User Interface Stereotypes: A Model-Based Approach for Information Systems User Interfaces

Sofia Larissa da Costa[2], **Valdemar Vicente Graciano Neto**[1,2], **Juliano Lopes de Oliveira**[1], **Bruno dos Reis Calçado**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia
P.O Box 131 - CEP 74001-970 - Goiânia - GO - Brazil

[2]Instituto de Ciências Matemáticas e Computação (ICMC)
Universidade de São Paulo - USP
Avenida Trabalhador São-carlense, 400 - Centro
CEP: 13566-590 - São Carlos - SP - Brazil

sofialc@icmc.usp.br, valdemarneto@usp.br

juliano@inf.ufg.br, onurbrc@gmail.com

Abstract. *This paper presents a model-based approach to build Information Systems User Interfaces (ISUI). In this approach, UI presentation and behavioral aspects are modeled as UI Stereotypes, which are high level abstractions of UI appearance and interaction features. A taxonomy of ISUI elements is proposed as the basis for definition of UI stereotypes. These elements are orchestrated on a software architecture which manages model-based UI building and integration with the IS applications. The proposed approach reduces software development efforts and costs, facilitating maintenance and evolution of ISUI. Moreover, UI stereotypes improve usability, consistency, reuse and standardization of both presentation and behavior of ISUI.*

Keywords: *Information Systems User Interface, User Interface modeling, User Interface Stereotype, Model-based User Interface generation.*

1. Introduction

The User Interface (UI) is one of the most complex components of modern Information Systems (IS) software. Engineering the UI involves modeling several types of business processes and data to make the user experience both pleasant and efficient. Besides that, the UI construction process is expensive and error-prone, since it must deal with many distinct questions, ranging from psychological aspects of user modeling to technological variations of graphical renderization in different platforms.

The UI Engineering discipline emerged from the lack of an effective process to cope with these UI issues along the software development life cycle [Vanderdonckt 2005]. Model-based UI (MBUI) development is one of the mainstream UI Engineering methods [Calvary et al. 2003]. It is defined as the process of creating and refining high level models for the automatic generation of UI. The use of abstract models enhances the

user comprehension of how the requirements are transformed into UI software and facilitates the reuse of UI concepts [Ahmed and Ashraf 2007, Memmel and Reiterer 2009, Falb et al. 2007].

In spite of the significant advances in this area, it is not yet possible to automatically create complete solutions for UI due to the lack of support for many important tasks of the UI construction process, such as composition and refinement of UI elements, reuse of UI components in different domains and applications, integration of UI components to the underlying IS business logic, and modeling of a wide range of events related to the behavior of UI [Vanderdonckt 2008, Ahmed and Ashraf 2007, Melia et al. 2008, Memmel and Reiterer 2009]. The limitations of current approaches become even more evident in the context of Information Systems (IS) when the Business Process Modeling is considered [de Oliveira et al. 2011].

The MBUI approach for automatic building of UI presented in this paper takes into account the behavioral and presentation aspects of IS applications and overcomes some important limitations of current methods, namely the specification of IS applications behavior in response to user interactions, and the integration of UI and IS applications software. This approach is based on the extension and integration of two frameworks: the UI development model proposed by [da Costa et al. 2010] and the IS software application framework described by [Almeida et al. 2009].

The cornerstone of the proposed MBUI development model is the concept of *User Interface Stereotype*, which allows the definition of a set of metamodels for building ISUI. Model-driven IS engineering process supported by the IS application framework relies on these metamodels for building the UI software components and to integrate these components to the IS application architecture [Graciano Neto and de Oliveira 2013].

The remainder of this paper describes the main features of this MBUI development approach and is organized as follow. Section 2 discusses related works in the context of UI Engineering. Section 3 presents the metamodels that describe the main aspects of the UI in our approach. Section 4 illustrates the application of the approach in a case study. Finally, Section 5 concludes the paper and points to future works.

2. Model-Based User Interface Engineering

The UI Engineering discipline focuses on methodologies for UI development, considering the whole software life cycle [Vanderdonckt 2005]. The CAMELEON reference framework establishes the basis for this discipline, decomposing UI development in four steps: (1) Modeling Domain Tasks and Concepts; (2) Definition of Abstract UI containers and components; (3) Concretization of UI components; (4) and Final UI generation [Calvary et al. 2003].

The domain model should describe objects that will be handled by users while they interact with the system. Typically UML class diagrams or other object-oriented notations are used to this purpose. User tasks modeling is a well established technique that describes tasks an user must perform and the relationship between them [Limbourg and Vanderdonckt 2003].

The other steps of the reference framework – Abstract UI, Concrete UI and Final UI – involve UI modeling and construction. MBUI approaches to these steps have been

extensively investigated [Blouin and Beaudoux 2010, Da Silva and de Oliveira 2009, Memmel and Reiterer 2009, Xudong and Jiancheng 2007, Vanderdonckt 2008, Falb et al. 2007, Valverde et al. 2007]. However, no consensus has been reached and no method has emerged as a standard [Vanderdonckt 2008].

MBUI development uses metamodels to specify essential characteristics for the HCI design through a high-level UI description. An important aspect of a metamodel is its ability to incorporate design patterns. Interaction design patterns focus on solutions for usability problems, forming a collection of UI best practices that can be applied in several projects consistently [van Welie and van der Veer 2003]. The combination of interaction patterns and UI description enables the building of concrete interaction patterns [Memmel and Reiterer 2009].

We have selected relevant UI development approaches to identify core requirements for MBUI building. The selected works were focused on WIMP (windows, icons, menus and pointers) based UI, and described an implementation validation of the proposed approach. Table 1 presents a comparative analysis of these approaches based on the UI quality criteria synthesized in [da Costa 2011]. The symbol "++" indicates that the corresponding quality criterion is satisfied; "+" indicates that the criterion is partially satisfied; and "-" indicates that the criterion is not satisfied.

Tabela 1. Comparative analysis of MBUI development approaches

Quality Criteria	Malai	Conceptual Discourse Metamodel	GDIG	Abstract Interaction Model	Metamodel for Web Applications	IMML based approach
Beautification	-	-	-	-	+	-
Layout	+	-	+	-	++	+
Separability	+	-	+	-	-	++
Intention	++	++	+	++	+	++
Decomposition	-	-	++	+	++	++
Standardization	++	+	+	+	++	++
Clarity	++	-	-	-	++	++
Flexibility	-	-	+	+	+	+
Direction	-	-	-	-	+	++
Generality	++	++	+	++	++	++
Structure	-	++	++	-	++	++
Contextualization	+	-	++	+	+	+
Correlability	-	-	+	-	+	++

Legend: -: Not satisfied; **+**: Partially satisfied; **++**: Totally satisfied

The Metamodel for Web Applications [Cadavid et al. 2009] and the IMML based approach [Costa Neto et al. 2009] satisfied most quality requirements, but their dominant elements are only in the concrete UI level and they are restricted to form, search, list and master-detail UI patterns. The Malai approach [Blouin and Beaudoux 2010] is the most adherent to CAMELEON while the Conceptual Discourse Metamodel [Popp et al. 2009] has higher expression in terms of interaction, since the use of communicative acts ma-

kes this approach closest to human communication. The Abstract Interaction Model [Valverde et al. 2007] is the most inclusive regarding a taxonomy of forms of interaction. Finally, GDIG [Da Silva and de Oliveira 2009] generates only CRUD (Create, Retrieve, Update and Delete) UI, but it has an automatic generation tool support and the generated UI are fully integrated with the underlying IS applications.

A common drawback of these approaches is placing UI refinement and style aspects in second plan. This makes the change of presentation or behavioral aspects of generated UI a difficult and error-prone task. Moreover, these approaches do not support integration of UI with the underlying IS applications, and provide limited capabilities for modeling UI behavior. The MBUI approach described in the next section uses different metamodels to solve these problems in the context of IS.

3. Metamodels for UI Building

Three metamodels are the cornerstone of the proposed MBUI development approach. The Domain Metamodel describes IS business data and associated rules. It is based on the UML Class Diagram and defines the main IS domain concepts, fulfilling the first step of the CAMELEON Framework. The Domain Metamodel will not be further discussed in this paper, since it is fully described in [da Costa et al. 2010].

The other two metamodels are directly associated to user interaction. The HCI Metamodel enables UI modeling in an abstract level, facilitating its reuse in a wide range of IS. The Presentation Metamodel enables UI description in a concrete level.

3.1. HCI Metamodel

The HCI Metamodel describes abstract characteristics of ISUI, which make intensive use of information stored in databases and organize users tasks related to business processes. This metamodel supports a wide range of IS applications, describing presentation and behavioral aspects of WIMP based UI in an integrated way. These two aspects are handled by separate components within the metamodel in order to isolate the properties of each aspect, but are integrated as a single metamodel to make explicit the way in which both aspects are correlated, representing complex ISUI.

Figure 1 presents the main concepts of the HCI Metamodel. The **Presentation** package describes structural and layout aspects of UI while the **Behavioral** package contains metadata to represent UI actions. The **User Interface Stereotype** package integrates metadata from the other two packages to link presentation and behavioral aspects of ISUI.

The key concept of the HCI Metamodel is the User Interface Stereotype, or simply UI Stereotype, which consists of an abstraction of the UI intention, independently of the application or underlying IS. Thus, each UI stereotype is a UI dominant element, which organize the presentation of the subordinate elements and its behaviors. The structural definition of the stereotype composition is based in a taxonomy of abstract elements of UI. This taxonomy corresponds to the **Presentation Aspect** of the HCI Metamodel, and allows the conceptual specification of the UI, abstracting the real (concrete) interaction objects that are used in the UI implementation.

For each ISUI, a given business concept must be presented with a particular intention [Valverde et al. 2007, Popp et al. 2009, van Welie and van der Veer 2003]. An Interaction Element (or HCI Element) is a constructor that describes the kind of UI element

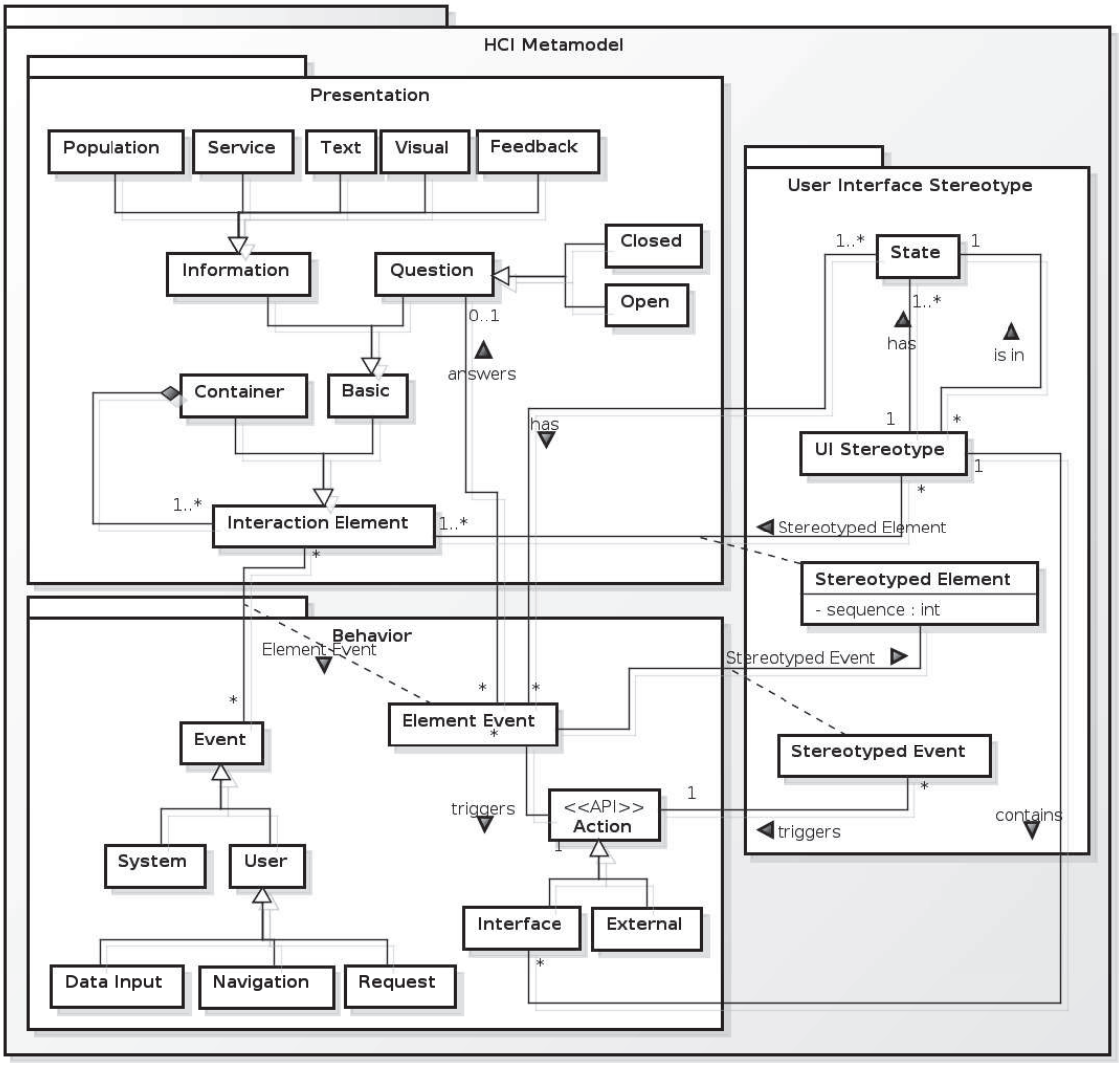


Figura 1. HCI Metamodel

used for communication between users and the system. A Container is a kind of HCI Element that represents a composition of interaction elements. A Basic HCI Element encapsulates an essential communication unit between user and system, and is similar to the concept of communicative acts [Falb et al. 2007]. This element is divided in two types: Question (communicates a questioning to user which requires an answer, which can be an upgrade or data entry) and Information (presents an informative content to the user).

The UI stereotype behavior is defined based on a mechanism for the specification of actions associated to UI events, representing the **Behavior Aspect** of the metamodel, that contains a finite state machine specification to describe the interaction, i.e. the reaction of the UI in response to a user interaction. A set of element events can be enumerated for a UI stereotype, and each of which triggers an action. When these Element Events occur they trigger an action on the UI or in the underlying application.

Among possible User Events there are Data Entry events, that are related to the Question Element for Presentation aspect; Navigation events, which update UI state and

are related to Service elements; and Request events, related to Service elements that trigger external actions. The occurrence of an element event in a given UI state within a UI stereotype characterizes an interaction. A state is a set of system properties perceptible to the users. There is a set of Events in which a UI stereotype may have interest, depending on the state of the UI. The Element Events are predefined for each HCI Element.

A state should have an identification, and within a UI stereotype, a single state should be the initial state of the UI. A UI stereotype has many states and during ISUI execution it should be in a state, enabling the UI to be *quasi* modal, i.e., the UI enters in a state in which the user needs to do some physical action in order to keep the system in such a state. This state is, in general, associated to user error prevention, such as confirmation of critical operations, for instance.

Given the occurrence of an event in a certain state, its behavior is executed via an Action. For each element event (or stereotyped event) in a state, there should be a corresponding action. An Action consists of a set of commands, restricted by conditions, that should be executed in the moment that an event occurs. Thus, an action characterizes a behavior executed after an user interaction.

There are two types of actions: Interface actions and External actions. The first is executed by the UI, since the external IS application does not have to be aware of its execution. Two notable kinds of interface action are navigation actions, which supports the transition between UI presentations, and state transition actions, which enable the transition of states in the same UI presentation.

External actions are implemented by the underlying IS application, or by another external system, since the action defers the binding of the concrete function to IS run time, but all the actions are specified in UI modeling time. Thus, the action behavior is well known and the actions can be reused whenever needed, since they are specified in isolation from the concrete UI behavior. For example, an action may be modeled to perform a query on a particular kind of business information; this abstract action can be reused in several UI stereotypes, and can be implemented in different ways in distinct IS.

The connection between UI elements and behavior, which is the core goal of a UI stereotype, is represented through the correlation between events captured by presentation elements and actions triggered by the UI stereotype that describes the UI action or application behavior in response to the event.

3.2. User Interface Stereotype

Most IS have common features and requirements to present information. A **UI Stereotype** captures some specific ISUI similarities, modeling UI presentation and behavior that are recurrent in different IS applications [da Costa et al. 2010]. This concept integrates and extends two previous concepts of MBUI: Design Context [van Welie and van der Veer 2003] and Screen Layouts [Neil 2010]. Design context is related to UI intention, or the kind of application developed, and presents some applications to certain domains, separated in three categories: Types of Sites, User Experience and Types of Pages. Screen Layouts [Neil 2010] abstract common UI presentation formats that are independent of the application domain, such as spreadsheet, form and portal.

The UI Stereotype enables to abstract interactions and tasks of the user as well

as the form of presentation of information for a particular task. Every UI Stereotype determines the form of presentation of information for a given need of IS and the UI behavior for every user interaction with that presentation.

A UI stereotype must have at least one HCI element, that characterizes a Stereotyped Element, and should organize its elements considering presentation and behavior properties. It is possible to define complex HCI elements by composing UI stereotypes, that is, placing a UI stereotype as an HCI element of another UI stereotype. These elements are populated according to business concepts of the selected application. Simple UI stereotypes, such as ordered lists of IS concepts, can generate complex and significant UI stereotypes to IS, such as composite queries, spreadsheets and master-detail reports.

Since an HCI Element is associated to a UI stereotype, this element can extend element events with Stereotyped Events. The UI actions are predefined by the UI stereotype. Thus, all instances of a given UI stereotype execute the same behavior. Actions which depend on the application or other external system are specified as external actions by the UI stereotype.

Each HCI Element that belongs to UI Stereotypes is a Stereotyped Element, and each stereotyped element can have a different behavior, according to the UI stereotype. For example, an Open Question HCI element in a Form Stereotype may trigger an action that populates another Question HCI element. However, an Open Question HCI element in a Login Stereotype may trigger a validation action. A UI Stereotype should have a set of states, i.e., all UI stereotype instances should have the same set of states and the same actions. The difference is in the implementation of interaction elements and external actions, that defines peculiarities required by each application.

3.3. Presentation Metamodel

The HCI Metamodel is abstract enough to describe UI independent of a specific technology. To complete the generation of UI code, information of the target computational platform must be considered. The Presentation Metamodel deals with this issue. It is worth noting that the Presentation Package of the HCI Metamodel deals with abstract definitions of UI appearance, while the Presentation Metamodel aims on mapping both abstract appearance and behavior to concrete UI components to allow code generation.

Figure 2 introduces the main concepts of the Presentation Metamodel. It is composed by **Templates**, which are concrete representations of UI Stereotypes in a specific platform. The external package Application represents the business logic of the underlying IS. A Template is referred as a visual description of a presentation of an application or document, standardizing its appearance [Xudong and Jiancheng 2007]. In our work, this concept is extended to represent not only the appearance of ISUI with the same intention, but also the expected behavior of the ISUI.

A Template is a composition of Panels. Each Panel is a concrete representation of a HCI Element that must transmit information or receive user interactions. There are four kinds of Panels: Input Panel, Navigation Panel, Output Panel and Composite Panel. The Input Panel receives user information to be stored in the IS applications. The Navigation Panel triggers actions according to the user interactions. The Output Panel presents information to the user. The Composite Panel allows composition of panels.

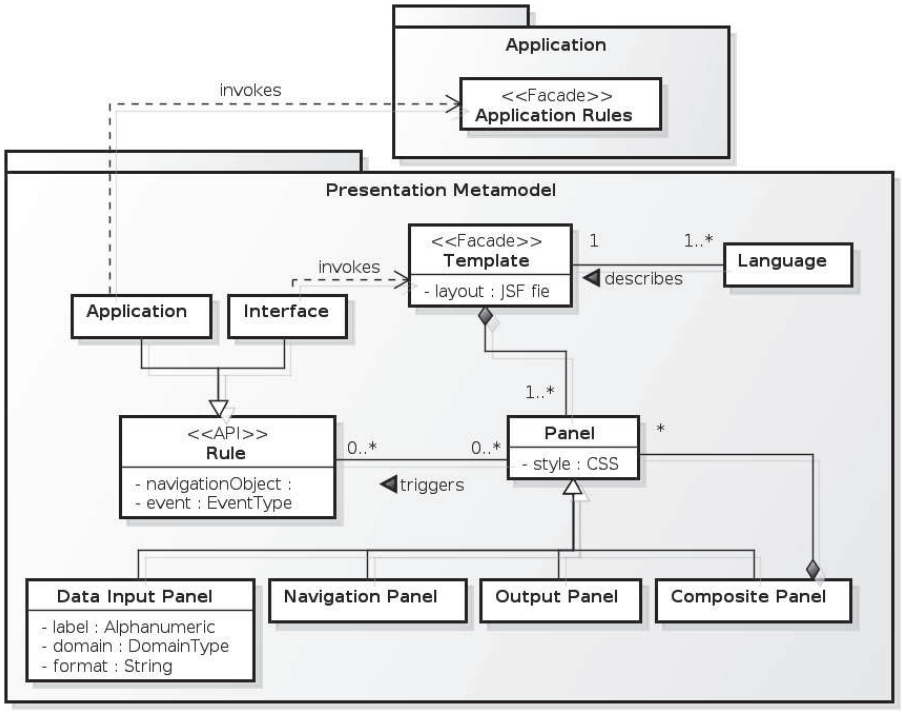


Figura 2. Presentation Metamodel

Style and Layout concepts must be considered in the concrete level of the Presentation Metamodel. Thus, they are associated to the Panel concept. Style is the UI appearance description considering size, font, colors, borders, and other renderization elements. They must be described in CSS (*Cascating Style Sheets*) format [Bost et al.]. Layout is related to UI panels positioning. A JSF (*JavaServer Faces*) file [Oracle] must have each panel position. CSS information complement the layout description.

There are two kinds of Rules that emerge from the Action concept in the abstract level: Interface Rules and Application Rules. Interface rules are completely resolved by the UI code, thus they do not rely on application logic to be executed. Application rules, on the other hand, are implemented by code that is external to the UI, that is, IS applications code or other external system code.

The UI behavior is specified conceptually through rules that define an API, independently from their form of implementation. Application Rules should be implemented by external systems or IS applications, which must implement the facade API defined by the Application package. In this way, every interaction that requires an application functionality must have a behavior definition in modeling time, but the implementation of this behavior will only be decided in run time.

The application or Template implements the rules defined in the API stereotype, and these rules must be invoked through a facade, isolating business logic and application. Thus, the IS generation becomes integrated to HCI viewpoint, while is decoupled, since every system aspect (UI and application) is handled separately.

A mechanism must interpret the concrete UI description in order to generate the final UI for execution in a specific platform. An architecture of components for UI mana-

gement has been implemented and integrated with an existent application framework for IS [Almeida et al. 2009, de Oliveira et al. 2011, Graciano Neto and de Oliveira 2013]. However, due to space limitations, it is not possible to discuss the details of this architecture here. A forthcoming paper will describe these details.

4. Case Study: the Survey Stereotype

A case study was conducted to validate the effectiveness of the UI stereotype concept as the basis for MBUI development. We defined a UI stereotype to represent a common type of IS application, the survey form. This stereotype is used in several contexts, but always with the intention of investigating, collecting or evaluating information about a specific topic. Every survey consists of a set of questions and each question is associated with one or more possible answers.

An instance of the HCI metamodel highlights the presentation and behavior features of a Survey UI Stereotype. The essential concepts of IS Surveys were modeled as a Domain Metamodel Instance [da Costa et al. 2010], whose main aspects are presented in Figure 3. A survey application UI consists of questions, answers and services related to a specific area of knowledge, which is the subject of the survey.

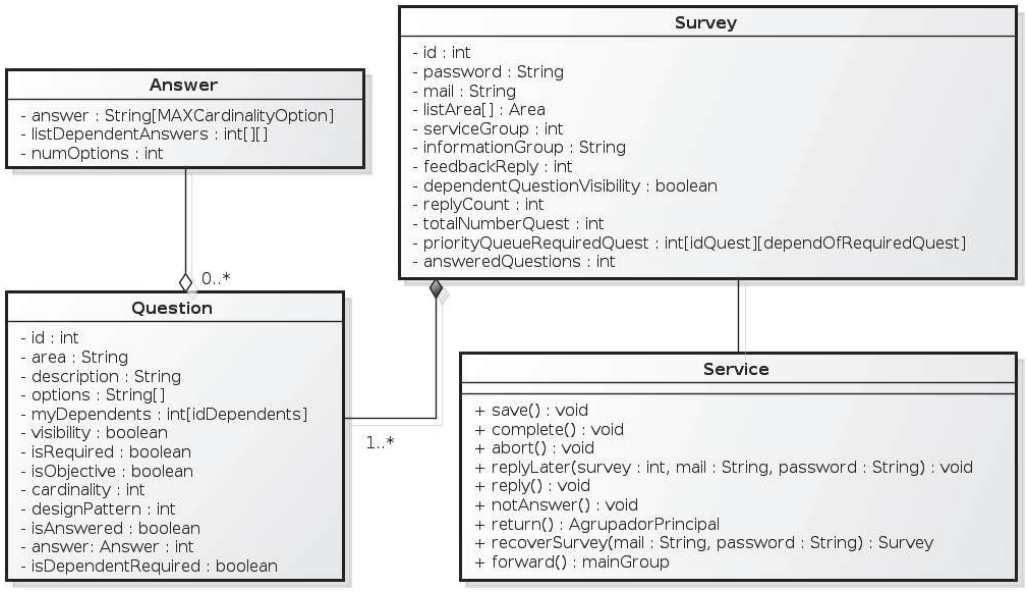
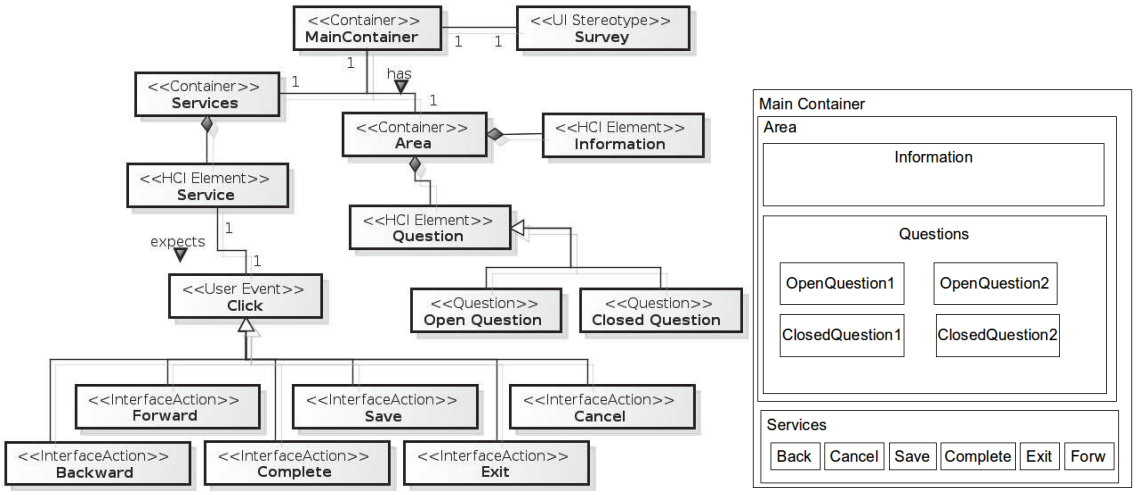


Figura 3. Domain Model for Survey Applications

Each Survey UI Stereotype can be instantiated for different IS applications, such as enrollment, socioeconomic or evaluation surveys. Questions can be open or closed. Closed questions restrict the range of possible answers. Furthermore, the answer of a question can redefine the remaining questions, i.e., each answer may define a different set of following questions.

Figure 4 summarizes the idea of an HCI Metamodel for Survey UI Stereotype. Figure 4a present the Survey modeling. It defines a main container to group the entire UI within the stereotype, a container for data exchange (areas, questions and informations), and a container for services presentation. Six UI actions were considered in this model: save, cancel, complete, exit, forward and backward.



(a) HCI Model for Survey UI

(b) Appearance of Survey

Figura 4. Survey UI Stereotype

A Survey UI Stereotype can be implemented with different layouts by mapping the interaction elements of the stereotype to panels in the Presentation Metamodel and transforming these panels in XHTML files. These files should be available to use during the execution of an instance of the UI stereotype. The templates used for UI automatic generation deal with this transformation. Figure 4b illustrates a layout of a Survey.

The Survey is a relatively simple IS application class which allows the creation of a comprehensive UI model. Modeling this type of application with a Survey UI Stereotype illustrates the feasibility of the proposed approach, since it is a widely used IS application. Building this kind of UI application based on the UI stereotype approach contributes to productivity, consistency and standardization of surveys in a variety of IS.

5. Conclusions and Future Work

This paper presented an approach to automatic generation of Information Systems User Interfaces (ISUI). The main contributions of the paper are: (1) the definition of a HCI Metamodel which uses the UI Stereotype concept to support requirements for the description of ISUI, enabling decomposition of UI components and the description of their behavior; (2) a set of metamodels that describes both presentation and behavior aspects of the UI in different abstraction levels, supporting automatic building of ISUI; and (3) the integration between the software architecture of UI and other components of IS applications, relating this approach with the whole IS development process.

A case study was described to illustrate the viability of the proposed approach by modeling the UI of survey applications, a common component of many IS. A UI Stereotype was defined and applied to automatically build the UI of this kind of IS application, allowing that programmers only configure each HCI element and implements the specific functions of an application.

The proposed method involves all quality criteria indicates in Table 1. The concept of UI Stereotype allows the use of a database with several instances of UI Stereotypes. The use of UI Stereotypes promotes standardization, consistency, productivity and reuse

of ISUI. Abstracting the UI behavior through UI Stereotypes enables the abstract elements to be instantiated in several IS. It enhances productivity, since the UI stereotype instances makes UI building faster because of presentation and behavior reuse of ISUI which have the same intention. Moreover, UI stereotype promotes standardization and consistency of each instance of a UI stereotype that have the same intention.

The MBUI development approach described in this paper was conceived within a research project conducted by the Databases and Software Engineering Research Group of INF/UFG, and integrates results of many subprojects. Future works within this research are related to the investigation of usability issues to add new capabilities to the current metamodels. Moreover, guidelines should be established to support the designer on the conception of new UI stereotypes. Applying the proposed approach to other types of IS applications is also important to assert its comprehensiveness.

Referências

Ahmed, S. and Ashraf, G. (2007). Model-Based User Interface Engineering with Design Patterns. *Journal of Systems and Software*, 80:1408 – 1422. Elsevier Science Inc.

Almeida, A. C., Boff, G., and de Oliveira, J. L. (2009). A Framework for Modeling, Building and Maintaining Enterprise Information Systems Software. In *Anais do XXIII Simpósio Brasileiro de Engenharia de Software*, pages 115 – 125, Fortaleza, Brasil. IEEE Computer Society.

Blouin, A. and Beaudoux, O. (2010). Improving modularity and usability of interactive systems with Malai. In *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing System*, pages 115 – 124, Berlin, Germany. ACM.

Bost, B., Lie, H. W., Lilley, C., and Jacobs, I. Cascading Style Sheets, level 2 - CSS2 Specification. Available in: http://www.w3.org/TR/2008/REC-CSS2-20080411/2.

Cadavid, J. J., Quintero, J. B., Lopez, D. E., and Hincapié, J. A. (2009). A domain specific language to generate web applications. In *Memorias de la XII Conf. Iberoamericana de Software Engineering (CIbSE 2009)*, pages 139 – 144, Medellín, Colombia.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289 – 308. Computer-Aided Design of User Interface.

Costa Neto, M., Souza, A., Lavor, R., Silva, C., and Leite, J. (2009). Desenvolvendo interfaces de usuário multiplataformas utilizando MDA. In *Anais do Primeiro Congresso Regional de Design de Interação*, pages 26 – 34, São Paulo. IXDA - SP.

da Costa, S. L. (2011). Uma abordagem baseada em modelos para construção automática de interfaces de usuário para sistemas de informação. Dissertação de mestrado, UFG.

da Costa, S. L., Graciano Neto, V. V., Loja, L. F. B., and de Oliveira, J. L. (2010). A Metamodel for Automatic Generation of Enterprise Information Systems. In *Anais do I CBSoft: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 45 – 52, Salvador, BA, Brasil. UFBA.

Da Silva, W. C. and de Oliveira, J. L. (2009). Gerência de interface homem-computador para sistemas de informação empresariais: uma abordagem baseada em modelos. *iSys - Revista Brasileira de Sistemas de Informação, Vol. 2, 2009*.

de Oliveira, J. L., Loja, L. F. B., da Costa, S. L., and Graciano Neto, V. V. (2011). Um componente para gerência de processos de negócio em sistemas de informação. In *VII Simpósio Brasileiro de Sistemas de Informação*, Salvador, BA, Brasil. UFBA.

Falb, J., Popp, R., Rock, T., Jelinek, H., Arnautovic, E., and Kaindl, H. (2007). Fully-automatic generation of user interfaces for multiple devices from a high-level model based on communicative acts. In *Proc. of the 40th Annual Hawaii Intern. Conference on System Sciences*, pages 26 – 35, Washington, DC, USA. IEEE Comp. Society.

Graciano Neto, V. V. and de Oliveira, J. L. (2013). Evolução de uma Arquitetura de Framework de Aplicação para Sistemas de Informação com Desenvolvimento Dirigido por Modelos. In *Anais do IX Simp. Brasileiro de Sistemas de Informação*, pages 1–12.

Limbourg, Q. and Vanderdonckt, J. (2003). *The Handbook of Task Analysis for Human-Computer Interaction*, chapter 6. Mahwah. Lawrence Erlbaum Ass.

Melia, S., Gomez, J., Perez, S., and Diaz, O. (2008). A model-driven development for gwt-based rich internet applications with ooh4ria. In *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, pages 13–23.

Memmel, T. and Reiterer, H. (2009). Model-Based and Prototyping-Driven User Interface Specification to Support Collaboration and Creativity. *Intern. Journal of Universal Computer Science - Special Issue on New Trends in HCI*, 14(19):3217–3235.

Neil, T. (2010). Rich Internet Application Screen Design. *UX Magazine*, (483). Available in: http://uxmag.com/articles/rich-internet-application-screen-design.

Oracle. JavaServer Faces Specification. Available in: http://javaserverfaces-spec-public.java.net/.

Popp, R., Falb, J., Arnautovic, E., Kaindl, H., Kavaldjian, S., Ertl, D., Horacek, H., and Bogdan, C. (2009). Automatic Generation of the Behavior of a User Interface from a High-Level Discourse Model. *Hawaii Intern. Conf. on System Sciences*, 0:1 – 10. IEEE Comp. Society.

Valverde, F., Panach, I., and Pastor, O. (2007). An abstract interaction model for a MDA software production method. In *Tutorials, posters, panels and industrial contributions at the 26th Intern. Confer. on Conceptual modeling*, pages 109–114. Australian Computer Society, Inc.

van Welie, M. and van der Veer, G. C. (2003). Pattern languages in interaction design: Structure and organization. In *Proc. of INTERACT 2003*, pages 1–5, Zurich, Switzerland. IOS Press.

Vanderdonckt, J. (2008). Model-Driven Engineering of User Interfaces: Promises, Successes, Failures, and Challenges. In *Proc. of Annual Romanian Conference on Human-Computer Interaction ROCHI'08*, pages 1–10, Bucarest, Romania. Matrix-ROM.

Vanderdonckt, J. (2005). A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In *Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05*, pages 16 – 31, Porto, Portugal. Springer.

Xudong, L. and Jiancheng, W. (2007). User Interface Design Model. *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, ACIS International Conference on*, 3:538 – 543. Los Alamitos, CA, USA.