Lessons of Experience in Model-Driven Engineering of Interactive Systems: Grand challenges for MDE?

Gaëlle Calvary[1], Anne-Marie Pinna[2]

[1] Laboratoire d'Informatique de Grenoble, Equipe IIHM, 385 Rue de la Bibliothèque, BP 53
38041 Grenoble Cedex 9, France
[2] Laboratoire I3S, Bâtiment ESSI, 650, Route des Colles, B.P. 145
06903 Sophia-Antipolis Cedex, France
Gaelle.Calvary@imag.fr, pinna@polytech.unice.fr

Abstract. Model-based approaches have been recognized as powerful for separating concerns when designing a User Interface (UI). However model-based generation of UIs has not met a wide acceptance in the past. Today, taking benefit from MDE advances, models are revisited in Human Computer Interaction (HCI) for engineering multi-contexts interactive systems. This paper relates some difficulties and requirements highlighted by experience. Among them is the need of support for sustaining the collaboration between models both at design time and runtime. Another requirement is about the capitalization of models, mappings and metamodels for saving the know how in HCI. Last but not least requirement is about evaluation. How shall we evaluate the strengths and weaknesses of MDE for HCI?

Keywords: Human Computer Interaction, Interactive system, User interface, Multi-models, Multi-actors, Multi-views, Capitalization, Evolution, Evaluation.

1 Introduction

Model-based approaches have been widely explored in Human-Computer Interaction (HCI) for long. The motivation was to force designers focus on user's task instead of blending several concerns including cosmetic considerations. Rapidly, task modeling has emerged as a wise starting point when engineering interactive systems. Automatic generators of User Interfaces (UIs) have appeared (e.g., ADEPT [2]) but the poor quality of the resulting UIs killed the approach for long. Later on, the increasing diversity of platforms (e.g., PC or PDA) and the rise of Platform Dependent versus Independent Models (PSM versus PIM) brought models back to life in HCI [4]. Now, we are at the point of blurring the distinction between design time and runtime for making it possible for the end-user to fashion his/her own interactive space according to his/her feelings and needs as well as to the arrival and departure of interaction resources. The purpose of this paper is to relate where we are in the tandem MDE-HCI and which issues need to be solved for going further in Meta-Design [1].

2 Where we are in MDE for HCI

The canonical functional decomposition of interactive systems makes the distinction between the Functional Core (FC) and the UI. From a methodological point of view, along a forward engineering process, the starting point is most of the time a task model which structures the user's goal into sub-goals. Conversely, reverse engineering consists in analyzing legacy systems for recovering the models that may have driven the design (e.g., the task model) and/or implementation (e.g., the architecture model). Using chains of tools (e.g., the UsiXML arsenal of tools - see http://www.usixml.org/), it is possible to hybrid forward and reverse engineering [3].
In this section, we first focus on the UI design, and then address the implementation of the whole interactive system.

2.1 MDE for the design of UIs

Designing a UI means setting all the degrees of freedom (e.g., the structure of the UI, the choice of interactors) based on given requirements (e.g., the user's task and preferences, the ergonomic criteria that have been elicited, the targeted platforms). Fig. 1 shows four functionally equivalent UIs: they support the same user's task T that consists in accomplishing T1 and T2 in interleaving. The UIs differ from an ergonomic point of view. In a, the subtasks T1 and T2 are directly observable whilst they are browsable in b-c-d. Browsing has a human cost: at least one physical action for commuting between subtasks.
Along a forward engineering process, once the task is modeled, decisions have to be made about the structure and rendering of the UI. Decisions are mostly driven by ergonomics (e.g., minimal actions). At this point, the UI is classically a mockup either drawn on paper sheets or prototyped using languages (e.g., Smalltalk) and tools (e.g., JBuilder). Predictive and/or experimental evaluations must be done.

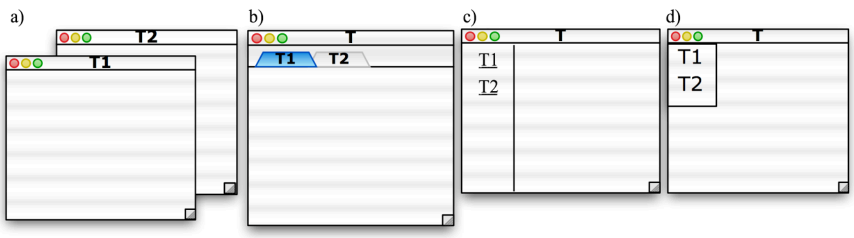


Fig. 1. Four functionally but not non functionally equivalent UIs.

Our lessons of experience are listed below:
MDE for capturing the know-how in HCI: the relevant models (e.g., task, structure, interactors) are identified giving rise to several notations (e.g., CTT [6]), metamodels and versions of metamodels. One open issue specific to HCI is the modeling of ergonomic properties.
MDE for platform independence: a plethora of Platform/Rendering Independent Languages has appeared (e.g., UsiXML, UIML (www.uiml.org), RIML (european CONSENSUS project), XIML (www.ximl.org), XAML (Microsoft), AUIML (IBM)). There is a need of reconciliation to clarify the state of the art.
Miscellaneous statements about MDE in practice: as the focus has mostly been set on

high level models so far (typically the user's task), these models are often overloaded with information typically related to transformation (e.g., the targeted platform). Most of the time, transformations are mono-technological (e.g., HTML). In addition, the rendering can be improved.

2.2 MDE for the implementation of interactive systems

Software architecture models (e.g., ARCH) improve software quality. ARCH refines the UI into sub functions among which is the Dialog Controller (DC). The DC is in charge of piloting interaction while ensuring consistency between the FC and the UI. The DC is a kind of implementation of the task model. General research in MDE deals with the FC only. Below are our lessons of experience about the whole system.

MDE for models collaboration: the global picture (Fig. 2) implies biaxial mappings and transformations: vertical for the engineering process, horizontal for complementary descriptions and/or code [5].

MDE for models at runtime: we are at the point of keeping the design models and design rationale (i.e., the biaxial lattice of Fig. 2) alive at runtime so that to enable the revision at runtime of any design decision.

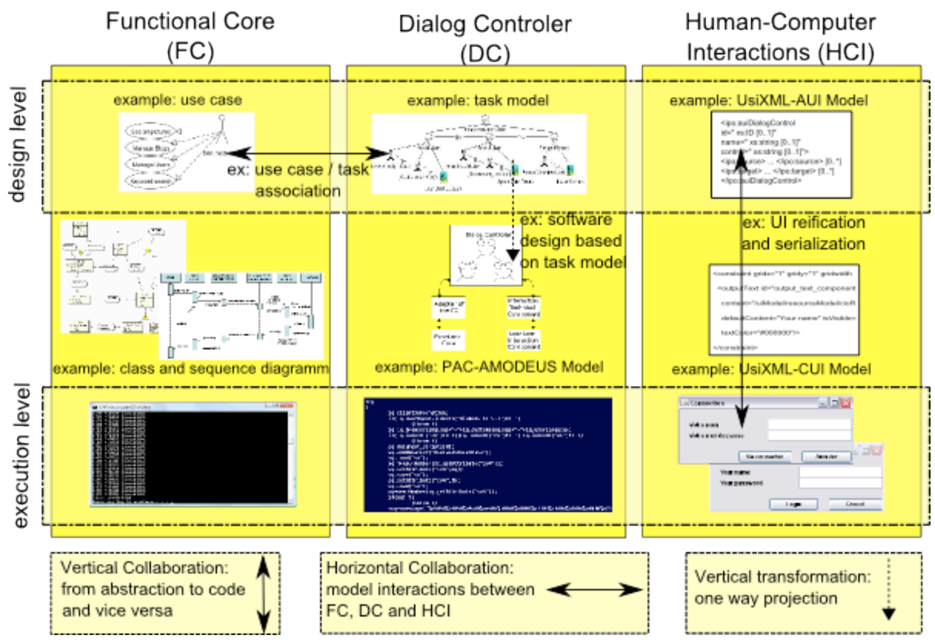

Functional Core (FC)
Dialog Controler (DC)
Human-Computer Interactions (HCI)
design level
execution level
example: use case
example: task model
example: UsiXML-AUI Model
ex: use case / task association
ex: software design based on task model
ex: UI reification and serialization
example: class and sequence diagramm
example: PAC-AMODEUS Model
example: UsiXML-CUI Model
Vertical Collaboration: from abstraction to code and vice versa
Horizontal Collaboration: model interactions between FC, DC and HCI
Vertical transformation: one way projection

Fig. 2. Biaxial and bidirectional transformations/mappings.

Miscellaneous statements about MDE in practice: automatic forward generators mostly produce "fast food" UIs in which models and transformations are lost. The transformations do not preserve collaborations at the code level, and horizontal collaborations between FC and UI are not yet well supported. In addition, consistency between models, transformations and code is not ensured at runtime along the models evolution. This calls for further research. Next section elicits three main requirements.

3 What we need for going further: three requirements to conclude

To our understanding, HCI is an interesting domain for both illustrating and inspiring research in MDE: on one hand, there is a long know-how in models and transformations in HCI. It provides a lot of knowledge to support experiments in MDE. On the other hand, keeping models at runtime raises new perspectives in both HCI and MDE. Whist the global picture has been demonstrated on simple case studies in HCI so far, we now aim at going beyond toy applications. This calls for mastering MDE for HCI, i.e., being able to reuse mature generic supports (methods and tools) from MDE so that to concentrate our efforts on the specific features of HCI (e.g., ergonomics). We identify three major requirements for going further: the support for multi-models and multi-actors, capitalization and evaluation of models.

#1. Support for multi-models and multi-actors

From now on, an interactive system is depicted as a graph of models that conveys both its design rationale and evolution over time. The models cover the internal state of the system (FC and UI) as well as its deployment on the interaction resources. We aim at formalizing the ergonomic properties so that to (1) characterize well-formed interactive systems, (2) be aware of the validity domain of interactive systems, (3) predict the effect of transformations, and (4) compute the conditions to satisfy and transformations to perform for ensuring a set of properties. Defining the actors in charge of such reasoning is interesting: who among designers and end-users might be in charge of observing and/or controlling graphs of models? Which views would be appropriate for whom and when? Fig. 3 and 4 provide two examples. In Fig. 3, the designer can delete interactors by placing a toolglass on either the graph of interactors or the UI itself. Consistency is ensured. In Fig. 4, the end-user controls the distribution of his/her web site among the two connected platforms. One step further, we can imagine putting metamodels under the human control as well. The global picture gives rise to the notion of Mega-UI [7].

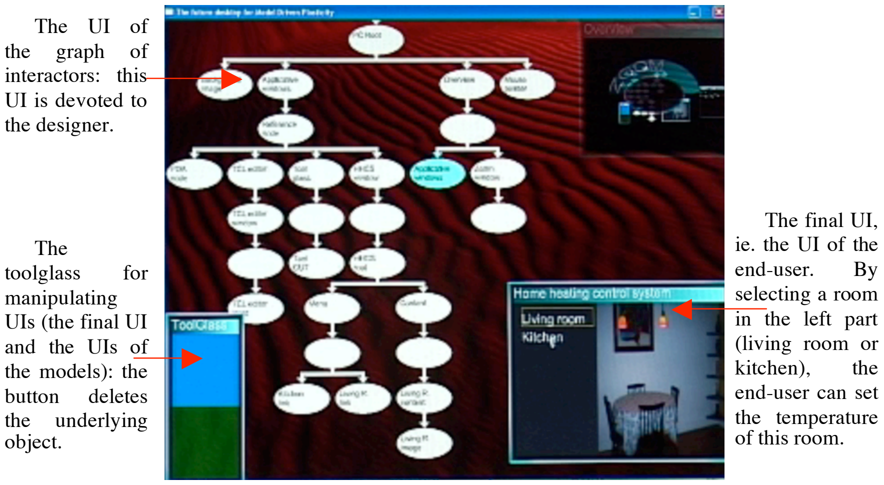
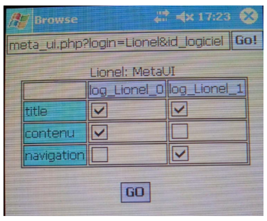

The UI of the graph of interactors: this UI is devoted to the designer.
The toolglass for manipulating UIs (the final UI and the UIs of the models): the button deletes the underlying object.
The final UI, ie. the UI of the end-user. By selecting a room in the left part (living room or kitchen), the end-user can set the temperature of this room.
Overview
ToolGlass
Home heating control system
Living room
Kitchen

Fig. 3. Additional UIs for the designer: the interactors model is made observable.



meta_ui.php?login=Lionel&id_logiciel

Lionel: MetaUI

	log_Lionel_0	log_Lionel_1
title	✓	✓
contenu	✓	
navigation		✓

Fig. 4. Additional UIs for the end-user: the check boxes represent the mappings between the UI structure (the rows: title, content, navigation) and the connected platforms (the columns).

#2. Support for capitalization

As modeling and metamodeling takes time, capitalization is crucial for going beyond small and basic interactive systems. We need a broad capitalization of models (e.g., post-WIMP interactors such as round windows, platforms such as multi-touch tables), metamodels as well as gateways between languages as one will never fit all.

#3. Support for evaluation

In HCI, evaluation is necessary. If evaluating a novel interaction technique is feasible, how can we measure the effect of MDE in HCI. More than evaluate the performance of the approach (cost and benefit), it is important to integrate evaluation in the design process so that to step by step check whether transformations fulfills ergonomic properties and platforms constraints. This goes far beyond evaluating the performance of the generated code. To that end, we need methods and tools for benchmarking our proposals.

References

1. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-design: a manifesto for end-user development. In: Communications of the ACM, Volume 47, Issue 9, pp 33-37, ACM Press, (2004)
2. Johnson, P., Wilson, S., Markopoulos, P., Pycock, J.: ADEPT-Advanced Design Environment for Prototyping with Task Models. In: Proceedings of InterCHI'93, p. 56, ACM Press, New York, Amsterdam, 24-29 April (1993)
3. Limbourg, Q.: Multi-path Development of User Interfaces, Ph.D. thesis, Université catholique de Louvain, Louvain-la-Neuve, Belgium, November (2004)
4. Myers, B., Hudson, S.E., Pausch, R.: Past, Present, and future of user interface software tools, ACM Transactions on Computer-Human Interaction (TOCHI), Volume 7, Issue 1, pp 3-28, March (2000)
5. Occello, A., Casile, O., Pinna-Déry, A.-M., Riveill, M.: Making Domain-Specific Models Collaborate. In 7th OOPSLA Workshop on Domain-Specific Modeling (DSM'07), pp 79-86, Montréal, Canada, 21-22 October (2007)
6. Paternò, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proceedings of Interact'97, pp. 362-369, (1997)
7. Sottet, J.S., Calvary, G., Favre, J.M., Coutaz, J.: Megamodeling and Metamodel-Driven Engineering for Plastic User Interfaces: Mega-UI. In: Human Centred Software Engineering: Software Engineering Architectures, Patterns and Models for Human Computer Interaction, Seffah, A., Vanderdonckt, J. and Desmarais, M. (Eds) (2008)