Mapping Model: A First Step to Ensure Usability for sustaining User Interface Plasticity

Jean-Sébastien Sottet[12] Gaëlle Calvary[1] Jean-Marie Favre[2]
University of Grenoble, CLIPS[1] and LSR[2] Labs
385, Rue de la Bibliothèque, BP53, 38041 Grenoble cedex 9, France
jean-sebastien.sottet@imag.fr gaelle.calvary@imag.fr jean-marie.favre@imag.fr

ABSTRACT

Ubiquitous computing has introduced the need for interactive systems to be able to adapt to their context of use (<User, Platform, Environment>) while preserving usability. This property has been called *plasticity*. Until now, efforts have been put on the functional aspect of adaptation, neglecting the usability part of the definition. This paper investigates MDE mappings for embedding both the description and control of usability. It first provides a general definition and metamodel of the notion of "mapping" that are not devoted to Human-Computer Interaction (HCI). A mapping describes a transformation that preserves properties. A transformation is performed by a set of transformation functions that can be described either by a function and/or an execution trace. The mappings properties provide the designer with a means for both selecting the most appropriate transformation functions and previewing the resulting design. When applied to HCI, mappings are an easy way for both describing and controlling ergonomic criteria either at design time or runtime. Mappings are rubber bands that link together different perspectives of a same User Interface (UI). They break when the UI goes outside its plasticity domain.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.* The ACM Computing Classification Scheme: http://www.acm.org/class/1998/

General Terms

Algorithms, Design, Human Factors, Standardization, Languages, Theory.

Keywords

Model, Metamodel, Mapping, Model transformation, Advanced User Interfaces, Plasticity, Usability, Adaptation.

1. INTRODUCTION

In Human-Computer Interaction (HCI), plasticity refers to the ability of a User Interface (UI) to withstand variations of context of use (<User, Platform, Environment>) while preserving usability. Until now, efforts have been put on the functional aspect of adaptation. Model Driven Engineering (MDE) has been seen as promising [3] [10]. At MDDAUI'05, we presented a MDE approach promoting the description of a UI as a net of models and mappings (called octopus) [17]. In this paper, we go one step further investigating the usability part of the plasticity definition. We show how usability can be described and controlled along the mappings that compose a UI (the octopus legs). The idea was roughly sketched in [18].

The paper is threefold. In a first section, it provides a short reminder of the octopus vision and a basic case study for illustration. Then, it elaborates a general definition and metamodel of the notion of "mapping" that are not devoted to HCI but applicable to the domain as demonstrated on the case study. Finally, the paper opens a discussion on issues and perspectives in the areas of advanced UIs and MDE in general.

2. TOWARDS OCTOPUSES

Taking benefit from the past in HCI, the idea is to describe a UI as a net of models and mappings. The models define different perspectives on a same UI: domain concepts, user's task, workspaces (Wks) and interactors (I) (Figure 1). For their deployment, these models require resources that are supplied either by the functional core (FC) and/or the context of use (in particular, the platform that provides the end-user with input and output devices). Deployment is modeled as a set of mappings (the gray boxes on Figure 1). Models and mappings are compliant to metamodels.

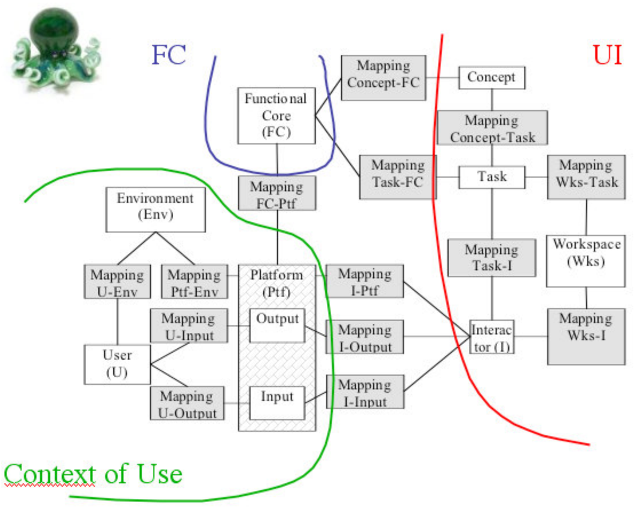

Figure 1. In our MDE vision, UIs look like octopuses. They are net of models whose mappings define the UI deployment on the functional core (FC) and the context of use.

As illustration, let us consider a basic booking system inspired from Nogier's book [13]. For making a reservation, the end-user is supposed to first specify the date, then the period of the day (morning versus evening), and finally the number of seats he/she would like to book. Figure 2 illustrates a sub-part of the corresponding octopus: the mappings between tasks, concepts and interactors. In Figure 2a, dashes have been introduced at the interactor level to make explicit the fact that the task "Specify date" is mapped on two guiding labels ("Date", "mm/dd/yy"). In this case, there is no (human) error protection: text fields do not prevent the end-user from bad entries. In contrast in Figure 2b, the calendar and the radio buttons decrease the risk of error when specifying the date and the period of the day.

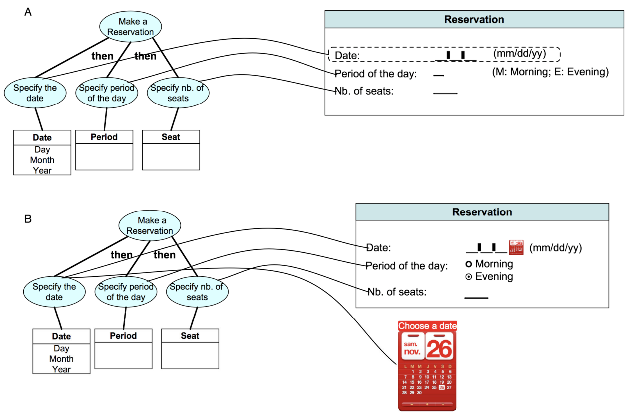

Figure 2. A basic case study illustrating a sub-part of the octopus: the mappings between tasks, concepts and interactors. For legibility, the equivalent dashes for the other mappings in a) and b) have not been drawn.

This paper deals with usability. It shows how usability can be described along mappings. To that end, it provides a general definition and metamodel of mappings that go beyond HCI.

3. MAPPING METAMODEL

Our mapping metamodel is centered on the notion of transformation. Thus, we first define the notions of mappings and transformations before presenting the metamodel.

3.1 Mappings and transformations

In the MDA literature (see Table 1), the term "mapping" is far from being clear. However, it is clearly coupled with transformations.

	Transfo. Instance	Transfo. function	Transfo. model	Transfo. program	Transfo. progr. lang.	Transfo. meta-model	Transfo. interpreter
MDA guide [8]	Transfo.		Mapping		Mapping language		
MDA Distilled [9]	Mapping		Mapping function				
MDA explained [10]	Transfo., Mapping		Transfo. definition				Transfo. tool
QVT DTSC [11]	"Tracking"		Transfo.			Transfo. model	Transfo. engine
QVT Partners [12]			Transfo. Relation	Mapping Relation			
[13]	Transfo.		Transfo pattern				
[14]		Mapping	Model of mapping		Mapping formalism		
[15]				Transfo spec.			Transformer
[16]	Transfo. process		Transfo descr.				

Table 1. A confusing literature on "mappings" and "transformations" terms.

Figure 3 aims at clarifying the situation according to [5]. In particular, it defines the labels of the columns of Table 1.

On Figure 3, "f(x)=x+2" is a *transformation model* that is compliant to a mathematical metamodel. A transformation model describes (the μ relation) a *transformation function* in a predictive way: in our example, {(1,3),(2,4),(3,5)…} for "f" when applied to integers. A transformation function is the set of all the *transformation instances* inside the variation domain (here, the integers). A *transformation instance* is a subset (the ε relation) of the *transformation function.* It is the execution trace of the function ("f").

Figure 3 refines the μ relation into μp and μd. These relations respectively stand for predictive and descriptive representations. Predictive means that there is no ambiguity: the transformation model (e.g., "f(x)=x+2") fully specifies the transformation function. Descriptive refers to a qualifier (e.g., "growing"). It is not sufficient for specifying the transformation function, but it is a means for providing additional information. Figure 3 illustrates two kinds of descriptive representations: one that deals with a transformation model ("f(x)>x"); another one that deals with transformation instances ("growing"). In the first case, the description is made a priori versus a posteriori in the second case. A posteriori descriptions are subject to incompleteness and/or errors due to too few samples.

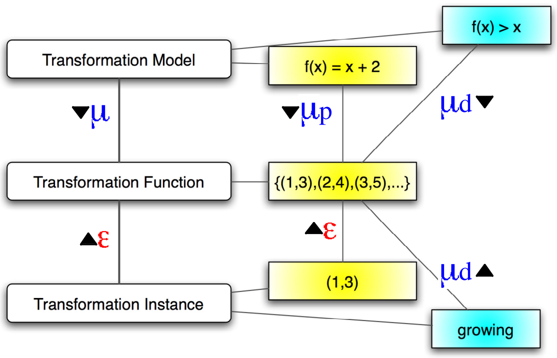

Figure 3. Clarification of the notions of transformation model/function/instance.

Next section provides a metamodel of mappings based on these clarifications.

3.2 A Mapping Metamodel

The metamodel is provided in Figure 4. The core entity is the *Mapping* class. A mapping links together entities that are compliant to *Metamodels* (e.g., Task and Interactor). A mapping can specify *Transformation functions* (e.g., {(Specify date, Date: --/--/-- (mm/dd/yy)), (Specify period of the day, "Period of the day: - (M: Morning; E: Evening)), …}) by patterns. A *Pattern* is a transformation model. It links together source and target elements (*ModelElement*) to provide a predictive description of the transformation function. In addition, a mapping can describe the execution trace of the transformation function. The trace is made of a set of *Links* between *Instances* of *ModelElements*. The couple (Specify date, Date: --/--/-- (mm/dd/yy)) is an example of *Link*.

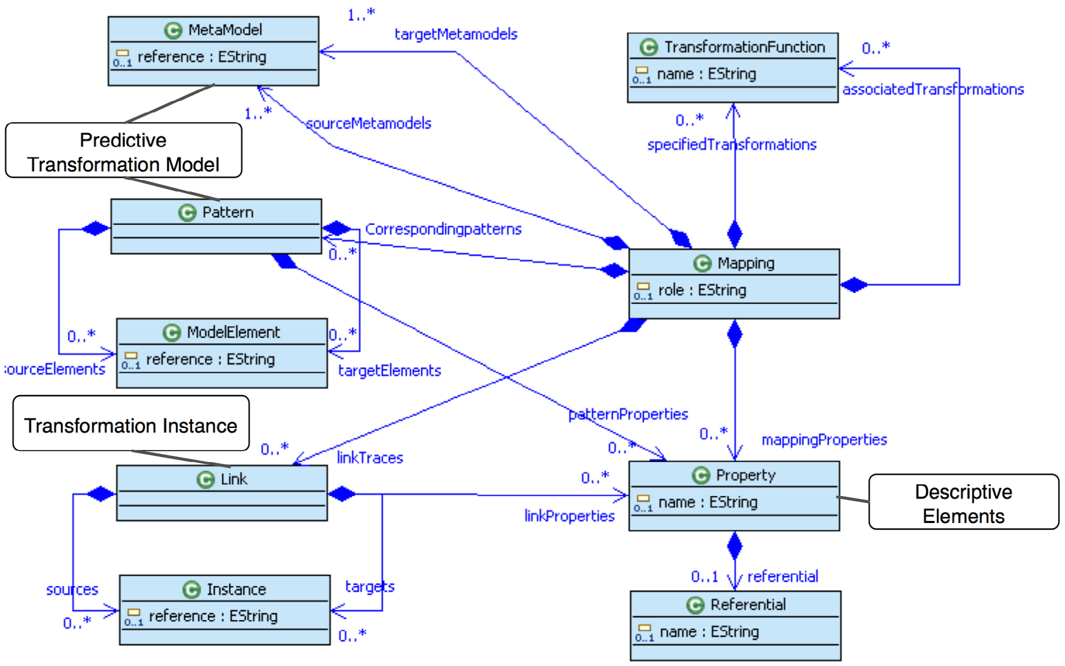

Figure 4. A Mapping MetaModel.

A mapping conveys a set of *Properties* (e.g., "Error protection"). A property is described according to a given *Referential* (e.g., Bastien&Scapin[1] that defines eight criteria among which is the "Error protection"). These properties are descriptive. They qualify either the global set of mappings or one specific element: a mapping, a pattern or a link.

Associated transformations are in charge of maintaining the consistency of the net of models by propagating modifications that have an impact on other elements. For instance, if replacing an interactor with another one decreases the UI consistency, then the same substitution should be applied to the other interactors of the same type. This is the job of the associated functions.

Figure 5 applies the mapping metamodel to the case study according to Bastien&Scapin's referential. Three criteria are considered:

- *Compatibility* to check the extent to with the UI design is compliant to the user's task;
- *Error protection* to measure the extent to which the UI prevents the end-user from bad actions;
- *Homogeneity-Consistency* to ensure a global consistency in the UI (e.g., style).

As pointed out in Figure 5:

- Compatibility is preserved along all the mappings linking together tasks and interactors: the UI fully supports the user's task (Figures 5 a and b);
- Homogeneity-Consistency is satisfied in Figure 5a as the transformation function (that is modeled by the mappings) associates the same type of interactor (input fields) to all the user's actions;
- Error protection is guaranteed in Figure 5b thanks to interactors that preserve the user from mistakes (calendar and radio buttons).

In Figure 5, the scope of compatibility (e.g., C1, C2, C3) is one mapping whilst homogeneity-consistency and error protection deal with the global net of mappings (C4 on a and b).

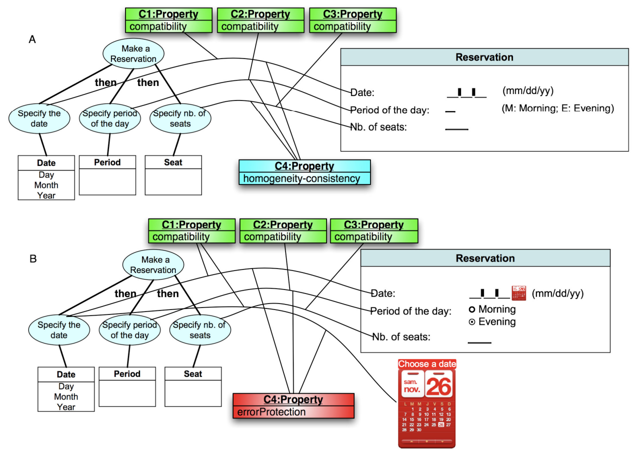

Figure 5. The Mapping Metamodel applied to the case study.

For legibility, Figure 5 only mentions the criteria that are satisfied. For instance, the "Error protection" that is not preserved in Figure 5a has not been mentioned. In reality, octopuses should tell the extent to which each criteria is satisfied (positively or negatively).

This work provides a sound basis for future work. Next section elaborates on perspectives for both HCI and MDE.

4. CONCLUSION AND PERSPECTIVES

In 2000, B. Myers stated that model-based approaches had not found a wide acceptance in HCI. They were traditionally used for automatic generation and appeared as disappointing because of a too poor quality of the produced UIs. He envisioned a second life for models in HCI empowered by the need of device independence. In our work, we promote the use, the description and the capitalization of elementary transformations that target a specific issue.

A UI is described as a net of models and mappings both at design time and runtime. At design time, mappings convey properties that help the designer in selecting the most appropriate transformation functions. Either the target element of the mapping is generated according to the transformation function that has been selected, or the link is made by the designer who then describes the mapping using a transformation model. We envision adviser tools for making the designer aware of the properties he/she is satisfying or neglecting.

At runtime, mappings are the key for reasoning on usability. However, it is not so easy as (1) there is not a unique consensual referential; (2) ergonomic criteria may be inconsistent and, as a result, require difficult trade-offs. Thus, (1) the metamodel will have to be refined according to these refentials; (2) a meta-UI (i.e., the UI of the adaptation process) may be relevant for negotiating trade-offs with the end-user.

Beyond HCI, this work provides a general contribution to MDE. It defines a mapping metamodel and clarifies the notions of mappings and transformations. Mappings are more than a simple traceability link. They can be either predictive (transformation specifications) or descriptive (supported properties), as a result covering both the automatic generation and the hand-made linking. This is new in MDE as most of the approaches currently focus on direct transformation. Our mapping metamodel will be stored in the ZOOOMM project.

5. ACKNOWLEDGMENTS

The work has been supported by the European project EMODE. Authors would like to thank Joëlle Coutaz and Alexandre Demeure for their strong contribution. Metamodels are edited under Topcased plugin for eclipse: http://www.topcased.org.

6. REFERENCES

[1] Bastien J.M.C, Scapin D. Ergonomic Criteria for the Evaluation of Human-Computer Interfaces, Technical report INRIA, N°156, June 1993

[2] Caplat, G., Sourrouille, J.L, *Considerations about Model Mapping*, Wisme 2003

[3] Clerckx, T., Luyten, K., Coninx, K. *The mapping Problem Back and Forth: Customizing Dynamic Models while preserving Consistency*, 3rd International Workshop on Task Model and Diagrams for User Interfaces Design, Prague, Czeck Republic, November 2004, pp 33-42

[4] DSTC, IBM, MOF STC *Query/View/ Transformation*, Submission by DSTC IBM, ad/2003-02-03, March 2003

[5] Favre, J.M. *Toward a Basic Theory to Model Driven Engineering*, Workshop on Software Model Engineering, WISME 2004, joint event with UML 2004, Lisboa, Portugal, October 11,2004

[6] http://zooomm.org

[7] Judson, S.R, France, R.B., Carver, D.L. *Specifying Model Transformation at on the Metamodel Level*, Wisme 2003

[8] Kleppe, A., Warmer, Bast, W. *MDA Explained. The Model Driven Architecture: Practice and Promise*, Addison-Wesley, April 2003

[9] Kurtev, I., Van den Berg, K. *A Synthesis-Based Approach to Transformations s in an MDA Software Development Process,* In Proc. of Model Driven Architecture: Foundations and Applications, pp. 121-126, University of Twente, Enschede, The Netherlands 2003

[10] Limbourg, Q., Vanderdonckt, J. *Adressing the mapping problem in User Interfaces Design*, 3rd International Workshop on Task Model and Diagrams for User Interfaces Design, Prague, Czeck Republic, November 2004, pp 155-163

[11] Mellor, S.J., Scott, K., Uhl, A., Weise, l.D *MDA Distilled: Principles of Model-Driven Architecture*, Addison-Wesley, March 2004

[12] 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

[13] Nogier, J.F. *De l'ergonomie du logiciel au design des sites Web*, Third edition, Dunod 2005

[14] OMG, MDA Guide Version 1.0.1, omg/2003- 06-01, June 2003

[15] Peltier, M. *Techniques transformations de modèles basées sur la méta-modélisation*, PhD, University of Nantes, October 2003

[16] QVT- Partners *Revised Submission for MOF 2.0 Query / Views / Transformation RFP*, http://qvtp.org, August 2003

[17] Sottet, J.S., Calvary, G., Favre, J.M., Coutaz, J., Demeure, A., Balme, L. *Towards Model-Driven Engineering of Plastic User Interfaces*, in Conference on Model Driven Engineering Languages and Systems (MoDELS'05) satellite proceedings, Springer LNCS, pp 191-2005

[18] Sottet, J.S., Calvary, G., Favre, J.M., Coutaz, J., Demeure, A. *Towards Mappings and Models Transformations for Consistency of Plastic User Interfaces* The Many Faces of Consistency. Proc. (CHI2006), Montréal, Québec, Canada, April 22-23, 2006,