Model Driven RichUbi – Model Driven Process for Building Rich Interfaces of Context-Sensitive Ubiquitous Applications*

Carlos Eduardo Cirilo[1]
Antonio Francisco do Prado[1], **Luciana Aparecida Martinez Zaina**[2]

[1]Graduate Program in Computer Science (PPG-CC)
Federal University of São Carlos (UFSCar) – São Carlos – SP – Brazil

[2]UFSCar, Campus at Sorocaba – SP – Brazil

{carlos_cirilo, prado}@dc.ufscar.br, lzaina@ufscar.br

Abstract. *The demand for software in ubiquitous computing has raised new challenges for Software Engineering. One such challenge is related to the adaptation of applications accessed by different devices in multiple contexts of use. Amid this diversified scenario in access possibilities, Software Engineering is defied to innovate its processes and technologies so as to leverage development of solutions to effectively meet the adaptation requirements of ubiquitous environments. In response to these challenges, in this Master's work we proposed a novel model driven process to address the content adaptation issues of rich interfaces for Web 2.0 applications which adapt themselves when viewed on distinct ubiquitous devices. With focus on software reuse, the process provides activities and artifacts that specially support the development of adaptive rich interfaces, contributing to save efforts and increase productivity. An experimental study demonstrated the potential of the proposed process to enhance efficiency, in terms of time and productivity, of teams developing adaptive rich interfaces.*

1. Background

Web 2.0 has allowed greater interactivity between users and Web applications. Among the principles which guide the development of Web 2.0 applications, the use of rich interfaces that afford users a meaningful experience is noticeable. In this context, applications have transposed the boundaries of read-only interfaces built purely in *Hypertext Markup Language (HTML)*. Through technologies that enable creating more advanced interfaces with interactive resources, Web 2.0 applications resemble appearance, behavior and usability of desktop ones.

With the miniaturization of computational devices for personal use and recent advances in wireless communication technologies, new usage scenarios became possible for applications in different fields. Nowadays, for instance, people can read e-mails, perform financial transactions, share resources (software, hardware, and data), and enjoy a variety of other applications by means of both mobile and stationary devices, either still or moving, at home or at work. In this scenario, the vision of ubiquitous computing, introduced by Mark Weiser about two decades ago [Weiser 1991], has been boosted by technological innovations that foster easy access to information anywhere, anytime and through any device at user's disposal.

The dynamic nature of ubiquitous computing imposes a series of challenges and additional requirements to software development. One of the critical aspects in developing applications for ubiquitous environments is the premise that they must run and adapt themselves to the diversity of computational devices and circumstances in which they are immersed. This implies the self-organization of the application's structure, and automatic adjustment of its behaviour and content to ensure the fulfillment of its requirements and users' needs. However, given the diversity of devices, networks, environments and other factors that influence the interaction between users and applications, meeting the demands of adaptation of ubiquitous environments has become a challenging task for software engineers. Still, in the case of Web 2.0 applications, this task is even more complex due to the need of preserving interaction aspects that afford users a richer experience.

*With financial support from the *Brazilian Coordination for the Improvement of Higher Level Personnel (CAPES).*

As a result, engineers must deal with the challenge of building and maintaining specific application versions to meet each interaction context. Nevertheless, among other problems, this cross-context design may require high investments for large development efforts, and yet inconsistencies can still occur. Furthermore, the existence of multiple application versions hinders the maintenance, since alterations and new features will have to be managed separately. In this sense, it is important to provide developers with an appropriate software process which can guide them through activities and artifacts that support attending the adaptation requirements demanded by ubiquitous environments.

Considering these challenges, in recent years attention has been directed to the so-called *Model Driven Development (MDD)* approaches [France and Rumpe 2007] in the construction of ubiquitous applications, as they allow raising abstraction level and increasing automation and productivity in development (e.g., [Serral et al. 2010, Ayed et al. 2007, Henricksen and Indulska 2006]). In view of this, we proposed in this Master's work the process named Model Driven RichUbi, which aims to explore the benefits of MDD practice geared for building rich interfaces for context-sensitive ubiquitous applications [Cirilo 2011]. Based on the MDD-related conception of *Domain-Specific Modelling (DSM)* [Kelly and Tolvanen 2008], the process defines activities and specialized artifacts that aid the modeling and the code generation of rich interfaces for ubiquitous applications, as well as the adaptation of the interfaces' content[1] for different devices.

1.1. Why is it important to concern about development of context-sensitive ubiquitous applications?

With the Internet becoming omnipresent, the use of network-based services and applications is being increasingly widespread and it is expected to grow in the upcoming years, while mobile devices with fast data connection proliferate quite rapidly.

Forecasts[2,3] indicate that between 2013 and 2014, over 3 billion people around the world will be able to transact electronically via mobile devices or *personal computers (PCs)*. The trend is that the number of mobile clients accessing online applications overtakes the one of stationary computers. By 2014, the penetration rate of mobile devices equipped with Internet access capabilities in the global market will be 90%, exceeding 1.82 billion units, whereas the total number of PCs in use will reach 1.78 billion units. Smartphones and tablets lead this new net growth in device adoption for the coming years. Increasing application platform capability across all classes of mobile devices is spurring a new frontier of innovation, particularly where mobile capabilities can be integrated with contextual information, such as location, presence and social information, to enhance the usefulness. The predictions indicate that by 2015, context will be the main source of information for applications to delivery personalized content and to adapt to different situations and devices. In fact, this reality has led to the urgent need for applications that provide support for ubiquity and different types of devices and users in multiple contexts of use.

Adapting applications according to context is a fairly recent research topic and has attracted the interest of many researchers throughout the last years. Different solutions have been proposed by the academic community. Some of them aims at defining frameworks that provide functionalities for content and behavior adaptation (e.g., [Forte et al. 2008, Woensel et al. 2009]). Other solutions focus on developing tools to aid the development of adaptive applications (e.g., [Paternò et al. 2008, Viana and Andrade 2008]). There are also studies that have focused on defining software processes to guide the construction of context-sensitive and ubiquitous applications (e.g., [Vieira et al. 2011, Serral et al. 2010, Ayed et al. 2007, Henricksen and Indulska 2006]).

To illustrate the importance that is being given to research focusing on the development of context-sensitive ubiquitous applications, in 2006 a committee organized by the *Brazilian Computer Society (SBC)*, composed by researchers from several areas, defined the five big challenges in Computer Science for the next 10 years [SBC 2006]. The application adaptation (content, information and behavior) based on contextual information is mentioned in three of these challenges, namely: information retrieval in large volumes of data to provide more appropriate information to user's needs and preferences; the production of flexible and adaptable content and interfaces

[1] In the scope of this work, content in a user interface refers to interaction components and widgets as well as their arrangement and layout.

[2] http://www.gartner.com/it/page.jsp?id=1278413.

[3] http://www.gartner.com/it/page.jsp?id=1862714.

that fit to users' socio-cultural context, in order to enable universal access to knowledge to all citizens; and development of ubiquitous applications to be accessed anywhere, anytime and from any device.

2. The Model Driven RichUbi process

The Model Driven RichUbi is based on the conceptions of MDD and DSM to support the development of adaptive rich interfaces for ubiquitous applications in the field of Web 2.0. In the process the interface modeling is performed from a Rich Interfaces domain metamodel, which expresses the abstract syntax of a *Domain-Specific Language (DSL)* [Sadilek 2008] that facilitates translating the application's requirements into suitable interface components for interaction of end users. Model-to-Code (M2C) transformations are employed to enable generating partial code of the interfaces for different target platforms, and content adapters are used to refine the interfaces at runtime in order to meet specific features retrieved from the context of the user's access devices.

As depicted in Fig. 1, the process is performed in two main steps: *Domain Engineering (DE)* and *Application Engineering (AE)*. The DE encompasses activities for constructing reusable artifacts that support the development of adaptive rich interfaces. These artifacts comprise: *a)* the Rich Interfaces domain metamodel to support the interface modeling; *b)* the M2C transformations for code generation; and *c)* the content adapters. The AE covers the activities for the development of rich interfaces for ubiquitous applications with reuse of the artifacts produced in the DE. As the process' support artifacts are focused on the Rich Interfaces domain - a cross-cut domain to the application ones - they can be reused on the development of adaptive rich interfaces for ubiquitous applications of several fields, which contributes to save efforts and increase productivity.

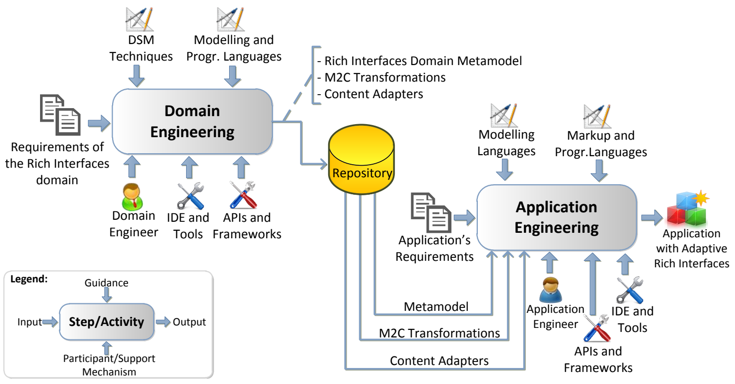
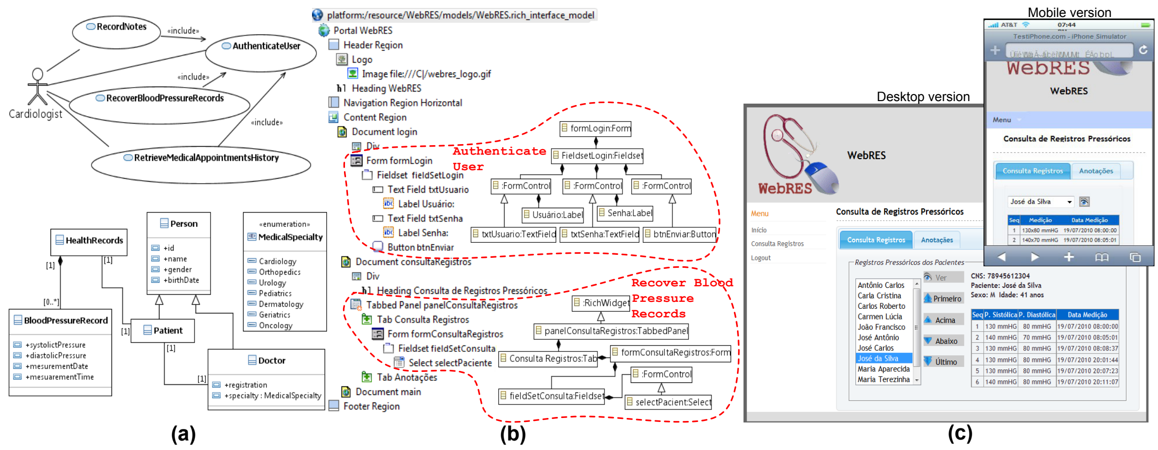


Figure 1. High-level overview of the Model Driven RichUbi process

The process begins in the DE, where the metamodel is built from the requirements of the Rich Interfaces domain. Besides, the M2C transformations and the content adapters are also developed based on the definition of the interface components in the metamodel. In the AE step the metamodel is instantiated to create the application's interface models. Since these models are platform-independent, the M2C transformations can be applied to generate partial code for distinct interface versions from the same models in order to meet features of different groups of ubiquitous access devices.

As an example, Fig. 2 illustrates some artifacts resulting from the AE's activities performed to develop the Web module (WebRES) of a ubiquitous application in the *Electronic Health Records (EHR)* domain, which allows health caregivers remotely monitoring patients' blood pressure through different access devices. Figure 2(a) shows an excerpt of the use case and class diagrams that specify the WebRES' requirements and its main domain entities. Figure 2(b) presents the interface model which translates the identified requirements into interface components. For instance, the use cases `AuthenticateUser` and `RecoverBloodPressureRecords` were mapped, respectively, into an authentication form inside a login page, and a search form inside a tabbed panel widget in a page designed for searching blood pressure records. The figure also illustrates the internal instantiation of the underlying metamodel for these portions of the model. Figure 2(c), in turn, shows the final interface versions for desktops and mobile devices, whose most of code was generated by applying the M2C transformations over the interface model.

Aiming at reducing the number of interface versions to be developed, the Model Driven RichUbi establishes a *hybrid approach* for adapting the interfaces' content, by combining *static*



RecordNotes
AuthenticateUser
RecoverBloodPressureRecords
RetrieveMedicalAppointmentsHistory
Cardiologist
«include»
HealthRecords
Person
«enumeration» MedicalSpecialty
BloodPressureRecord
Patient
Doctor
platform:/resource/WebRES/models/WebRES.rich_interface_model
Authenticate User
Recover Blood Pressure Records
Desktop version
Mobile version
WebRES
Consulta de Registros Pressóricos
(a)
(b)
(c)

Figure 2. Examples of artifacts produced during the AE's activities. (a) Application's specification from the analysis. (b) Application's interface model from the design activity. (c) Interface versions developed on the implementation activity

adaptation (implementation of a specific interface version for each target device at development time) with *dynamic adaptation* (automatic generation of the interface code when the user accesses the application at execution time). In this approach, the application's requirements are mapped into a few generic interface versions, each one implemented for a particular group of devices, with the aid of the metamodel and the transformations. The content adapters perform the dynamic part of the adaptation by selecting, at runtime, the interface version that best fits the device context, and adapting its code snippets so as to meet the characteristics of the access device.

The early experiences with the Model Driven RichUbi process led to its refinement. The content adapters were organized and structured into a framework, called *Ubiquitous Context Framework (UbiCon)*, to facilitate their reuse in the AE step. In addition, UbiCon was outlined as an extensible project so that contextual information related to other entities, such as the user and the network, may be promptly integrated in adaptation. In this manner, it is possible to extend UbiCon for enabling interface adaptations based on the combination of contextual information from different entity profiles in order to meet adaptation requirements of distinct applications. UbiCon then encapsulates the tasks of context manipulation and provides modularized features for rich interface adaptation in a hybrid manner.

3. Experimentation

Some case studies and a controlled experiment were conducted to evaluate the effectiveness of the proposed process and the support provided by its artifacts. In this paper we focus in describing the results of the experimental study, in which we followed the experimentation methodology, introduced in [Wohlin et al. 2000], for assessing the impact of the Model Driven RichUbi over the development practice. The experiment consisted of a comparative study between using the Model Driven RichUbi (AE step) for building adaptive rich interfaces and using a not model driven process, based on the classic life cycle, for the same purpose.

The experiment was conducted by 31 volunteers, 3^{rd} and 4^{th} year Computer Science and Computer Engineering undergraduate students, split into 10 homogeneous teams according to their experience levels. Half of these teams performed the proposed process, while the other half performed the classic life cycle. The assignment of the teams for performing the Model Driven RichUbi or the classic life cycle was accomplished in a completely random way so as not to bias the results of the experiment.

The task of the teams was to develop rich interface versions of the Web module of a ubiquitous application for user tracking, so that it could properly be accessed from both mobile devices and personal computers. During the experiment operation, the teams recorded in a form the start and end of each activity performed, and the number of *lines of code (LOC)* automatically and manually implemented. Table 1 presents some descriptive statistics of the collected data.

In order to statistically verify the effect of the processes on the team's efficiency, in terms of productivity and time, we applied the *t-test* on the collected data. The samples in our study were composed by the datasets concerning the teams' productivity and the total time spent by them to

Table 1. Descriptive statistics of the collected data

Process	Spent Time (in hours)			Productivity (LOC/hour)		
	Mean	Std. Dev.	Median	Mean	Std. Dev.	Median
Model Driven RichUbi	1.00	0.38	0.90	301	122.72	233
Classic Life Cycle	1.50	0.17	1.50	104	32.65	115

build the interface versions. The test was then performed in two separate steps: one to compare the samples concerning the teams' total time, and another one to compare the teams' productivity. The checking of the experiment's null hypothesis (H_0), which states equality between the means of productivity (μ_ρ) and spent time (μ_τ) for the teams from both processes, was based on the combination of the rejection criteria of the two test's steps. Accordingly, H_0 would be rejected if and only if it could be rejected according to both total time and productivity criteria.

Test's 1st step ($H_{0\tau}$: $\mu_{\tau RichUbi} = \mu_{\tau Classic}$)
It has been possible to reject the null hypothesis' statement that there is no difference between the means of time of the teams from both processes with $|t_{0\tau}| > t_{0.0275,8}$ ($\alpha = 0.055$; $t_{0\tau} = -2.7148$; e $t_{0.0275,8} = 2.6899$).

Test's 2nd step ($H_{0\rho}$: $\mu_{\rho RichUbi} = \mu_{\rho Classic}$)
It has been possible to reject the null hypothesis' statement that there is no difference between the means of productivity of the teams from both processes with $|t_{0\rho}| > t_{0.01,8}$ ($\alpha = 0.02$; $t_{0\rho} = 3.4806$; e $t_{0.01,8} = 3.3554$).

Since H_0 could be rejected in the two test's steps, conclusions were drawn about the experiment's results. The test demonstrates that the experiment's alternative hypothesis (H_1) can be validated, i.e., there are evidences to claim that teams using the Model Driven RichUbi for building adaptive rich interfaces are, in general, more efficient than teams which use the classic life cycle. This conclusion is in line with initial expectations about the experiment, that the reusable artefacts built in Domain Engineering could make more agile application engineers' tasks.

The replication of the experiment in an industrial setting is an important step forward in validating our work, as it will collaborate to extend the amplitude of the results to new contexts, where the Model Driven RichUbi shall be compared with other development approaches (e.g. agile methods, software product lines) also considering factors absent in academic environment. In addition, other effects related to the quality of the interfaces, such as usability, may also be studied. In this case, empirical evaluations with users and inspection tests should be planned in order to collect relevant metrics for assessing the degree to which the produced interfaces reach the goals of efficiency, effectiveness and subjective satisfaction from the point of view of end users.

Please refer to [Cirilo 2011] for further remarks about the experimental data analysis, and discussions on threats to validity and generalizability of the experiment's results.

4. Contributions and results of publications

The main contributions of this work for Software Engineering are twofold: firstly, it depicts a new MDD process focused specifically on the presentation layer of context-sensitive ubiquitous applications; secondly, it contributes to the state of the art of MDD so as to encourage research and enhancements in this field, pointing out feasible gains from applying the model driven approach for developing adaptive rich interfaces in ubiquitous computing.

Other contributions are distilled from the artifacts already developed in the DE step (Rich Interfaces domain metamodel, M2C transformations, and UbiCon framework), which are available to support modeling and partial code generation of rich interfaces for different target platforms, as well as content adaptation to various ubiquitous access devices. Furthermore, the hybrid adaptation approach introduced in the process is another important contribution of this work, given the potential economy of development efforts by reducing the number of rich interface versions to be developed without neglecting the increasing range of access devices released on the market.

The results of this work were published and presented at major conferences on Software Engineering. The proposal was initially presented and discussed at the *XV Workshop of Theses and Dissertations in Software Engineering (WTES)*, where we received valuable suggestions and contributions to improve the work. The summary of this initial proposal is published in (i). The definition of the Model Driven RichUbi and detailed description of its activities, including results of early case studies, are reported in (ii) and (iii). The organization of the experimental study and quantitative analysis of the experiment's results are published in (iv). Qualitative analysis on the use of the Model Driven RichUbi from the perspective of the experiment's participants is reported in (v). The presentation of the hybrid adaptation approach along with the UbiCon framework is part of the work published in (vi), which received the 1st place best paper award at the conference. An extended journal version of this latter paper is in press (vii). Finally, a book chapter was written covering the whole work, including the description of the process in its current state and comprehensive discussion of the experiment's results (viii).

Publications:

(i) "Model Driven RichUbi - Processo Dirigido a Modelos para a Construção de Interfaces Ricas de Aplicações Ubíquas Sensíveis ao Contexto", CBSoft/XV WTES, 2010.

(ii) "Model Driven RichUbi - Processo Dirigido a Modelos para a Construção de Interfaces Ricas de Aplicações Ubíquas Sensíveis ao Contexto", CBSoft/XXIV SBES, 2010.

(iii) "Model Driven RichUbi - A Model Driven Process for Building Rich Interfaces of Context-Sensitive Ubiquitous Applications", ACM 28th SIGDOC, 2010.

(iv) "Experimentação do Processo Model Driven RichUbi no Desenvolvimento de Interfaces Ricas Adaptativas", CBSoft/XXV SBES, 2011.

(v) "Estudo Experimental do Processo Model Driven RichUbi: Análise Qualitativa", XV Congresso Ibero-Americano em Engenharia de Software (CIbSE), 2012.

(vi) "A Hybrid Approach for Adapting Web Graphical User Interfaces to Multiple Devices using Information Retrieved from Context", 16th Int. Conf. on Distributed Multimedia Systems (DMS), 2010.

(vii) "Towards a Hybrid Approach for Adapting Web Graphical User Interfaces to Heterogeneous Devices using Context", Int. Journal of Software Engineering and Knowledge Engineering (IJSEKE), 2012.

(viii) "Building Adaptive Rich Interfaces for Interactive Ubiquitous Applications", In: Ioannis Deliyannis (editor). Interactive Multimedia. InTech – Open Access Company, 2012.

References

Ayed, D., Delanote, D., and Berbers, Y. (2007). Mdd approach for the development of context-aware applications. In *Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*, CONTEXT'07, pages 15–28, Berlin, Heidelberg. Springer-Verlag.

Cirilo, C. E. (2011). Model Driven RichUbi – Processo Dirigido a Modelos para a Construção de Interfaces Ricas de Aplicações Ubíquas Sensíveis ao Contexto. Master's thesis, UFSCar, Brazil.

Forte, M., de Souza, W. L., and do Prado, A. F. (2008). Using ontologies and web services for content adaptation in ubiquitous computing. *J. Syst. Softw.*, 81(3):368–381.

France, R. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54, Washington, DC, USA. IEEE Computer Society.

Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive Mob. Comput.*, 2(1):37–64.

Kelly, S. and Tolvanen, J.-P. (2008). *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley.

Paternò, F., Santoro, C., and Scorcia, A. (2008). Automatically adapting web sites for mobile access through logical descriptions and dynamic analysis of interaction resources. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '08, pages 260–267, New York, NY, USA. ACM.

Sadilek, D. A. (2008). Prototyping domain-specific language semantics. In *Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications*, OOPSLA Companion '08, pages 895–896, New York, NY, USA. ACM.

SBC (2006). Grandes desafios da pesquisa em computação no brasil - 2006–2016. Technical report, Sociedade Brasileira de Computação.

Serral, E., Valderas, P., and Pelechano, V. (2010). Towards the model driven development of context-aware pervasive systems. *Pervasive Mob. Comput.*, 6(2):254–280.

Viana, W. and Andrade, R. M. C. (2008). Xmobile: A mb-uid environment for semi-automatic generation of adaptive applications for mobile devices. *J. Syst. Softw.*, 81(3):382–394.

Vieira, V., Tedesco, P., and Salgado, A. C. (2011). Designing context-sensitive systems: An integrated approach. *Expert Syst. Appl.*, 38(2):1119–1138.

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 3(3):3–11.

Woensel, W., Casteleyn, S., and Troyer, O. (2009). A framework for decentralized, context-aware mobile applications using semantic web technology. In *Proc. Confederated Int. Workshops and Posters on On the Move to Meaningful Internet Systems*, OTM '09, pages 88–97, Berlin, Heidelberg. Springer-Verlag.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA.