# Um sistema crítico e coletor de design rationale integrados em um ambiente para análise e projeto orientados a objetos

Jair S. Ferreira Jr.[1] Dogeval A. Sachett[1] Cleidson R. B. Souza[1]
Jacques Wainer[2]

[1]Departamento de Informática
Universidade Federal do Pará
{crbs@ufpa.br, j2@amazon.com.br, sachett@supridad.com.br}

[2]Instituto de Computação
Universidade Estadual de Campinas
{wainer@dcc.unicamp.br}

**Sumário**

O objetivo deste trabalho é descrever um ambiente para análise e projeto orientado a objetos que implementa a idéia de DODE's proposta por Fischer. Este ambiente contém um kit de construção e um sistema crítico integrados. O sistema crítico utiliza críticas definidas pelo usuário para avaliar a corretude de um diagrama. Desta forma, ele efetua uma avaliação dos modelos de análise e projeto desenvolvidos por projetistas diminuindo os erros e aumentando as chances de reutilização. Além disso, ele utiliza uma abordagem integrada entre DODE's e sistemas para coleta de design rationale.
**Palavras-Chave**: Sistemas críticos, design rationale, ambientes de desenvolvimento de software.

**Abstract**

This paper discusses an environment for object-Oriented analysis and design developed at UFPA in cooperation with UNICAMP. This environment implements Fischer's DODE idea supporting a construction kit and an integrated critiquing system. The critiquing system uses user-defined critics to evaluate the diagram correctness. Besides, this environment also provides an integration between Fischer's environment and a semi-automated design rationale collector.
**Key words**: critics systems, design rationale, software development environments.

## 1 Introdução

A utilização do paradigma de objetos no desenvolvimento de aplicações traz uma série de benefícios ao processo de desenvolvimento de software como por exemplo, a arquitetura resultante do software

é modular e reutilizável, a manutenção simplificada, custo de desenvolvimento reduzido devido a possibilidade de reutilização, etc.

Entretanto, a simples utilização do paradigma de objetos (através dos conceitos de objetos, classes, herança, etc) não garante a qualidade das aplicações: hoje, já é possível encontrar vários sistemas desenvolvidos com estes conceitos e que apresentam estruturas rígidas e inflexíveis, tornando difícil a incorporação de novas funcionalidades. Isto ocorre porque nem todos os modelos de objetos são necessariamente modelos flexíveis, reutilizáveis e de boa qualidade. Para evitar este problema, é necessário que os modelos desenvolvidos sejam avaliados (criticados) visando identificar aspectos do sistema que devem ser modificados para torná-los mais flexíveis e reutilizáveis. Isto é, deve-se identificar construções que possam resultar em futuros problemas de manutenção e reutilização e modificar estas construções para evitar estes problemas.

O objetivo deste trabalho é descrever o ambiente ABCDE-Critic que implementa um sistema crítico para diagramas de análise e projeto orientados a objetos. Um sistema crítico, ou simplesmente crítico, é um software que monitora as ações do usuário e dispara um sinal quando qualquer uma dessas ações viola a definição de uma de suas críticas[3]. Este crítico identifica as construções problemáticas existentes no modelo de objetos e faz sugestões de como resolver estes problemas.

Para que possam ser efetivamente úteis[3], os sistemas críticos devem ser inserido em outros ambientes complexos e de alta funcionalidade chamados de *Domain Oriented Design Environments*(DODEs) [3]. DODEs são sistemas computacionais baseados em conhecimento que dão suporte aos projetistas na especificação de um problema e na construção de sua solução[3].

O ambiente ABCDE-Critic implementa a idéia de DODE's proposta por Fischer[4]. Ele contém: (i) um kit de construção; (ii) um sistema hipermídia argumentativo; (iii) e um sistema crítico onde o usuário pode definir suas próprias críticas. O sistema crítico também apresenta uma abordagem integrada para representar as críticas e o *design rationale*(argumentações sobre o projeto) sobre o diagrama em construção.

O restante deste artigo está organizado como segue. Na próxima seção a definição de críticas e sistemas críticos será apresentada. Esta seção também apresenta a definição de DODEs, bem como a sua arquitetura, seus componentes e os mecanismos de integração destes. A seção 3 apresenta o ambiente ABCDE-Critic, um sistema que implementa a idéia de DODE no domínio da engenharia de software. Finalmente, algumas conclusões e trabalhos futuros são descritos.

## 2 Críticas e *Domain Oriented Design Environments*

O ato de criticar é a apresentação de uma opinião justificada sobre um produto ou ação que pede reflexões futuras ou mudanças sobre o artefato sendo projetado[3]. Um agente - homem ou máquina - capaz de criticar nesse contexto é um *crítico*. Críticas de computador são compostas de conjuntos de regras ou procedimentos para avaliar diferentes aspectos de um produto.

Através do processo de crítica, os projetistas podem obter um melhor entendimento dos problemas de projeto ao ouvir diferentes pontos de vista de outros participantes do projeto. Um *sistema crítico* monitora as ações do usuário e dispara um sinal quando qualquer uma dessas ações viola a definição da crítica. Pode-se adaptar esta idéia para o domínio de desenvolvimento de software orientado a objetos: assim, uma crítica poderia ser formulada com a seguinte definição: *"Uma classe não deve ter atributos públicos"*. Essa crítica seria disparada quando o usuário criasse uma classe que possuísse um atributo público.

Os sistemas críticos são usados em diferentes domínios tais como projetos de circuitos, tomada de decisão, edição de texto, etc[10]. Entretanto, um sistema crítico deve ser inserido em outros ambientes chamados de *Domain Oriented Design Environments*(DODE) para serem mais úteis[3]. Essa inserção aumenta a riqueza do processo de crítica. Os DODEs são sistemas computacionais baseados em conhecimento que dão suporte aos projetistas na especificação de um problema e na construção de sua solução[2].

Baseado nos numerosos esforços de criação destes ambientes e na análise dos problemas encontrados em esforços anteriores, Fischer[2] desenvolveu uma arquitetura independente de domínio para servir como um ponto de partida para a criação de DODEs. Esta arquitetura foi desenvolvida durante anos de pesquisa, entrevistas e, principalmente, com testes com usuário. Ela possui diversos componentes[2], entretanto apenas alguns destes componentes são adequados para este trabalho: (i)*Kit de construção:* é o meio principal para a modelagem de um projeto. Ele possui uma paleta de conceitos de domínio e suporta a construção de projetos utilizando manipulação direta e formulários eletrônicos. (ii) *Sistema de argumentação hipermídia:* contem questões, respostas e argumentos sobre o domínio de projeto; e o (iii)*Catálogo:* é uma coleção de projetos previamente armazenados que ilustram os possíveis projetos no domínio, oferecendo apoio a reutilização.

O poder desta arquitetura baseia-se na integração de seus componentes[2]. Novamente, existem vários mecanismos para realizar a integração, mas o mais importante dele é o *analisador de construção* que é um sistema crítico analisa o artefato que está sendo projetado. O disparo de uma crítica sinaliza uma interrupção para os usuários e leva-os a uma entrada no local exato do sistema argumentativo hipermídia onde a argumentação correspondente está localizada.

# 3 O Ambiente ABCDE-Critic

Esta seção descreve o sistema ABCDE-Critic, um ambiente para análise e projeto orientados a objetos baseado no conceito de DODEs. Ele foi construído a partir do framework de aplicações ABCDE[11]. Este framework permite a construção de editores de diagrama cooperativos baseados em anotações. Desta forma, o ambiente permite a construção de diagramas de classes segundo a notação UML. Isto é, o kit de construção do ambiente foi completamente reutilizado. O ABCDE-Critic foi implementado usando a linguagem de programação Java e possui aproximadamente 50 classes. Estas classes implementam o sistema crítico e o modelo de anotações para coleta semi-automática de design rationale. Estes dois componentes serão descritos nas seções a seguir.

**Coleta de Design Rationale**

*Design rationale*(DR) é uma representação utilizada para documentar o raciocínio e a argumentação que fazem sentido em um artefato específico[6]. DR inclui as razões que levaram a uma decisão de projeto, assim como a justificativa para esta decisão, as outras alternativas consideradas, os compromissos avaliados, e a argumentação que conduziu à esta decisão[5]. Esta informação é útil de diversas maneiras tais como: colaboração, manutenção, aprendizado, etc.

Souza *et al.*[12] desenvolveu um modelo de anotações para coleta semi-automática de DR utilizando anotações como forma de representar os elementos de DR. Este modelo é considerado semi-automático porque permite a coleta da informação durante a construção do diagrama pelos projetistas. Este registro não pára, é não intrusivo e não necessita de um especialista em

documentação.

Este modelo foi reutilizado no ABCDE-Critic e apresenta os seguintes tipos de anotações: (i) *questão:* modela um problema a ser resolvido. Elas podem ser utilizadas para delinear contextos dentro do espaço de projeto, garantindo que as opções sejam comparadas umas com as outras. (ii) *opção:* é uma solução para uma questão. Uma vez que um projetista identificou uma solução para o problema, ele pode construir esta solução. Existem dois tipos de opções: opiniões ou alternativas. A opinião é um descrição textual da solução. A alternativa representa um conjunto de objetos de diagrama arranjados de maneira a resolver a questão. Finalmente, a (iii) *justificativa:* é usada pelos usuários para documentar suas decisões. Quando alguma opção é escolhida como sendo a melhor solução para o problema modelado pela questão, uma justificativa é criada para documentar as razões que conduziram a escolha da opção selecionada. A integração destas anotações com o sistema crítico será descrita na próxima seção.

**Críticas no ABCDE-Critic**

A parte mais importante do ambiente é o sistema crítico, que dispara críticas quando as suas condições de ativação tornam-se verdadeiras no projeto. As críticas no ABCDE-Critic possuem as seguintes propriedades: (i) *nome* que identifica a crítica; (ii) um *modo de ativação* que define o comportamento de execução e disparo da crítica. Em outras palavras, o modo de ativação define o estado da crítica. Esse estado pode ser: *ativo*; *passivo*; ou *desabilitado*; (iii) a *explicação rápida* que representa uma explicação minimalista[3] da crítica. Essa explicação é uma descrição sucinta do problema que causou o disparo da crítica; (iv) uma *argumentação* que é uma explicação detalhada da crítica, a qual pode ser visualizada pelo usuário se ele não entender a crítica ou quiser mais informações sobre ela. Esta explicação é importante porque identifica as vantagens e desvantagens de se seguir as sugestões da crítica auxiliando o usuário a entender as consequências de se seguir tais sugestões. Se o usuário não entender porque uma sugestão é feita, ele tende a seguir "cegamente" o conselho da crítica sem considerar se tal conselho é ou não apropriado para a situação em questão[3]. O formato dos arquivos de argumentação é HTML, portanto de fácil modificação pelo usuário final. (v) uma *prioridade* que determina a importância da crítica para o usuário; (vi) uma *definição* que é o conjunto de regras que, quando verdadeiras, dispara a crítica. Finalmente, (vii) zero, uma ou muitas *soluções* que são as regras as quais especificam uma disposição de novos elementos para substituir os elementos responsáveis pelo disparo da crítica de maneira que ela não seja mais disparada. Em outras palavras, uma solução modela como o diagrama deve ser contruído de forma que não viole a definição da crítica.

Existem três tipos de crítica no ABCDE-Critic[4]: (i)*Java:* na qual as regras de ativação e solução são codificadas na linguagem de programação Java. Essas regras não podem ser criadas e nem modificadas pelo usuário final, somente por desenvolvedores; (ii) *JEOPS:* neste caso, as regras de ativação e solução são codificadas em JEOPS(*Java Embedded Object Production System*)[1]. O JEOPS é um motor de inferência de primeira ordem que permite uma fácil integração entre regras de produção e a linguagem Java; e (iii) *Prolog:* onde as críticas possuem sua definição e solução codificadas na linguagem Prolog. As críticas em Prolog e JEOPS podem ser criadas e modificadas diretamente pelo usuário final.

Sempre que um crítica é disparada a janela "Things to take care of" é mostrada e apresenta o nome da crítica e a sua explicação rápida em uma lista, como mostra a Figura 1. A crítica disparada indica que o uso de associações "muitos-para-muitos" não é recomendável. Além desta janela, as críticas também são apresentadas na coluna de anotações do ambiente. A coluna de

anotações é um componente da interface gráfica do ABCDE que tem como objetivo indicar as anotações que foram feitas sobre o diagrama[11].

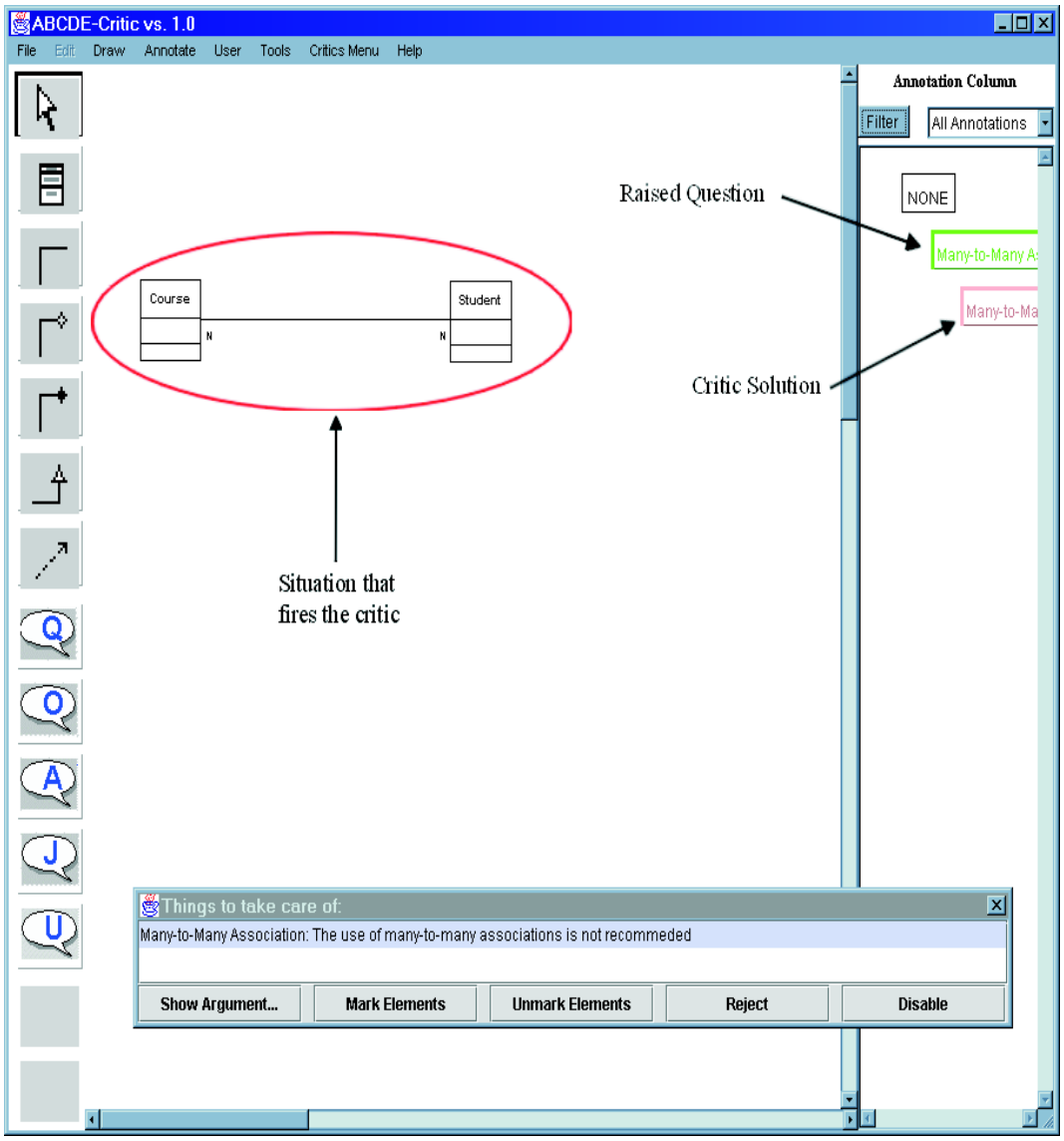


Figura 1: O ambiente ABCDE-Critic

O sistema crítico também está integrado com o sistema de coleta de DR. Tal integração ocorre da seguinte maneira: no momento do disparo da crítica, uma anotação é inserida na coluna de anotações. Esta anotação corresponde a uma *questão* do modelo de DR descrito anteriormente. Se houverem soluções para a crítica, elas são associadas a questão como *opções*. Finalmente, quando uma opção é escolhida como sendo a solução para o problema modelado pela questão, uma *justificativa* é criada para documentar as razões que conduziram a escolha da opção selecionada. As justificativas são utilizadas para o usuário informar as razões pelos quais ele não concordou com as soluções propostas. Se ele concordar com uma das soluções, não é necessário registrar a justificativa, pode-se assumir que as razões que o levaram a modificar o diagrama são óbvias. Isto é necessário porque o usuário não precisa concordar com a crítica e aceitar uma das soluções

que ela propõe. Neste caso, é importante documentar as razões (o DR) que o levaram a esta atitude.

**Prioridades e Perspectivas**

Segundo Fischer, os usuários desejam o poder de organizar e gerenciar o conhecimento de projeto e as críticas devem refletir suas perspectivas no projeto[3]. Em outras palavras, os ambientes de projeto precisam permitir que um diagrama seja interpretado sob várias perspectivas diferentes. Por exemplo, um ambiente de projeto para cozinhas pode oferecer perspectivas técnicas, estruturais e funcionais, e criticar o projeto de acordo com cada uma destas. Desta forma, Fischer criou críticas interpretativas que permitem aos projetistas criar críticas e conhecimento de projeto agrupados em tópicos; tais grupos ajudam os projetistas a examinar seus artefatos sob pontos de vista diferentes.

No ABCDE-Critic, dois conceitos são usados para fornecer aos usuários esta característica: prioridades e perspectivas. *Prioridades* são valores associados as críticas utilizados para ordenar as críticas de maneira que, se duas críticas disparam, aquela com maior prioridade é testada primeiro e portanto, apresentada inicialmente na coluna de anotações. Os usuários podem modificar as prioridades das críticas para refletir seus interesses no projeto atual.

A *perspectiva* é um conceito similar a crítica interpretativa. O ambiente permite que os usuários criem diferentes perspectivas para criticar o diagrama. Por exemplo, para um diagrama em UML poderia existir uma perspectiva Java que descreveria como implementar o diagrama nesta linguagem de programação. Possivelmente, essa perspectiva conteria uma crítica que dispararia quando detectasse herança múltipla, visto que Java não implementa esse conceito.

# 4 Conclusões e Trabalhos Futuros

Este artigo apresentou o ambiente ABCDE-Critic que implementa o conceito de DODEs aplicado ao domínio de análise e projeto orientados a objetos. Nosso ambiente implementa diversos componentes da arquitetura dos DODEs: kit de construção; sistema de argumentação hipermídia; e o sistema crítico. Desta forma, ele auxilia a construção de modelos de objetos ao avaliar estes modelos a procura de construções incorretas ou inflexíveis. Além disso, ao integrar o sistema crítico ao modelo de DR ele auxilia a tomada de decisões dos usuários, permitindo assim a melhoria da qualidade dos modelos desenvolvidos.

O ABCDE-Critic implementa um modelo de usuário simples onde o usuário pode controlar as estratégias de intervenção das críticas e suas prioridades de apresentação, sua definição e suas soluções. As críticas podem ser especificadas utilizando-se o JEOPS[1] ou a linguagem de programação Prolog.

Este trabalho é similar à ferramenta Argo/UML desenvolvida por [8, 9]. Esta ferramenta também implementa um sistema crítico para verificação de diagramas em UML identificando erros comuns dos projetistas. Entretanto, ela não permite a criação de novas críticas, assim como a criação de perspectivas e prioridades.

Este ambiente está sendo semeado com críticas e, atualmente, possui cerca de quinze críticas implementadas que correspondem a heurísticas que representam um conhecimento comum sobre modelagem de objetos e identificam potenciais problemas no diagrama. As heurísticas de Riel[7] foram utilizadas, assim como críticas identificadas pelos autores. Apesar do pequeno número de críticas implementadas, acredita-se que essas críticas proporcionam um retorno valioso para os

projetistas. Esta afirmação é confirmada por Fischer[3] que descreve o HYDRA, um sistema com 24 regras, mas que mesmo assim foi considerado útil por projetistas amadores e especialistas.

De fato, o ambiente está sendo utilizado em sala de aula como auxílio à disciplina de análise e projeto orientado a objetos. A partir do *feedback* dos alunos, espera-se identificar críticas que possam se tornar *pró-ativas*, ou seja, críticas cuja solução é considerada bastante simples, portanto podem modificar diretamente o diagrama sem a intervenção do usuário. Resultados preliminares indicam que determinadas críticas podem vir a ser implementadas desta forma, entretanto, outras críticas continuarão a funcionar da forma atual.

# Referências


[1] Carlos Santos da Figueira Filho. *JEOPS - Uma Ferramenta para o Desenvolvimento de Aplica es Inteligentes em Java*. Technical Report, Departamento de Inform tica, Universidade Federal de Pernambuco, 1999.

[2] Gerhard Fischer. Domain-oriented design environments. *Automated Software Engineering*, 1:177–203, 1994.

[3] Gerhard Fischer, Kumiyo Nakakoji, Jonathan Ostwald, Gerry Stahl, and Tamara Sumner. Embedding critics in design environments. *The Knowledge Engineerign Review*, 8(4):285–307, 1993.

[4] Jair S. Ferreira Jr, Dogeval A. Sachett, Cleidson R. B. Souza, and Jacques Wainer. Um ambiente de projeto par an lise e projeto orientados a objetos (in press). In *Anais do Terceiro workshop Ibero-americano de Engenharia de Requisitos*, Canc n, Espanha, 2000.

[5] Jintae Lee. Design rationale systems: understanding the issues. *IEEE Expert*, 78–85, may/june 1997.

[6] Allan MacLean, Richard M. Young, Victoria M. Belloti, and Thomas P. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Human-Computer Interaction*, 6(3-4):201–250, 1991.

[7] Arthur J. Riel. *Objetc-Oriented Design Heuristics. Object-Oriented Technology*, Addison Wesley, 3 edition, 1996.

[8] Jason E. Robbins and David F. Redmiles. Cognitive support, uml adherence, and xmi interchange in argo/uml. In *Proceedings of the International Conference on Construction of Software Engineering Tools*, 1999.

[9] Jason Elliot Robbins. *Cognitive Support Features for Software Development Tools*. PhD thesis, Department of Information and Computer Science, University of Califorina, Irvine, 1999.

[10] B. G. Silverman. Survey of Expert Critiquing Systems: practical and theoretical frontiers. *Communications of the ACM*, 35(4):106–127, april 1992.

[11] Cleidson R. B. Souza. *Um Framework para Editores de Diagramas Cooperativos Baseados em Anotações*. Master's thesis, Instituto de Computação - Universidade Estadual de Campinas, Campinas - SP, september 1998.

[12] Cleidson R. B. Souza, Jacques Wainer, Daucikelem B. Santos, and Klissiomara L. Dias. An annotation model and tool for semi-automatic recordign of design rational in software diagrams. In *Cyted Rhytos International Workshop on Groupware - CRIWG*, Cancún, Espanha, 23 - 25 Sep 1999.