Towards Mappings and Models Transformations for Consistency of Plastic User Interfaces

Jean-Sébastien Sottet, University of Grenoble, CLIPS-IMAG Lab., 385, rue de la Bibliothèque, BP53, 38041 Grenoble cedex 9, France, jean-sebastien.sottet@imag.fr

Gaëlle Calvary, University of Grenoble, CLIPS-IMAG Lab., 385, rue de la Bibliothèque, BP53, 38041 Grenoble cedex 9, France, gaelle.calvary@imag.fr

Jean-Marie Favre, University of Grenoble, LSR-IMAG Lab, 681, rue de la Passerelle, BP 72, 38402 Saint Martin d'Hères Cedex, France, jean-marie.favre@imag.fr

Joëlle Coutaz, University of Grenoble, CLIPS-IMAG Lab., 385, rue de la Bibliothèque, BP53, 38041 Grenoble cedex 9, France, joelle.coutaz@imag.fr

Alexandre Demeure, University of Grenoble, CLIPS-IMAG Lab., 385, rue de la Bibliothèque, BP53, 38041 Grenoble cedex 9, France, alexandre.demeure@imag.fr

Developing many variants of a same User Interface (UI) on different platforms is costly, may result in inconsistent behavior and does not address the problem of the variability of the context of use in ubiquitous computing. As a result, a new property has been introduced in Human-Computer Interaction (HCI): the plasticity property. In HCI, plasticity refers to the ability of a UI to withstand variations of context of use while preserving usability. A context of use is defined as a triple < User, Platform, Environment >. This paper shows how Model Driven Engineering (MDE) can be used for reasoning on both the design and execution of UIs at different levels of abstraction. These levels of abstraction define different perspectives on a same UI. They are linked together through mappings that tell their properties. Keeping this net of models and mappings at runtime is powerful for solving plasticity as a models or mappings transformation that preserves properties. This paper elaborates on the consistency property.

INTRODUCTION

The need of models in Human Computer Interaction (HCI) has been recognized for long. Nevertheless full automatic generation of User Interfaces (UI) rapidly shown its limits [1]. That does not mean that model-based techniques are not valuable, but that they have to be further and differently explored. We identify two ways: (1) investigating reverse and cross engineering instead of being limited to forward engineering; (2) keeping the models alive at runtime to make the design rationale available at runtime and so to be able to reason about it when the context of use changes. This paper investigates this second point.

Model Driven Engineering (MDE) [2] advocates the systematic use of “productive” models that can be processed by the machine. Full engineering processes are described with explicit models that are linked together through explicit mappings and transformations [3]. This paper investigates MDE for both the development and execution of plastic UIs.

In HCI, plasticity denotes the capacity of a UI to withstand variations of context of use while preserving usability. A context of use refers to the triple <User, Platform, Environment>. Plasticity has been introduced to face the *variety*, *variability* and *unpredictability* of the context of use in ubiquitous computing. For instance, from the platform dimension perspective, in the vision of ubiquitous computing, any object of the physical world may play the role of interaction resources (e.g., walls and tables as display surfaces). As a result, UIs are no longer confined to a unique desktop, but are distributed and migratable among a dynamic set of interaction resources. In case of heterogeneous resources (e.g., PC and PDA), redistribution may require a graceful remolding to accommodate the UI to the target platform. Whilst product line approaches deal with the production of variants of UIs (e.g. a UI specifically crafted for a PC or a PDA), plasticity is much more challenging coping with the change of the context of use and the need of preserving usability. Nevertheless, they share one issue (at least): consistency.

Whilst code-centric approaches might be suited for the development of simple and single UIs, they simply fail for plasticity. Our approach is to revisit model-based approaches taking benefit from recent advances in MDE. Our vision is to model an interactive system at different levels of abstraction and dynamically maintain the mappings between these levels. The mappings tell the properties they guarantee and as a result serve the adaptation process. Consistency is one of the properties.

This paper is twofold. Section 1 introduces a MDE framework for sustaining our vision. Section 2 focuses on mappings and transformations for ensuring consistency when adaptation occurs.

A MDE FRAMEWORK FOR PLASTIC UIS

The core concepts of MDE are development processes, models and metamodels, mappings and transformations. This section focuses on development processes, models and metamodels. Mappings and transformations are discussed next.

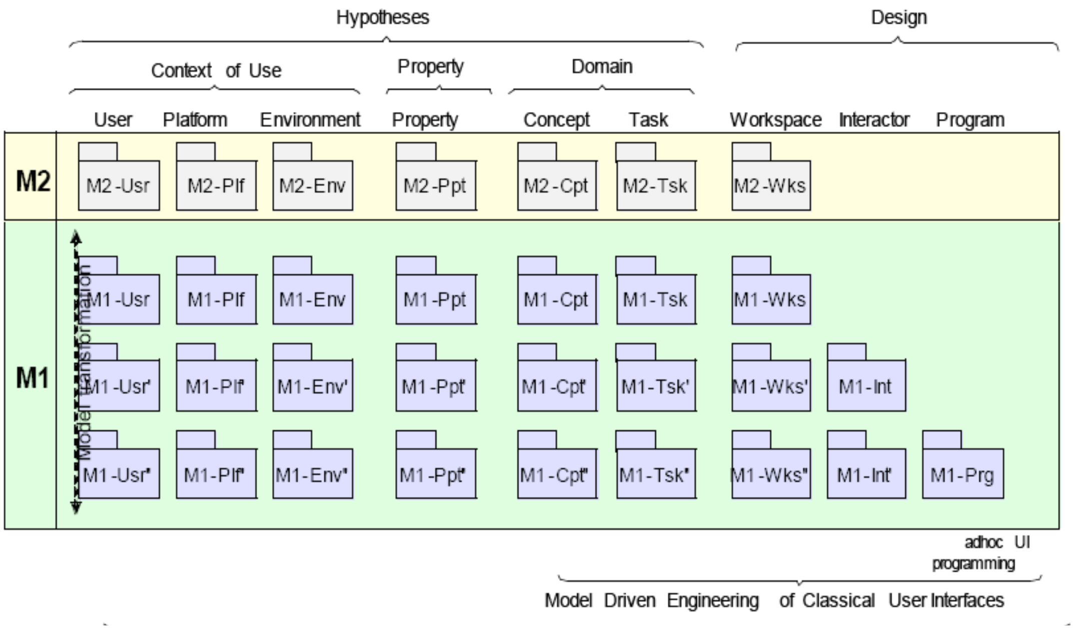

Figure1. A MDE Framework for the development of UIs

DEVELOPMENT PROCESSES

Current industrial practices are still mostly code-centric (right bottom of Figure 1). At the opposite, MDE processes are based on successive models refinements, integrating new information step by step [5]. They start at an entry point (in a top-down approach, the task model) until reaching an exit point (the final running program in case of forward engineering). Typical forward development processes start with the task and the domain models. Then, based on a design know-how, the UI is progressively refined in terms of workspaces, interactors and finally program elements that are this time dependent on the target platform and the available libraries. As suggested in Figure 1, models are revised at each step to take into account the constraints related to the current level of abstraction (for instance, M1-Task can be revised into M1-Task’ and M1-Task’’ according to new constraints).

Product line processes favor both the factorization of the product common parts and the decoration (annotation) of specific parts. Plasticity goes one step further for coping with the change of the context of use. This calls for new (meta)models as explained below.

(META)MODELS FOR PLASTIC UIS ENGINEERING

As explained in CAMELEON [5], five (meta)models structure the development process of classical UIs.

- Task. A task refers to “a goal, together with some procedure or set of actions that will achieve the goal”. From a MDE perspective, a task model can be described as a tree made of binary and unary operators.

- Concept. A concept is an entity “relevant to users to accomplish tasks in a particular domain”. A concept model can be described as an UML class diagram.
- Workspace. A workspace enables a “set of logically/semantically connected tasks. In graphical UIs, a workspace can be mapped onto a window, a set of panels”.
- Interactor. An interactor is “an abstraction of a software component that allows users to manipulate and/or observe domain concepts and functions”.
- Program. The program is “the UI produced at the very last step of the reification process supported by a multi-target development environment. It is expressed as source code”.

A simplified backbone of four (under elaboration) metamodels is provided in Figure 2. The mappings are discussed in the next section.

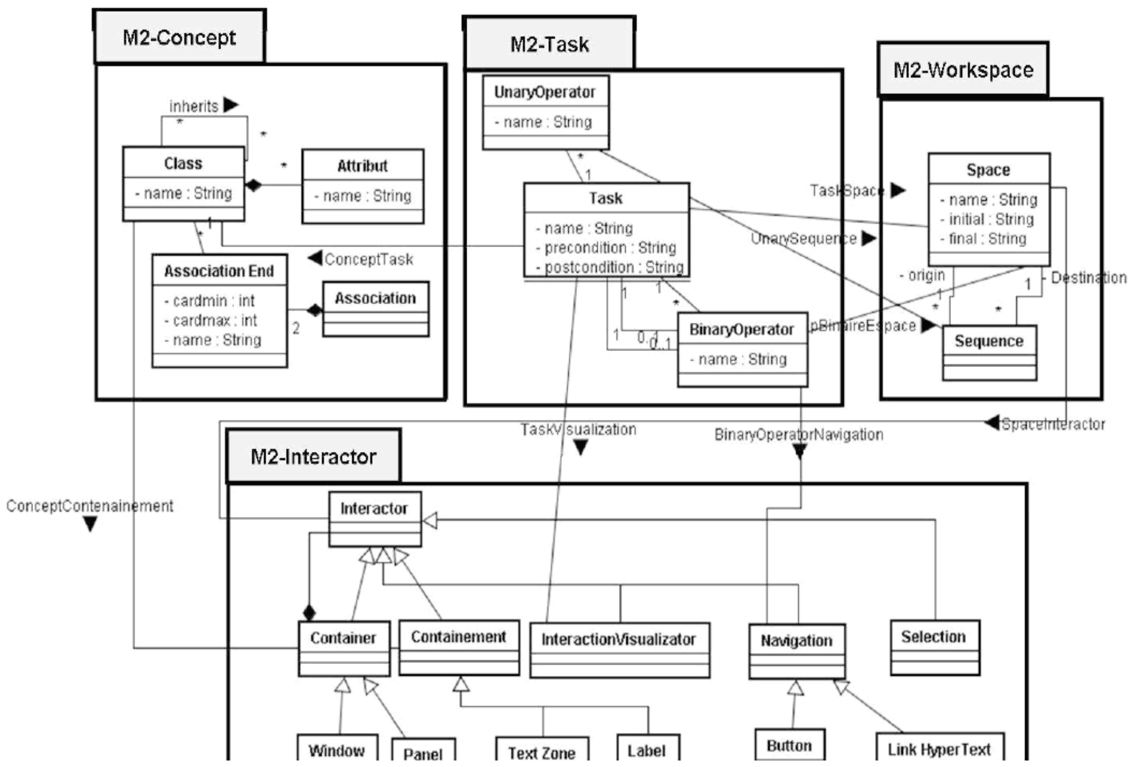

Figure2. A simplified backbone of four metamodels.

As explained in CAMELEON [5], these (meta)models are not sufficient for plasticity. New (meta)models are introduced to describe both the context of use and the expected usability.

- User. The user is “the archetypal set of end-users envisioned for the interactive system”. This model captures general information (e.g., age, gender) as well as skill level.
- Environment. The environment refers to “the physical setting where the interaction takes place. It can be modeled as the set of objects, persons and events that are peripheral to the current activity but that may have an impact on the system and/or users behavior”. From an engineering perspective, the environment can be modeled as a graph of contexts and situations [9].
- Platform. The platform denotes “the set of physical and software resources that function together to form a working computational unit whose state can be observed and/or modified by a human user. It may be an elementary platform or a cluster of platforms”. Platform modeling is a core issue in MDE.
- Usability. Usability refers to “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. Existing frameworks (e.g. Scapin & Bastien [7], IFIP [8]) enumerate a set of properties without localizing them on the map of metamodels (like Figure 2). Whilst some properties are

confined to a unique (meta)model (for instance, the ergonomic consistency that is computable at the interactor level), some others depend on the mappings between different metamodels.

MAPPINGS AND TRANSFORMATIONS FOR CHECKING AND ENSURING CONSISTENCY

Without mappings and transformations, models would be isolated. In this area, MDE promotes extensible libraries of small transformations to be composed. The designer selects and, if necessary, tunes the appropriate transformations. If no transformation is available then a new one can be written thanks to transformation languages. The new one can then be added to libraries. Quite often, transformation engines are limited to the metamodels they manipulate. HCI could take benefit from software engineering transformation environments not devoted to HCI.

We use generic MDE techniques and extensible libraries of transformations and metamodels. Whilst emerging standards for expressing MDE transformations are under active development (e.g. QVT), we investigate [4] the appropriateness of a generic MDE transformation language for plasticity. This language should be able to describe both the mappings and transformations that link together elements of metamodels (for instance in Figure 2, the association Class-Task links together the Concept and Task metamodels). Based on this generic transformation language, we are elaborating libraries for forward engineering and plasticity in HCI. Now, we envision the integration of consistency along the mappings and transformations.

MAPPINGS AND CONSISTENCY

The mapping problem has been defined by Puerta and al [10]. It states that mappings are the key for acceptable model-based UIs. Some tools like DynaMo-Aid [11] demonstrate the benefit of mappings at design time. The goal is now to ensure consistency at runtime when the context of use changes. From our understanding, the key lies in the mappings:

- Mappings connect together a set of metamodels, at least two. Each mapping describes its role (for instance, the Task *t* manipulates the Concept *c*) and its cost/benefit ratio (for instance, the interaction length for performing the task). By nature, the role can deal with either functional or structural concerns.
- A functional mapping is a coupling between models that produces a new function. For instance, a label “Name” coupled with an input field provides the end-user with the function “Specify name”. For plasticity, the coupling with the physical entities (mouse, keyboard, table, etc.) is a key issue as the user evolves in a changing environment. These functional mappings will help for reasoning about consistency at a structural level.
- A structural mapping links together models in order to propagate any relevant modification that occurs in one model to the connected ones. We identify two kinds of structural mappings whether they are exogenous (i.e., involving two metamodels at least) or endogenous (i.e., limited to a given metamodel). Deleting a task *t* should suppress the corresponding interactors (exogenous). Interactors should be compliant to a given style to provide the end-user with the feeling of a global consistency (endogenous). Whatever the mapping is (either endogenous or exogenous), it should convey its *elasticity*, i.e., the extent to which it can be tuned when adapting the UI to its context of use.

CONSISTENCY DRIVEN TRANSFORMATIONS

The core idea is to drive the transformations and support consistency by the descriptions that are embedded in the functional and structural mappings. When the context of use changes, the net of models and mappings is carefully transformed without breaking the elastic mappings for as long as possible staying within the elasticity domain of the mappings. Advances in graceful degradation [12] could be relevant in this area.

Mappings should either locally embed their transformations to maintain consistency (close transformations) or be able to request and use external transformations (open transformations). Of course, mappings have to describe their open/close capacities.

CONCLUSION

Models are not new in HCI but they traditionally were limited to a poor forward engineering. The poor quality of the produced UIs was disappointing. In this paper, we revisit models, promoting a dynamic net of models in which mappings and transformations play a central role. Mappings and transformations support the description and management of consistency. Transformations are performed with respect to consistency.

Actually, we are dealing with three open issues: (1) modeling consistency along the net of models and mappings; (2) investigating the observability and control of models, mappings and transformations by the end-user (we call this UI the *meta-UI*); (3) exploring MDE advances to avoid the production of ad-hoc and technological domain dependent solutions.

REFERENCES

[1] Myers B., Hudson S.E., Pausch R. "Past, Present, and Future of User Interface Software Tools", Transactions on Computer-Human Interaction (TOCHI), Vol 7, Issue 1, 2000

[2] Planet MDE, "A Web Portal for the Model Driven Engineering Community" http://planetmde.org

[3] Favre J.M., "Foundations of Model (Driven) (Reverse) Engineering", Dagsthul Seminar on Language Engineering for Model Driven Development, DROPS, http://drops.dagstuhl.de/portals/04101, 2004

[4] Sottet, J.S., Calvary, G.,Favre, J.M., Toward Model Driven Engineering of Plastic User Interfaces. International workshop on Model Driven Development of Advanced User Interfaces, MDDAUI, Jamaica, 2005

[5] Calvary G., Coutaz J. Thevenin, D. Limbourg, Q., Bouillon, L., Vanderdonckt J. "A Unifying Reference Framework for Multi-Target User Interfaces, Interacting With Computers, 2003

[6] Demeure, A., Calvary, G., Sottet, JS.,Vanderdonkt, J. A Reference Model for Distributed User Interfaces TAsk MOdels and DIAgrams for user interface design, Gdansk, 2005

[7] Scapin D., Bastien, C.H., "Ergonomic Criterias for Evaluating the Ergonomic Quality Interactive Systems." Behaviour and Information Technologies, Vol 16, 1997

[8] Abowd G., Coutaz J., Nigay L., "Structuring the Space of Interactive System Properties", Proceeding of the IFIP, 1992

[9] Crowley, J., Coutaz, J., Rey, G. , Reignier, P., Perceptual Components for Context-Aware Computing, UbiComp 2002:, Göteburg, Sweden Sept./Oct. 2002

[10] Puerta, A., Eisenstein, J. Toward a General Computational Framework for Model-Based Interface Development Systems. International Conference on Intelligent User Interfaces, Los Angeles 1999

[11] Clerckx, T., Luyten, K., Coninx, K. The Mapping Problem Back and Forth: Customizing Dynamic Models while preserving Consistency. TAsk MOdels and DIAgrams for user interface design, Prague, 2004

[12] Florins, M., Vanderdonkt, J. Graceful Degradation of user Interfaces, Intelligent User Interfaces, ACM, 2004