilizado para obter a opinião dos participantes perante a utilização da ferramenta Kahoot!, isto é, permitiu a obtenção da percepção dos alunos em relação às quatro perspectivas citadas na subseção 3.1. As opções de respostas para as questões de respostas únicas no formulário, foram estabelecidas de acordo com a escala Likert de cinco pontos, considerando os valores de 1 a 5, sendo: 1 - Totalmente favorável, 2 - Favorável, 3 - Neutro, 4 - Desfavorável e 5 - Totalmente desfavorável. Tal formulário foi criado na ferramenta Google Forms, *online* e gratuita, e que faz parte da gama de recursos do Google Drive.

3.4. Passo-a-passo da experiência

A experiência foi dividida em diversos passos, a saber:

Passo 1 - Ambiente de configuração: alunos das duas turmas foram para um laboratório de informática no prédio em que o curso de Ciência da Computação é ofertado. Como a ferramente é *online*, foi necessário o roteamento da rede Wi-fi disponibilizada pela faculdade para conexão com nível de acesso satisfatório dos *smartphones* dos participantes, além dos computadores do laboratório. Por fim, conectou-se um computador ao projetor disponível no local para exibição do *link* do formulário de online, além do PIN do jogo, das questões, placar e *ranking* da ferramenta Kahoot!.

Passo 2 - Apresentação e consentimento: os participantes tiveram acesso ao termo de consentimento livre e esclarecido, onde puderam ler e assinar o documento.

Passo 3 - Demonstração do Kahoot!: foi feita uma demonstração sobre o funcionamento do Kahoot! com os participantes, com intuito deles familiarizarem com a ferramenta. As três questões contidas neste quiz demonstrativo eram bem simples, de conhecimento geral. Assim, os participantes acessaram o site `kahoot.it`, inserindo o PIN específico do jogo e também o *nickname* solicitado para cada um dos alunos. Nesta etapa foram apresentadas as funcionalidades de rankeamento dos participantes por meio da pontuação obtida, exibição das questões utilizando-se um projetor, tempo de resposta das mesmas, exibição das alternativas e alternativas de respostas.

Passo 4 - Execução do jogo: com os participantes devidamente instruídos e conectados ao Kahoot!, foi iniciado o *quiz* específico da experiência, com as questões de

[1] `https://play.kahoot.it/#/?quizId=17a624fc-c400-4af2-95c8-90673bd286c9`

racicínio lógico e computacional, conforme citado anteriormente. O tipo de quiz escolhido foi o clássico (Player vs Player), no qual cada aluno deveria responder de forma individual. O tempo total da aplicação do *quiz* teve duração de aproximadamente 30 minutos.

Passo 5 - Formulário de retorno (ou *feedback*): finalmente, os participantes preencheram o formulário, após a conclusão do jogo. O *link* do mesmo foi disponibilizado e visualizado por todos, por meio do projetor. Cada aluno preencheu o formulário usando *smartphone* próprio ou computador disponvível no laboratório.

4. Resultados e discussões

Nesta seção são apresentados os resultados alcançados da experiência e algumas discussões pertinentes. Ainda, são respondidas as questões de pesquisa levantadas na Seção 3.1.

Faz-se necessário informar que a ferramenta Kahoot! fornece um arquivo com extensão .XLS com todas as informações obtidas no *quiz*. Neste arquivo constam, para cada questão do jogo, o número de respostas por opção, respostas por participante, percentual de acerto, além do tempo de resposta médio e individual.

A Figura 1 ilustra a coleção de dados a respeito do formulário de retorno apresentado na Seção 3.3. Foram coletadas as respostas dos participantes com respeito às seguintes perspectivas já mencionadas: avaliação da aprendizagem, fixação de conhecimento, motivação e conteúdo disciplinar apresentado em formato de jogo.

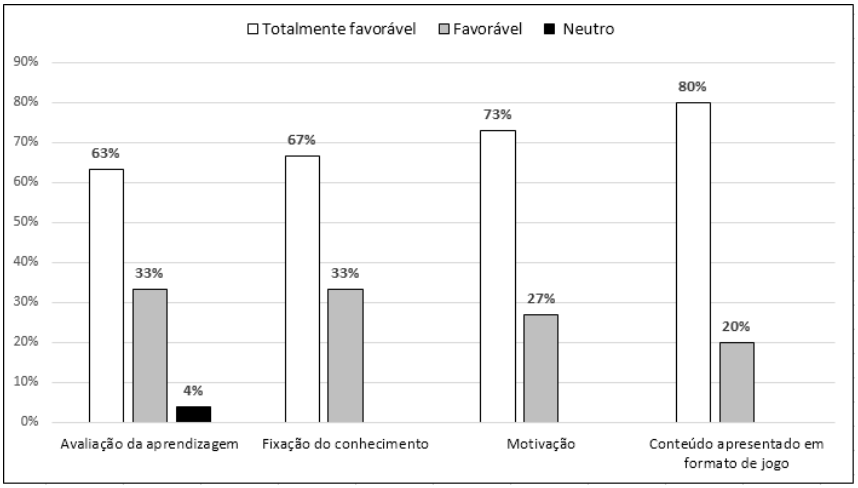


Figura 1. Percepção dos alunos sobre o Kahoot!

Ainda, a Figura 1 demonstra que não foram obtidas respostas para as opções desfavorável e totalmente desfavorável para nenhum dos tópicos de arguição. Entretanto, pode ser observado que foram obtidas um percentual maior de respostas com relação a classificação totalmente favorável. Apenas um percentual de 4% dos participantes disseram ter uma avaliação neutra sobre a perspectiva de avaliação da aprendizagem.

Em resposta à questão de pesquisa QP1: a ferramenta Kahoot! tem boa aceitação para uma atividade de aprendizagem e avaliação em contexto acadêmico? Foi feita uma contagem dos valores obtidos para cada opção de resposta. Mediante os dados obtidos,

foi considerado que a ferramenta, para fins de aprendizagem e avaliação em contexto acadêmico tem bom índice de aceitação.

Considerando as mesmas perspectivas apresentadas anteriormente, para cada questão de resposta única do questionário, havia um campo de justificativa, obrigatório, para a mesma. Cada aluno redigiu a seu modo, resultando em diversos tipos de respostas. Com o intuito de elaborar uma apresentação categórica dos resultados encontrados, as frases de todos os respondentes foram analisadas minuciosamente a fim de identificar características relevantes relacionadas à percepção que tiveram sobre a ferramenta Kahoot!.

Desta maneira, com base em uma análise textual, cada palavra expressiva que remetia à uma característica do Kahoot!, deu origem à uma categoria, ou seja, em uma mesma frase poderiam haver mensão à mais de uma categoria. Portanto, a ocorrência de cada uma destas palavras foi levada em conta ao estabeler sua representatividade nas sete categorias criadas, conforme mostra a Figura 2.

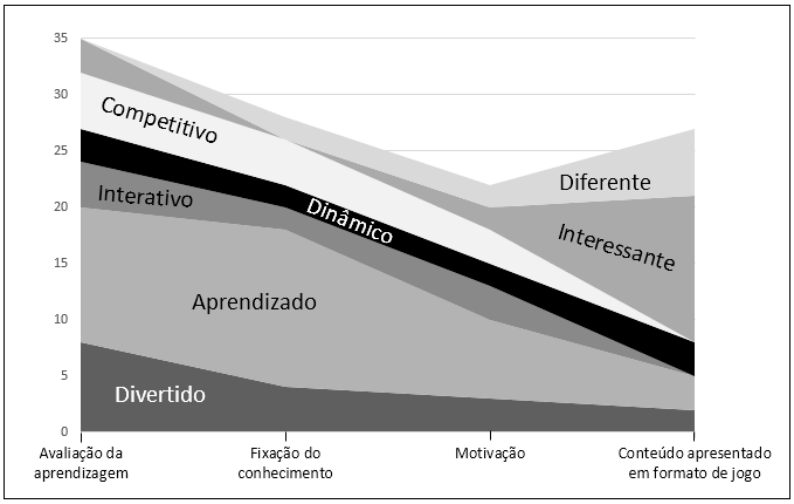


Figura 2. Categorização das percepções dos alunos sobre o Kahoot!

Pode ser inferido ao observar a Figura 2, que a percepção dos alunos sobre a utilização da ferramenta Kahoot! remete ao aprendizado em todas as perspectivas analisadas, sendo mais acentuada sua abrangência na perspectiva de fixação do conhecimento. Outras categorias que tiveram representatividade em todas as perspectivas, foram o caráter dinâmico e divertido da ferramenta, sendo este último, com maior significância na perspectiva de avaliação da aprendizagem. Já as características de interatividade e competitividade da ferramenta Kahoot! só não foram percebidas na perspectiva de conteúdo disciplinar apresentado em formato de jogo. Porém, o caráter interessante e diferente do Kahoot! foi visto com maior expressividade exatamente nesta última perspectiva, conforme a opinião dos alunos.

Além disso, mais três categorias foram identificas, mas por não terem muita significância (pequeno número de ocorrências), não foram exibidas no gráfico. Tais categorias remetem ao caráter de motivação, revisão de conteúdo e raciocínio rápido com que a ferramenta foi percebida pelos alunos. Vale ressaltar também que respostas contendo somente as palavras “bom”, “muito bom”, “excelente”, “ótimo”, as negativas “não” e “nunca” e as afirmativas “sim” ou “claro”, não foram consideradas na categorização.

Para responder a questão de pesquisa 2 (QP2) e evidenciar o que de mais importante a experiência de uso da ferramenta representou para os alunos, em termos de vantagens e desvantagens, algumas das respostas dos mesmos serão transcritas a seguir.

Na perspectiva de avaliação da aprendizagem, a maioria dos alunos percebeu a competitividade como uma vantagem, de modo a estimular os estudos e o aprendizado em si, como percebe-se em: "Competindo aprende mais" (Aluno 01) e "Uma plataforma interativa e competitiva sempre influencia os alunos a estudarem" (Aluno 02). Outro aluno descreveu sua percepção positiva da seguinte forma: "Método interativo e divertido de aprendizagem" (Aluno 03). Entretanto, um aluno demonstrou uma desvantagem neste ponto, ao afirmar que "O uso da ferramenta é bom, mas não serve como critério avaliativo, pois é ambiente competitivo" (Aluno 04).

Já sobre a perspectiva de fixação do conhecimento, as respostas dos alunos demonstraram, em sua totalidade, somente vantagens, evidenciando que a ferramenta reforça o aprendizado, além do seu caráter divertido: "Forma lúdica, literalmente aprender brincando" (Aluno 05); "Aprende e se diverte" (Aluno 06); "Uma plataforma interativa e competitiva sempre influencia os alunos a estudarem, o que é o principal para o aprendizado" (Aluno 07).

Em relação à perspectiva motivacional a ser verificada, os alunos perceberam, além do caráter de aprendizagem proporcionado pela ferramenta, também sua dinamicidade, interatividade e competitividade: "Uma plataforma interativa e competitiva sempre influencia os alunos a estudarem" (Aluno 08); "Jeito fácil, interativo e dinâmico de aprender" (Aluno 09); "Um jogo dinâmico que estimula a aprendizagem" (Aluno 10). Não foram identificadas desvantagens neste aspecto.

E, por fim, na perspectiva de conteúdo apresentado em formato de jogo, os alunos também só perceberam vantagens, evidenciando as categorias diferente e interessante: "Foi muito interessante e divertido" (Aluno 11); "É interessante ver algo que aprendemos em sala de aula em um jogo" (Aluno 12); "Bacana fazer algo de diferente, jogando e aprendendo" (Aluno 13).

Com mais resultados a serem expostos, a Figura 3 apresenta a distribuição das questões utilizadas no *quiz* do Kahoot! em três níveis de classificação: fácil, médio e difícil, a fim de construir grupos distintos em relação à complexidade das mesmas. Deste modo, por meio dos dados obtidos, foi possível ter uma indicação se a ferramenta apresenta um efeito colateral, ou seja, se o Kahoot! possui elementos na sua interface que podem comprometer a qualidade de uma atividade utilizando este recurso tecnológico.

Para responder a questão de pesquisa 3, se a ferramenta Kahoot! apresenta efeito colateral na percepção dos alunos ao responderem o *quiz online*, foi observado qual era a distribuição dos acertos dos alunos para as questões propostas. A premissa seguida para realização dessa análise tem por base que, em um processo de avaliação, o maior índice de acertos se concentra em questões mais fáceis, ficando o índice menor de acertos para questões mais complexas. Sendo assim, buscou-se uma análise descritiva da concentração de acertos com base nas respostas dos alunos. É importante frisar que o objetivo do estudo não foi medir o conhecimento dos alunos com base nos temas abordados.

Ademais, por meio da Figura 3 pode-se observar no gráfico *boxplot* que a mediana de acerto para questões consideradas fáceis, ficou próxima dos 90%; já questões de

complexidade média e difícil tiveram taxa de acerto acima dos 40%. Como pode ser observado ainda na Figura 3, o percentual de acertos para questões fáceis é muito maior que questões consideradas médias e difíceis, ficando a concentração do percentual de acertos para questões fáceis acima dos 75% e o percentual para questões médias e difíceis abaixo dos 50%.

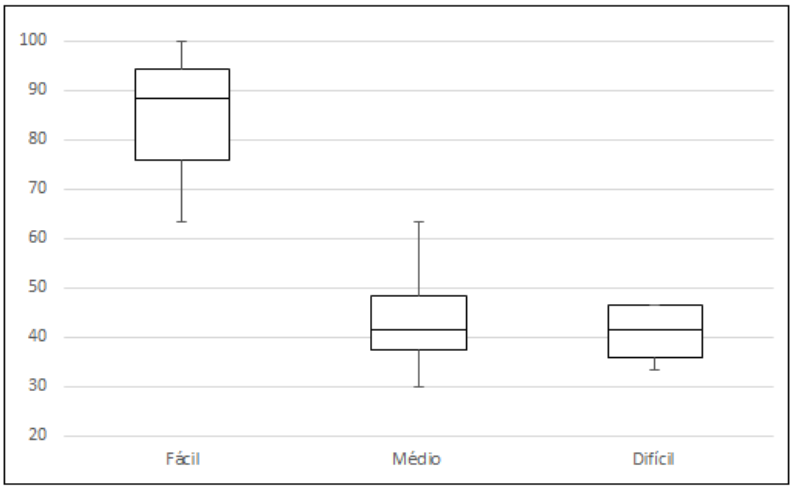


Figura 3. Acertos das questões com uso do Kahoot!

Com base na análise anteriormente feita e sobre as condições particulares do estudo proposto, pode-se afirmar que existe um forte indício que a ferramenta Kahoot! não acrescenta efeito colateral ao processo de avaliação por meio de questionário apresentado em formato de quiz online.

4.1. Experiência do usuário

Nesta seção são relatadas as concepções com uso do Kahoot! do ponto de vista dos docentes e discentes, isto é, os aspectos positivos e negativos que foram observados durante a configuração e realização do relato de experiência.

As principais vantagens encontradas foram: a ferramenta é gratuita, de fácil manipulação para criação dos questionários, intuitiva, permite a inclusão de mídias como vídeos e imagens nas questões, recuperação e apresentação dos resultados em planilha de maneira muito bem estruturada, facilidade de compartilhamento e de uso, com uma interface gráfica bastante amigável.

Algumas desvantagens também foram identificadas, tais como: o número de caracteres para a exibição das perguntas é limitado e pequeno; durante o jogo, ao receber alguma notificação no celular, o aluno era automaticamente desconectado da ferramenta; a regulagem do tempo para a resposta de algumas questões foi insuficiente para que todos os alunos conseguissem ler, analisar e responder; em alguns casos é difícil estabelecer o tempo para a resposta e, finalmente, alguns alunos reportaram problemas de usabilidade da ferramenta como, por exemplo, quando havia somente 2 opções de respostas, a alternativa com conotação positiva estava associada à cor vermelha e a alternativa com conotação negativa, associada à cor azul. Este é o padrão seguido pela ferramenta em que a atribuição das cores em relação às opções de respostas é feita de forma padronizada e automática.

5. Ameaças à validade

A aplicação do estudo foi cuidadosamente realizada e projetada conforme descrito na Seção 3. Foi uma preocupação disponiblizar equipamentos necessários para realização das atividades relacionadas ao Kahoot! bem como o prenchimento do formulário online pertencente ao estudo. Porém, apesar de todo cuidado, existem algumas ameaças que podem afetar os resultados alcançados. Será apresentada nesta seção a discussão sobre cada um dos quatro tipos de ameaças à validade do relato de experiência segundo [Wohlin et al. 2012]: construção, interna, conclusão e externa. Ainda, em cada uma, será apresentado o respectivo tratamento.

Validade de construção: experimentos pilotos foram realizados em três etapas com intuito de encontrar a melhor equalização nos seguintes tópicos referentes às questões do jogo: tempo de resposta de cada pergunta, o número de perguntas para o tempo total previsto do experimento (uma hora de duração) e ainda se as questões poderiam ser respondidas adequadamente pelos alunos. O formulário de *feedback* foi elaborado com o intuito de levantar a opinião dos participantes sem o mínimo possível de interferências, como por exemplo, o participante marcar qualquer opção apenas para cumprir um requisito. Por isso, em cada questão, foi colocado um campo onde o participante precisava descrever sua opinião sobre a aquela questão. Ainda, foi escolhido a lógica de programação como tema principal da atividade com intuito de ser um tema de conhecimento de todos os participantes da atividade realizada.

Validade interna: foi preparando um pequeno treinamento utilizando o Kahoot!. Esse treinamento foi feito com intuito de evitar que o desconhecimento da ferramenta por parte dos alunos pudesse ofuscar a percepção dos mesmos quanto à real contribuição da ferramenta para apoiar atividades acadêmicas. Para minimizar esse impacto foi estipulado que os participantes seriam recompensados com pontuações na nota do curso, e os primeiros lugares na classificação dada pelo Kahoot! ganharam uma pontuação maior.

Validade de conclusão: Foi realizada uma análise cautelosa dos dados obtidos com o experimento com a finalidade de minimizar os problemas com relação à interpretação dos dados. Para isso, os dados encontrados foram submetidos a vários gráficos e tabelas. Ainda, foi observado os trabalhos relacionados para garantir os dados alcançados fossem úteis para corroborar com conclusão do trabalho.

Validade externa: Alguns fatores podem impedir a generalização dos resultados alcançados nesse experiência. Por exemplo, o número de participantes e o fato de ter sido feita em apenas uma instituição de ensino superior. Esses participantes podem não representar adequadamente todos os estudantes de computação, considerando sua quantidade, histórico, aspectos culturais, experiência profissional e outros. No entanto, os participantes disponíveis para o experimento foram selecionados aleatoriamente com intuito de minimizar esse problema e diversificar a representatividade dos estudantes desta área.

6. Conclusão e trabalhos futuros

A experiência sobre os aspectos concernentes da ferramenta Kahoot! em ambiente acadêmico, apresentou diversos resultados sobre suas características, que foram sumarizados e analisados perante as percepções dos alunos e da metodologia proposta. Por meio do formulário de feedback, foi possível realizar análises quantitativas e qualitativas das respostas obtidas dos alunos.

Os resultados foram bastantes satisfatórios no que diz respeito à aceitação da ferramenta no contexto educacional, pelo seu caráter divertido, interessante e dinâmico os quais remetem à aprendizagem, conforme a percepção dos alunos. Também foi verificado que a sua utilização não causou efeito colateral nas respostas dos participantes.

Como trabalho futuro se faz necessário um experimento para verificar a evidência da qual foi apontada no presente trabalho que o Kahoot! não acrescenta efeito colateral nas respostas dos alunos ao responderem o *quiz online*.

7. Agradecimento

Esta pesquisa foi parcialmente patrocinada pela agência de fomento brasileira CNPq.

Referências

Abidin, H. Z. and Zaman, F. K. (2017). Students' perceptions on game-based classroom response system in a computer programming course. In *Engineering Education (ICEED), 2017 IEEE 9th International Conference on*, pages 254–259. IEEE.

Bates, T. (2016). *Educar na era digital: desing, ensino e aprendizagem*. São Paulo: Artesanato Educacional.

Chaiyo, Y. and Nokham, R. (2017). The effect of Kahoot, Quizizz and Google Forms on the student's perception in the classrooms response system. In *Digital Arts, Media and Technology (ICDAMT), International Conference on*, pages 178–182. IEEE.

Correia, M. D. L. and Santos, R. (2017). Game-based learning: The use of Kahoot in teacher education. *2017 International Symposium on Computers in Education (SIIE)*, pages 1–4.

Mauricio, R. d. A., Veado, L., Moreira, R. T., Figueiredo, E., and Costa, H. (2017). A Systematic Mapping Study on Game-related Methods for Software Engineering Education. *Information and Software Technology*.

Petri, G., Battistella, P. E., Von Wangenheim, C. G., Cassettari, F. T., and Hauck, J. C. R. (2016). Um Quiz Game para a Revisão de Conhecimentos em Gerenciamento de Projetos. In *Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016)*, pages 320–329, Uberlândia, Brasil.

Pintor Holguín, E., Gargantilla Madera, P., Herreros Ruiz Valdepeñas, B., López del Hierro Casado, M., et al. (2014). Kahoot en docencia: una alternativa practica a los clickers.

Tajra, S. F. (2011). *Informática na educação: novas ferramentas pedagógicas para o professor na atualidade*. São Paulo: Érica.

Tobias, S., Fletcher, J. D., and Wind, A. P. (2014). Game-based learning. In *Handbook of research on educational communications and technology*, pages 485–503. Springer.

Wang, A. I. (2015). The wear out effect of a game-based student response system. pages 217–227. Computers & Education.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Uso de Realidade Aumentada para Ensino de Arquitetura de Computadores com MIPS

Geofrangite Câmara da Silva[1], Leiva Casemiro Oliveira[1], Sílvio Roberto Fernandes[1]

[1]Programa de Pós-Graduação em Ciência da Computação (PPgCC), Departamento de Computação, Universidade Federal Rural do Semi-Árido (UFERSA) – Mossoró, RN - Brasil

gcscamara@gmail.com, {leiva.casemiro, silvio}@ufersa.edu.br

Abstract. *This paper presents a simulator under development to assist the MIPS processor architecture learning process. The aim of the tool is provider it higher integration with the standard teaching material, enhancing learning activity. For this, augmented reality is used to recognize figures from the book "Computer Organization and Design, 4ed". For the development of the tool was used Unity and Vuforia. The present results are experimental, obtained in the laboratory, which allowed the recognition of 3 figures of the book. These figures present, in sequence, more and more details of the architecture and different learning purposes.*

Resumo. *Este artigo apresenta um simulador em desenvolvimento para auxiliar no aprendizado da arquitetura do processador MIPS. A ferramenta tem como objetivo permitir maior integração entre o material de estudo e as ferramentas de apoio ao aprendizado. Para isso faz uso da realidade aumentada para reconhecer as figuras do livro "Organização e Projeto de Computadores de Patterson e Hennessy, 4ed.". Para o desenvolvimento da ferramenta foi utilizado Unity e Vuforia. Os resultados atuais são experimentais, obtidos em laboratório, os quais permitiram reconhecimento de 3 figuras do livro. Essas figuras apresentam, em sequência, cada vez mais detalhes da arquitetura e propósitos de aprendizado diferentes.*

1. Introdução

Os cursos de graduação em Ciência da Computação, em geral, possuem grades curriculares compostas de disciplinas que englobam diversas subáreas da computação, desde o *hardware* até o *softwar*e. Dessas disciplinas, as que abordam o *hardware*, como as focadas no ensino de arquitetura de computadores, são tidas como mais complexas em termos de aprendizado. Devido a essa complexidade, tais disciplinas são responsáveis por grande parte da retenção e evasão dos alunos.

Nesse sentido, é comum a adoção de metodologias que recorrem a tecnologias para facilitar o aprendizado. Os simuladores, ferramentas que diminuem o nível de abstração dos conteúdos, são uma tecnologia a qual se recorre muito. Outro exemplo de tecnologia pode ser a Realidade Aumentada (RA) que, pela forma de utilização interativa e rica em elementos visuais, tem sido bastante aplicada no campo da educação.

No contexto do ensino de arquitetura de computadores, vários simuladores foram desenvolvidos. Como, em geral, os cursos baseiam-se na aprendizagem de arquiteturas padronizadas tal como o MIPS (*Microprocessor without interlocked*

pipeline stages) para facilitar o aprendizado dos relacionamentos entre os componentes de um computador, alguns dos simuladores mais populares são o MARS [Vollmar e Sanderson, 2005], WEBMIPS [Branovic, Giorgi e Matinelli, 2004], DIMIPSS [Felix, Pousa e Carvalho, 2006] e VISIMIPS [Kabir, Bari e Haque, 2011]. Embora esses simuladores proporcionem ao aluno uma noção desses relacionamentos, não há uma integração entre eles e o material de estudo (livro) de forma que o aluno não precise se "desligar" de um para utilizar o outro.

A tecnologia RA consiste na sobreposição em tempo real de elementos virtuais no mundo real através do uso de um dispositivo tecnológico que permite a manipulação e visualização desses elementos [Kirner e Siscoutto, 2007]. Uma revisão sistemática conduzida por [Akçcayir e Akçcayir, 2017] mostrou que a quantidade de trabalhos que aplicam RA na educação tem aumentado significativamente nos últimos anos e que ela é aplicável aos diversos níveis de escolaridade, desde a alfabetização até o nível superior. Nessa revisão os autores também identificaram várias vantagens da aplicação da RA na educação, dentre elas: melhora do aprendizado; facilitação do entendimento; aumento da motivação para aprender, do nível de engajamento, do interesse e da satisfação dos alunos; promoção do auto-aprendizado, do aprendizado multissensorial e do aprender fazendo; visualização de conceitos invisíveis, eventos e conceitos abstratos; redução do custo do material de laboratório. No entanto, há poucos trabalhos que aplicam RA para o ensino da computação em especial voltados para o ensino de arquitetura de computadores. Além disso, desconhecemos, até a presente data, a existência de aplicações que utilizam RA no ensino do MIPS e que sejam integradas com materiais didáticos já consolidados.

Esse trabalho apresenta um simulador para dispositivos móveis que está sendo desenvolvido para auxiliar no ensino de arquitetura de computadores. O simulador faz uso da RA para reconhecer algumas das figuras do livro "Organização e Projeto de Computadores" [Patterson e Hennessy, 2004], de Patterson e Hennessy – os ganhadores do Prêmio Turing 2017[1]. O livro é uma das referências mais utilizadas no ensino de arquitetura de computadores, apresenta de forma didática o caminho de dados do MIPS, o que explica a escolha de suas figuras como marcadores para o simulador. A medida que estuda pelo livro, o aluno pode utilizar o simulador para fazer o reconhecimento das figuras e interagir com os modelos 3D presentes na RA para facilitar o aprendizado por meio da obtenção de informações e visualização do funcionamento interno do MIPS.

O restante do trabalho está organizado como segue: a seção 2 apresenta os trabalhos relacionados; a seção 3 detalha a ferramenta proposta descrevendo seu projeto e implementação; a seção 4 mostra os resultados preliminares; e por fim, a seção 5 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nessa seção são apresentados alguns trabalhos relacionados que demonstram o funcionamento da linguagem *assembly* no MIPS.

[1] https://amturing.acm.org/byyear.cfm

2.1. MARS

O MARS (*Mips Assembly and Runtime Simulator*) é um simulador desenvolvido em JAVA que simula parte do conjunto de instruções do MIPS32. O MARS disponibiliza um editor de texto e um montador MIPS. Dessa forma, para verificar a execução de uma sequência de instruções, o usuário deve fornecer um código *assembly* que terá sua sintaxe verificada pelo montador e, caso esteja correto, será permitida a execução da etapa de simulação. Após a montagem, é possível modificar os valores da memória de dados e dos registradores e, em seguida, realizar a simulação onde o usuário pode escolher executar todas as instruções de uma vez ou a quantidade de instruções executas por segundo ou ainda o modo passo-a-passo, em que a transição acontece apenas ao clique do usuário. Além disso, existem as opções de pausar, retroceder e parar a execução. Uma extensão [Araújo, Pádua e Corrêa Junior, 2014] disponível para as versões mais atuais, a partir da 4.5, possibilita ainda, a visualização do caminho de dados mostrando, por meio da alteração das cores dos barramentos, a sequência em que os dados são transferidos de um componente para outro. Contudo, essa extensão não exibe os valores desses dados.

2.2. WEBMIPS

O WEBMIPS é um ambiente Web de simulação do MIPS que tem como vantagens a execução multiusuário e o fato do usuário não precisar instalar qualquer programa ou extensão para acessá-lo. Seu principal objetivo é a criação da simulação dos cinco estágios passo a passo do funcionamento do *pipeline*. Para tanto, o WEBMIPS disponibiliza um editor e um montador que suporta parte do conjunto de instruções do MIPS e permitem que o usuário insira um código e opte por executá-lo etapa por etapa ou todas as etapas de uma vez. Ao finalizar a execução do código a quantidade de ciclos de *clock* necessários é exibida. Embora as modificações de valores que ocorrem nos caminhos de dados e controle durante a execução das instruções não sejam apresentadas diretamente, é possível visualizar os valores de entrada e de saída dos componentes e os estados dos sinais, no caso da unidade de controle, clicando sobre eles.

2.3. DIMIPSS

O DIMIPSS (*Didact Interactive MIPS Simulator*) é um software multiplataforma de simulação da execução das instruções do MIPS Monociclo. Ele recebe um programa em linguagem de montagem (*assembly*) e o converte para a linguagem de máquina. Após a conversão é possível simular as instruções do código uma por uma visualizando as alterações (destacadas em cinza) feitas na memória de dados, na memória de instruções, no contador de programa e no banco de registradores. Para facilitar o entendimento, os barramentos que possuem sinais usados durante a execução da instrução também são destacados para cada instrução através das cores azul (para o caminho de dados) e vermelho (para o caminho de controle). Semelhante ao WEBMIPS, é possível visualizar os valores de entrada e de saída dos componentes e os estados dos sinais da unidade de controle bastando, para tanto, passar o mouse sobre eles.

2.4. VISIMIPS

O VISIMIPS é um simulador multiplataforma voltado para a simulação dos cinco estágios do funcionamento do pipeline do MIPS32. Ele contém um montador que traduz

as instruções do MIPS32 em código de máquina. Na simulação é possível avançar/retroceder a execução de uma instrução por vez. Também é possível visualizar os valores contidos nos barramentos dos caminhos de dados e de controle passando o *mouse* sobre eles.

2.5. DRMIPS

O DRMIPS [Nova, Araújo e Ferreira, 2013] é um simulador que permite tanto a simulação do MIPS monociclo quanto *pipeline*. Ele contém um montador que traduz as instruções do MIPS em código de máquina. Após a montagem o usuário pode modificar os valores da memória de dados e dos registradores e então, realizar a simulação executando todas as instruções de uma vez ou uma por uma. Na simulação os registradores e posições da memória ativos na execução de cada instrução são destacados de forma colorida. O simulador também mostra o caminho de dados graficamente onde é possível visualizar os valores contidos nos barramentos dos caminhos de dados e de controle. Além da versão multiplataforma para PCs ele apresenta uma versão para dispositivos móveis com Sistema Operacional (SO) Android. Contudo, os recursos de visualização e usabilidade tornam o uso desse simulador desmotivante. A falta de integração com um material de apoio e de recursos para avaliação dos alunos, deixam-no limitado como ferramenta didática.

3. Trabalho Proposto

3.1. Projeto

O simulador denominado de ARMS (Augmented Reality MIPS Simulator) foi pensado para ser usado em conjunto com o livro para auxiliar no aprendizado dos conteúdos nele abordado. Como o livro aborda um subconjunto de instruções do processador MIPS e apresenta seu caminho de dados de forma incremental, por meio de figuras, a ideia é que o aluno possa usar o simulador para visualizar a execução das instruções suportadas pelo caminho de dados representado em cada figura do livro, em cada fase da construção do caminho de dados. Assim, o ARMS está sendo desenvolvido como um aplicativo voltado para dispositivos móveis que rodam o SO Android. Tal aplicativo faz uso da câmera do dispositivo para reconhecer as figuras do livro e criar uma RA que usa modelos 3D dos caminhos de dados com os quais é possível interagir.

3.2. Ferramentas usada para o desenvolvimento

Para a implementação, estão sendo usados as ferramentas Unity 2017.2.0[2], Vuforia SDK (*software development kit*) para Unity[3], Tinkercad[4] e Android SDK[5]. Unity é um *game engine* para o desenvolvimento de jogos e aplicativos de visualização 3D que possibilita fácil exportação para PC (Windows, Mac, Linux), Android, IOS, UWP, consoles e outros. Vuforia é um SDK para desenvolvimento de aplicações RA voltadas para as

[2] https://unity3d.com/pt/

[3] https://www.vuforia.com/

[4] https://www.tinkercad.com/

[5] https://developer.android.com/index.html

plataformas Android, iOS, UWP e óculos digitais. Ela pode ser integrada ao Android Studio, Xcode, Visual Studio e Unity. Tinkercad é ambiente Web que proporciona a criação fácil e intuitiva de objetos 3D bem como sua exportação nos formatos .obj e .stl. O Android SDK é o kit de desenvolvimento de aplicações Android e é requerido pelo Unity para a criação de aplicações para Android.

3.3. Implementação

O primeiro passo da implementação foi transformar as figuras do livro em marcadores. A transformação foi feita convertendo-as em *Image Target,* um dos tipos de *Target* (alvo) que pode ser criado no Vuforia e que possibilita transformar uma imagem qualquer em um marcador. Quando uma imagem é transformada ela passa por uma análise que indica o quão reconhecível ela é. Isso é indicado através da quantidade de estrelas que ela recebe. Essa quantidade varia entre 0 e 5, sendo que quanto mais estrelas uma imagem recebe mais fácil seu reconhecimento. Após adicionar todas as figuras, foi feito o download do banco de dados e sua importação para o Unity.

O próximo passo foi criar os modelos 3D dos componentes do processador. Embora no Unity seja possível criar modelos 3D de algumas formas geométricas como cubo, esferas, cilindros e capsulas, não é possível remodelá-los. Assim, apenas os componentes que podem ser representados por estas formas foram criados no Unity. Os demais foram modelados no Tinkercad e exportados no formado .obj para o Unity.

Com os componentes modelados e as figuras transformadas em marcadores, o modelo 3D de cada figura foi gerado associando os modelos dos componentes aos marcadores. Em seguida, foram inseridos os elementos de GUI (*Graphical User Interface*) e, por fim, foram criados scripts na linguagem C# para definir o comportamento dos componentes 3D e dos elementos de GUI.

4. Resultados Preliminares

Até o momento, foram criados modelos 3D para algumas figuras do livro, na versão em português, e algumas funções foram implementadas para essas figuras. Uma dessas funções permite obter informações dos componentes do processador clicando sobre eles. A outra permite selecionar um dos tipos de instruções já implementados, opcionalmente definir suas configurações de execução (registradores envolvidos, seus valores e valores da memória de dados) e executá-lo, navegando entre suas etapas.

Quando executa o aplicativo é exibida uma tela na qual o usuário deve selecionar qual figura deseja reconhecer. Feito isso, a câmera e o *flash* (para evitar problemas de reconhecimento devido iluminação) do dispositivo são ativados e devem ser apontados para a figura do livro para que a mesma seja reconhecida. Quando a figura é reconhecida o modelo da arquitetura é gerado sobre ela juntamente com os elementos GUI que permitem configurar e controlar a execução de instruções, bem como a visualização dos valores contidos nos barramentos a cada etapa da execução. Caso a figura não seja reconhecida, devido problemas na qualidade da impressão do livro, o simulador permite o carregamento automático dos modelos 3D após um tempo (10 segundos) tentando fazer o reconhecimento.

Um exemplo do funcionamento da RA é mostrado na Figura 1, nela é mostrado o reconhecimento da figura 5.1 do livro que mostra o primeiro modelo incremental do

caminho de dados do MIPS. Quando o ARMS identifica a figura do livro, possibilita ao usuário obter informações sobre os componentes básicos (PC, memória de instruções, banco de registradores, somadores, ALU e memória de dados) do processador MIPS. Para tanto, o usuário pode clicar sobre o modelo 3D gerado de cada um deles, o qual será destacado e informações textuais serão exibidas (em um *pop-up*) sobre a função do componente clicado. Para que o *pop-up* desapareça basta clicar novamente no componente ou em algum outro componente.

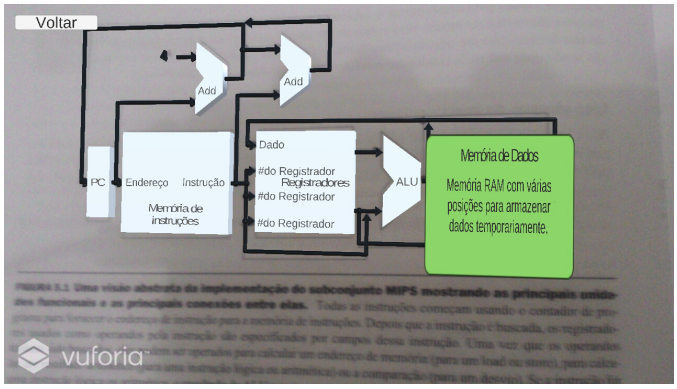


Figura 1. Reconhecimento da figura 5.1 do livro pelo simulador. Destaque para o modelo 3D gerado pela RA

Um outro marcador para a RA é a figura 5.2 do livro. Com ela reconhecida, o usuário poderá fazer a simulação da execução de instruções. Para mostrar como a simulação funciona executamos o programa contido na Figura 2, em que dois valores são lidos da memória de dados, somados e o resultado da soma é escrito na memória de dados.

```
lw t0, 4(t1)
lw t1, 8(t1)
add t2, t0, t1
sw t2, 0(t3)
```

Figura 2. Código *assembly* para testar a RA.

O primeiro passo para executar o código foi salvar os valores na memória de dados. Para isso, com o marcador reconhecido, a instrução *load* foi selecionada fazendo aparecer sobre os componentes a tela exibida na Figura 3(a). Essa tela permite selecionar os registradores envolvidos na execução da instrução, definir seus respectivos valores e os valores da memória de dados e outros valores (*offset* e *immediat*e) quando necessários (A inserção de todos esses valores é opcional, e no caso do usuário não informar os valores padrões são zero). Assim, com a posição 8 da memória de dados selecionada foi inserido o valor 14 e clicado no botão "Salvar" para que o dado fosse salvo. De forma semelhante, o valor 5 foi salvo na posição 12. Com os valores salvos, o registrador t1 foi escolhido como rs e recebeu o valor 4, o registrador t0 como rt e o *offset* recebeu 4. Para salvar os valores dos registradores selecionados e demais valores foi clicado no botão "Definir Valores" (com isso a tela para definir valores também é ocultada).

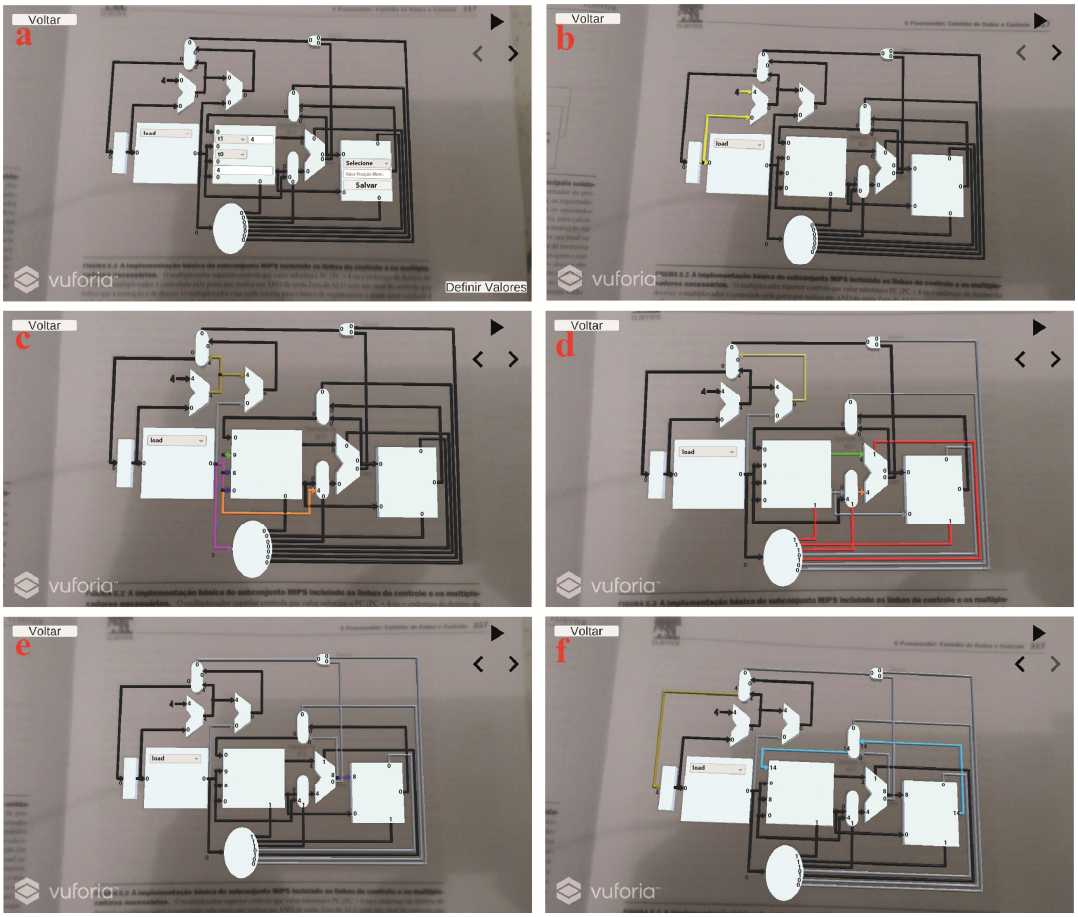


Figura 3. Sequência de etapas realizadas na execução da instrução *load*. (a) Registrador t1 escolhido como rs, registrador t0 escolhido como rt e valor 4 atribuído ao *offset*; (b) O PC (Program Counter) repassa o endereço da instrução a ser executada para a Memória de Instruções e para o Somador; (c) O Somador soma 4 ao endereço vindo do PC e a instrução é decodificada; (d) O endereço do dado a ser lido é calculado pela ALU (Arithmetic Logic Unit) somando o *offset* ao valor de t1; (e) O resultado da ALU é repassado para a Memória de Dados; (f) O dado contido no endereço repassado pela ALU é escrito em t0 e o PC é incrementado em 4.

Com os valores definidos, é possível visualizar as etapas da execução da instrução no modo automático ou no passo a passo. No modo automático, ativado pelo botão "Play", a cada 2 segundos uma etapa da execução da instrução é exibida até que a execução termine. Quando o botão "Play" é clicado também é exibido um controle que possibilita alterar a velocidade de execução, selecionando um valor de 1 (mais lento) a 5 (mais rápido). No modo passo a passo os botões "Anterior" e "Próximo" permitem navegar entre as etapas da execução. Vale salientar que os passos ou etapas da execução não representam ciclos, uma vez que essa organização é monociclo. Tais etapas representam os atrasos dos circuitos combinacionais e tem intuito didático, para facilitar o entendimento do funcionamento do caminho de dados pelos estudantes.

Em cada etapa da execução é possível visualizar a atualização dos valores que saem dos componentes ativos nessa etapa. Além disso, os barramentos ativos nos caminhos de dados e de controle são destacados de forma colorida. Os barramentos do caminho de dados que não são necessários para a instrução em execução são coloridos

com cinza. Os demais são coloridos com cores diversificadas sendo que barramentos que tem os mesmos dados são coloridos com a mesma cor e os que apresentam valores diferentes são coloridos com cores diferentes. No caminho de controle, os barramentos ativos (valor lógico 1) são coloridos em vermelho e os inativos (valor lógico 0) em cinza. Como forma de destacar os barramentos ativos em cada etapa, todos os barramentos que foram coloridos na etapa anterior, exceto os coloridos em cinza, voltam a ser preto na etapa atual.

Continuando a execução do código, a segunda instrução *load* foi executada de forma semelhante a primeira para trazer o valor 5 para o registrador t1. Com os valores nos registradores, foi executada a instrução *add* como seu resultado sendo escrito em t2, como mostra a Figura 4.

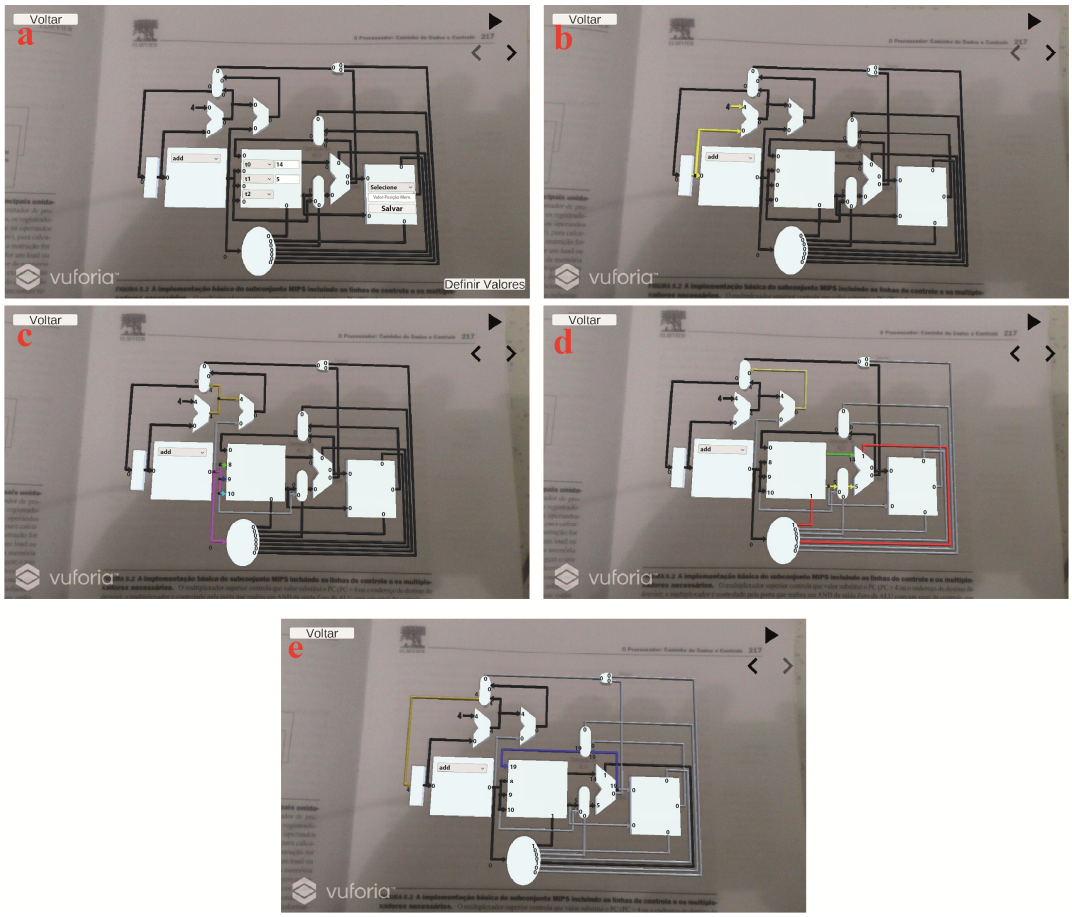


Figura 4. Sequência de etapas realizadas na execução da instrução *add*. (a) Registrador t0 escolhido como rs, registrador t1 escolhido como rt e registrador t2 escolhido como rd; (b) O PC repassa o endereço da instrução a ser executada para a Memória de Instruções e para o Somador; (c) O Somador soma 4 ao endereço vindo do PC e a instrução é decodificada; (d) Os valores contidos em t0 e t1 são somados pela ALU; (e) O resultado da ALU é escrito em t2 e o PC é incrementado em 4.

Por fim, através das etapas mostradas na Figura 5, a instrução *store* armazena o valor contido em t2 na posição 16 da memória de dados.

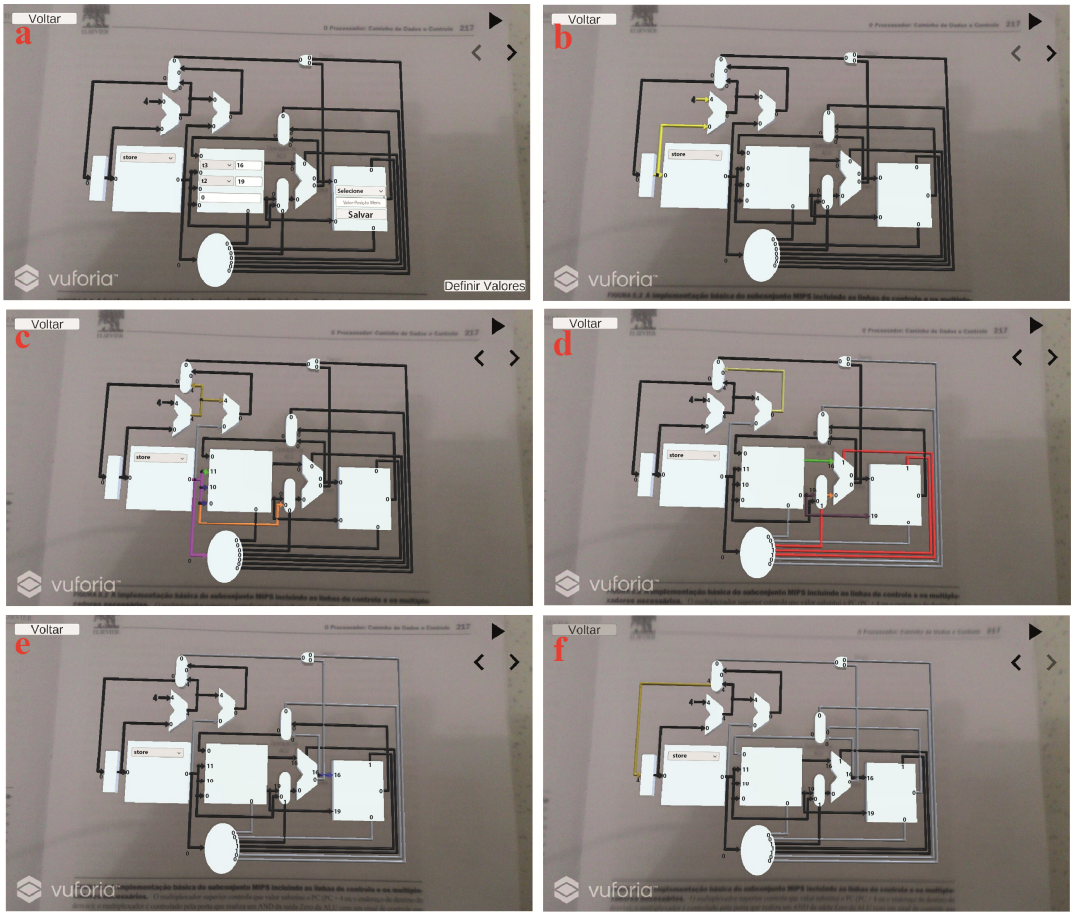


Figura 5. Sequência de etapas realizadas na execução da instrução *store*. (a) Registrador t3 escolhido como rs, registrador t2 escolhido como rt e valor 0 atribuído ao *offset*; (b) O PC repassa o endereço da instrução a ser executada para a Memória de Instruções e para o Somador; (c) O Somador soma 4 ao endereço vindo do PC e a instrução é decodificada; (d) O endereço do dado a ser escrito é calculado pela ALU somando o *offset* ao valor de t3; (e) O endereço calculado pela ALU é repassado para a Memória de Dados e o dado contido em t2 é escrito nele; (f) O PC é incrementado em 4.

Assim como o livro [Patterson e Hennessy, 2004], que apresenta as figuras da arquitetura de forma incremental, cada vez com mais detalhes e compatibilidade com mais instruções, o objetivo final do ARMS é que o reconhecimento das figuras seja feita em sequência. Isso porque as figuras têm objetivos específicos. A primeira reconhecida (Figura 1) objetiva apresentar a função de cada componente e a segunda (Figuras 3, 4 e 5) para fazer uso de registradores, memória e instruções simples. Dessa forma, a ferramenta cria um fluxo para sua utilização como recurso didático, de modo que os recursos mais avançados serão disponibilizados a medida que as etapas anteriores são cumpridas pelo usuário.

5. Conclusões e Trabalhos Futuros

Este artigo apresentou o simulador ARMS que faz reconhecimento das figuras 5.1 e 5.2 do livro "Organização e Projeto de Computadores de Patterson e Hennessy, 4ed.", provendo ao aluno uma ferramenta de simulação em realidade aumentada, desenvolvida para dispositivos com o SO Android com câmera, totalmente integrada ao material de

estudo. Tal livro é o mais utilizado no ensino de Organização e Arquitetura de Computadores, e por isso já existem diversas soluções de simulação, entretanto elas não apresentam integração com o livro didático. Com a ferramenta proposta para dispositivos móveis, ao estudar pelo livro, o aluno pode recorrer ao simulador para facilitar a assimilação dos conceitos apresentados tendo como possibilidade a execução de instruções e obtenção de informações sobre o funcionamento dos componentes do processador.

O ARMS continua em desenvolvido, assim, em breve ele deve fazer o reconhecimento de mais figuras do livro na versão em português e das mesmas equivalentes na versão em inglês e incluir ainda mais funções. Como recurso didático, o livro apresenta apenas um subconjunto das instruções do MIPS, de modo que os caminhos de dados e controles apresentados são limitados a tais instruções. Assim, pretende-se também estender o caminho de dados, adicionando componentes as figuras que são necessários para a execução de instruções não abordadas no livro, a exemplo das instruções jr (*jump register*) e lui (*load upper immediate*), tornando a ferramenta um recurso adicional ao livro. Outra funcionalidade que será adicionada são exercícios que poderão ser aplicados pelo professor para verificar o aprendizado, os quais poderão retornar um *feedback* para o professor. Além disso, pretende-se fazer a validação da ferramenta. Para isso, será feita uma comparação entre o desempenho obtido pelo grupo de teste (que usará o ARMS) e o obtido pelo grupo de controle (que será submetido ao método tradicional de ensino) em testes que serão aplicados antes e depois da utilização da ferramenta. Ao grupo de teste também será aplicado um questionário para avaliação da ferramenta.

Referências

Araújo, M. R. D., Pádua, F. L. C., Corrêa Junior, F. L. (2014). MIPS X-Ray: A MARS Simulator Plug-in for Teaching Computer Architecture. Em *Recent Contributions from Enginnering, Science & IT (iJES)*.

Akçcayir, M. e Akçcayir, G (2017). Advantages and challenges associated with augmented reality for education: A systematic review of the literature. Em *Educational Research Review*.

Branovic, I, Giorgi, R e Matinelli, E. (2004). WebMIPS: A New Web-Based MIPS Simulation Environment for Computer Architecture Education. Em *31st Annual International Symposium on Computer Architecture*, Munique, Itália.

Felix, A. F., Pousa, C. V. e Carvalho, M. B. (2006). DIMIPSS: Um simulador didático e interativo do MIPS. Em *Workshop sobre Educação em Arquitetura de Computadores (WEAC)*, Ouro Preto - MG.

Kabir, M. T., Bari, M. T. e Haque, A. L. (2011). Visimips: Visual simulator of MIPS32 pipelined processor. Em *Computer Science & Education (ICCSE).*

Kirner, C. e Siscoutto, R. A (2007). Fundamentos de Realidade Virtual e Aumentada. Em *Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações*.

Nova, B., Ferreira, J. C. e Araújo, A. (2013). Tool to Support Computer Architecture Teaching and Learning. Em *International Conference of the Portuguese Society for Engineering Education (CISPEE)*.

Patterson, D. A. e Hennessy, J. L. (2004) Computer organization and design: the hardware/software interface. Morgan Kaufmann. 4ª edição.

Vollmar, K. e Sanderson, P. (2005). A MIPS Assembly Language Simulator Designed For Education. Em *The Journal of Computing Sciences in Colleges*, Vol. 21, No. 1.

Patrocinador Diamante

GOVERNO DO RIO GRANDE DO NORTE

Patrocinadores Bronze

ESIG

dynavideo

CABOTELECOM

Sistema FIERN

NATAL

PARNAMIRIM

Apoio Financeiro

CAPES

MINISTÉRIO DA EDUCAÇÃO

BRASIL

CNPq

Conselho Nacional de Desenvolvimento Científico e Tecnológico

nic.br

cgi.br