

At the Cross-Roads between Human-Computer Interaction and Model Driven Engineering

¹ Gaëlle Calvary, ² Anne-Marie Dery-Pinna, ³ Audrey Ocelllo, ⁴ Philippe Renevier-Gonin, ⁵ Michel Riveill

¹ Laboratoire d'Informatique de Grenoble, Campus de Grenoble – Bâtiment B – 385, avenue de la Bibliothèque – B.P. 53, F-38041 Grenoble Cedex 9, FRANCE

^{2,3,4,5} Université de Nice - Sophia Antipolis - CNRS, Bâtiment Polytech' Sophia – SI 930 route des Colles – B.P. 145, F-06903 Sophia Antipolis Cedex, France

gaelle.calvary@imag.fr, pinna@unice.fr, ocello@unice.fr, renevier@unice.fr, riveill@unice.fr

ABSTRACT

Human-Computer Interaction (HCI) engineering was oriented during its infancy towards the use of models in order to formulate its inherent knowledge. The formulation and exploitation of this know-how was then rapidly explored, through the use of automatic User Interface (UI) generation tools, in order to reduce its development costs. However, because of the disappointing quality of the resulting UI's this approach was quickly abandoned. Nowadays, these same models are being rediscovered under the umbrella of Model Driven Engineering (MDE), to tackle the requirements driven by dynamic environments inherent to ubiquitous computing or cloud computing application domains for example. The present paper recalls the key points of the interaction between HCI engineering and MDE, and reveals the compelling potential of combining the research efforts of the two communities.

Keywords: *Human-Computer Interaction, Model Driven Engineering, View, Actor, Phase.*

1. INTRODUCTION

Interactive systems [Wegner 97] are commonly described by a minimal breakdown distinguishing between the Functional Core (FC) and the User Interface (UI). The FC includes all means of processing data independently of any user representation. The UI makes choices in terms of presentation, as a function of the targeted usage context and ergonomic properties to be satisfied. The usage context includes: the user (his/her characteristics, competencies, preferences, etc.), the execution platform (its computational and communication capacity, and available I/O interaction resources), as well as the physical/social environment in which the user is placed (conditions of lighting and noise, public or private space, open or closed, etc.) [Calvary 03]. The ergonomic properties refer to the quality of interactive system usage (cognitive requirements of using interfaces). Ergonomic criteria [Bastien 93] allow the latter to be characterised: for example, the user's workload when accomplishing a task, or the UI's compliance with standards.

Research in Human-Computer Interaction (HCI) engineering has been strongly driven by improvements in software and hardware. These have been numerous in recent years, leading to a revolution in human-computer interactions: from monolithic applications reserved for experts in a fixed location, today's research is oriented towards widespread applications, accessible to all, at any time, in any place (public transport, office, home, street, etc.), with any sort of device (computer, basic or 'smart' phone, interactive display, domestic appliance, etc.). Such interactions are implemented via new types of UI which go beyond the interactional keyboard-mouse-screen context, favouring new interactive means such as the displacement of the devices themselves (location based interaction but also shaking or tilting the device). These UI's represent a major break from conventional UI's based on windows, menus, etc.

The present paper deals with the contributions of Model Driven Engineering (MDE) [Schmidt] to HCI engineering. It covers its history, indicating the key events of the use of MDE in HCI engineering, ranging from the formulation of knowledge through models (Section 2), to the formulation of know-how by means of model transformations (Section 3). Today, the requirements of ubiquitous applications are such that MDE has become crucial to HCI engineering. Conversely, HCI engineering represents a rich application domain for MDE, through the diversity of its models, the maturity of its transformations and the prospects for research into the presentation of models to users (UI's of models). Section 4 presents some avenues for research at the intersection between HCI engineering and MDE, i.e. promising MDE techniques for HCI engineering, and research topics in HCI engineering, which have a wealth of potential applications in MDE. Our paper concludes with a set of perspectives, which encourage further collaboration between the two communities, in line with the widely applauded French MDE initiative known as "Action IDM".

2. FORMULATION OF KNOWLEDGE: FAMILIES AND MODELS USED IN HCI ENGINEERING

Models are traditionally used in HCI engineering to formulate knowledge of requirements and/or designs. This section presents the models used in HCI engineering, and their evolution through the needs which have successively appeared with the mutation of computing: democratisation of computing, mobile computing and ubiquitous computing.

2.1 Modelling Interactive Applications and Their Users

The democratisation of computing has placed the user at the centre of HCI research. The challenges are to improve the quality of human-computer interactions, and to support the resulting developments using the high performance methods and tools needed to ensure this quality. User-centred approaches have evolved over the course of time, from cooperative [Greenbaum 91], to participative [Schuler 93], and currently contextual [Beyer 98] designs. Generally, the user-centred design processes introduce specific steps and activities, whereby the human is taken into account in the process. For example, Figure 1 takes the conventional V development cycle of UI's one step further, defining the ergonomic evaluation as the key activity to be carried out at each step of the process. Indeed, the regular evaluation of the ergonomics of an UI reduces the risk of rejection at the end of a cycle.

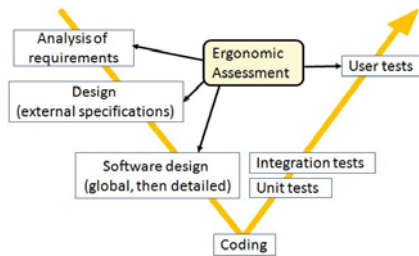


Fig 1: Example of a user-centred design process. Feedback to preceding steps is not indicated.

Historically, the main phases in which models are involved are those concerning analysis and design: analysis of the requirements, external specifications, and software design. These three phases and the models they use or produce are described in detail in the following sections.

2.1.1 Analysis of Requirements and its Models

A key aspect at this stage is to establish a user model. This model describes a typical user. It sets general parameters (age, gender, etc.), professional competence (novice/expert, etc.) and computing ability (cautious/inquisitive/skilful, etc.) of the supposed user. Studies in cognitive psychology allow some of these characteristics to be quantified, which makes it possible to predict the user's ease or rapidity in performing a task. For example, the degree of knowledge could be novice, intermediate or expert, according to the Rasmussen model [Rasmussen 83], knowing that the expert will have reflex reactions (automatisms), whereas a novice will need to be guided, trained or even reassured. The model of the user currently remains informal, and is saved in the form of free text.

The personality (persona) technique [Cooper 99, Blomkvist 02] is currently a well-recognised approach for the modelling of users. A personality is a key individual (for example, Susie, a young, sparkling woman – Figure 2) whose model describes character traits, behaviour, etc., with some characteristics being standard, and others less

expected [Perfetti 07]. Such a model is produced following observations and interviews with real users. The personalities are then placed into scenarios, which challenge them with objectives to be achieved.


First name	Susie	 SUSIE The "sparkling" woman
Age	30 years	
Nationality	French	
City	Nice	
Qualifications	Accounting and Management	
Profession	Chartered accountant	
Family situation	Shared lodgings, no children	
Level of computing and internet competence	Everyday knowledge of office tools and internet; electronic mail, information search	
...		

Fig 2: Example of a personality.

The scenarios [Rosson 02] support the complete development process: problem scenarios (problem description), are then broken down into activity scenarios (description of activities needed to deal with the problem), and illustrated by interaction scenarios (staging of the activity). The scenarios can be written in text form, drawn in the form of scenario pictures (Figure 3), or even recorded using other media forms such as a situational video [Renevier 04]. These scenarios allow the different players in the design process (computer scientists, ergonomists, users, etc.) to communicate. Their informal character stimulates creativity [Nielsen 93]. They are recorded in the test plan [Lim 94] and are used during the evaluation phase for the testing or commissioning of the interactive system. They are also used in an early phase to verify with the users the accuracy of the foreseen process.



Fig 3: An example of a scenario picture. Excerpt taken from the European project GLOSS.

In order to refine the process, the task and domain concept models are commonly used. The task model formulates the user's intentions (for example, book a seat at a show) as well as the procedure used to achieve this objective (for example, provide one's identity and then specify the desired seat). The procedure is a recursive break-down of the main objective into sub-objectives, with the latter being related to one another by logical and/or temporal operators. The task model is certainly more advanced and operational. Languages for task modelling include, for example, the MAD (Méthode

Analytique de Description de tâches) [Scapin 89] and CTT (Concurrent Task Tree) [Paterno 97].

Figure 4 is a graphical representation of a task model written in CTT: it explains how to book a seat at a show. It also shows that a task description is independent of the presentation options: these options depend on the external specifications. In addition, it shows that a task manipulates concepts in the relevant field: for example, notions related to seat, address, identity, etc. These concepts are described separately in a concept model. The UML class diagrams are widely used to formalise these models.

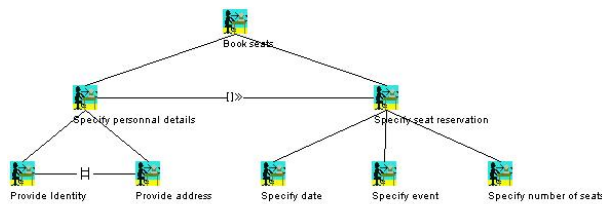


Fig 4: Graphical representation of a task model written in CTT.

2.1.2 External Specifications and their Models

Once the user-centred models have been established, the design of an UI is built around models which capture the UI's presentation as well as its interactions. The presentation models describe the UI in terms of structure and/or rendering choice. The interaction models describe the dynamics in the form of events/reactions. These descriptions can be formulated in natural language or expressed in languages and formalisms such as UAN [Hix 93].

In order to create these models, a conventional process involves starting with the task model or scenarios, and enriching it by describing the interactions in increasingly fine detail. Each refinement involves design choices (structuring of the UI, preference for one or another interactor, etc.). The UI development process has been implemented in different ways. However, a consensus has been reached around four levels of abstraction [Szekely 96]: user tasks and field-specific concepts, abstract UI, concrete UI and final UI.

The abstract UI model structures the UI into dialog zones (the graphically shown interaction zones). It organises the UI into abstract interactors (for example, the notion of choice) without defining whether these should be represented by drop-down menus, tick-boxes, etc. The concrete UI model makes the rendering choices. At this stage, all of the presentation choices have been made, but their means of implementation have not yet been defined. These choices are made at the so-called final UI stage which sets the development and execution environment. In addition to distinguishing between different objectives, the formulation of these models allows the chosen options to be validated by the various players involved: not only users, but also ergonomists, graphic designers and computer scientists.

A natural consequence of the formulation of knowledge has been the operationalisation of the models, in order to automate code generation. A number of tools (for example, ADEPT [Johnson 93] have been proposed. They comply with a model-based approach (MB-IDE for Model-Based Design Environments). The four levels of abstraction mentioned previously establish the architectural basis for the currently explored methods.

2.1.3 Software Design and its Models

Once the UI has been designed, the next step is to design the software architecture allowing the UI to be linked to the functional core. Numerous studies in HCI engineering have dealt with the software architecture of interactive systems [Coutaz 01]. As in the case of software development in general, the importance of the architectural model in HCI engineering can be explained by the iterative nature of the design. This calls for the use of modular code corresponding to a reference architectural style in order to simplify software upgrades [Coutaz 01].

Two types of model allow the interactive system to be structured. The first type of model is based on a structure that isolates the functional part of an interactive system from its presentation to the user, as in the case of MVC (Model View Controller) [Reenskaug 79] or PAC (Presentation Abstraction Control) [Coutaz 87, Avgeriou 05]. MVC distinguishes between three different aspects in software: the 'model' (the business part), the 'view' (business data representation) and the 'controller' (management of user interactions). The model communicates with the view in accordance with the "observer-observable" design pattern, with the view being able to query the model to obtain current values. The controller manages the user events by initiating UI updates and by transmitting any data changes to the model. The PAC model has a slightly different breakdown. The 'A' (Abstraction) facet corresponds to the model part of MVC. The 'P' (Presentation) facet corresponds to the view and controller part of MVC. The PAC model thus introduces its own additional facet 'C': control. The control has two purposes: to enable the exchange of communications (and translations) between abstraction and presentation, and also to ensure the UI's consistency, by means of a hierarchy (tree) of PAC units linked together by their control facets.

The second type of model corresponds to a functional breakdown of the interactive system. This is the ARCH [UIMS 92] model which refines the UI into four functions: a Functional Core (FC) adapter which provides for data translation between the FC and the UI; a Dialog Controller (DC) which drives the dynamics of the UI; a logical presentation which assumes the role of an adapter between the DC and the physical presentation; and the physical presentation corresponding to the implementation of the UI in the given language and toolbox.

Mobile IT introduces a new dimension to these notions: the development of multi-platform UI. This

dimension opens up new perspectives for the aforementioned models.

2.2 Platform Modelling

The democratisation and success of mobile media (mobile phones, personal assistants) has made the task of UI developers more difficult. The aim is to avoid having to redevelop an UI when the physical or software platform changes, and to ensure ergonomic consistence between versions. The European project CAMELEON [Calvary 03] (<http://giove.cnuce.cnr.it/cameleon.html>) was involved in the four previously established levels of abstraction (Section 2.1) in order, firstly to enable multi-target UI development, and secondly to provide tools for such development. The abstract and concrete UI's are then recognised for their importance in terms of pivotal models: they assume the role of a PIM (Platform Independent Model) in a MDA (Model Driven Architecture) context [OMG 01], whereas the final UI assumes the role of the PSM (Platform Specific Model).

The variety of platforms, and more generally of prevailing usage contexts motivated the search for multi-platform UI's (aka Plastic UI) in 2001 [Thevenin 01]. New models were introduced in order to describe the usage context (user platform, environment) and changes in usage context. These are evolution models (a set of Event-Condition-Action rules) and transition models, with the purpose of the latter being to accompany the user in any change (for example, by means of a deformation of the UI enabling visual continuity).

With the advent of ubiquitous computing, usage context and changes in usage context are no longer always predictable at design time. The HCI community has thus been faced with new challenges.

2.3 Modelling the Usage Context

Ubiquitous computing as it was imagined by [Weiser 91] promises citizens the permanent availability of services: the user is mobile [Lyytinen 02]; he/she can access his/her data and applications from anywhere, at any time, from any device. This desire for universality [Scholtz 99] calls for the modelling of knowledge and know-how, in such a way as to prepare the system for situations unforeseen at design time. The Meta approach [Fischer 04] is thus an obvious solution, thereby disrupting various dichotomies.

Between phases. The development and execution phases were until recently distinct: development decisions were the developer's prerogative, whereas the execution was driven by the user. With ubiquitous computing, since the usage contexts are no longer systematically predictable at design time, the system needs to be endowed with generating capacity allowing it to deal with usage contexts for which no tailor-made UI would have been designed at the outset. The unification of design and execution corresponds, in reality, to a shift to the Meta level, in order to design what is no longer an interactive system, but a generator tool. This transition in reasoning requires the flexible points (i.e. decision points in HCI engineering) of an UI to be identified, the corresponding

metamodels to be formulated, and calls for various transformations of the corresponding models [Sottet 08].

Between players. The unification of design and execution results in the abolition of the dichotomy between the designer and the user. Now, the user shapes his interactive space, just as the designer invents an UI. End user programming [Smith 77] and DSL's (Domain Specific Language) [van Deursen 00] are domains, which are today being rediscovered in HCI engineering. For example, in Jigsaw, by means of a dedicated graphical editor, the user builds simple programs by assembling jigsaw puzzle pieces such as "If someone rings the doorbell, take a picture and transfer it to the PDA" [Rodden 04]. This perspective raises an interesting question: that of the view associated with models and metamodels. How can (meta)models be presented to users who are not necessarily IT experts? How much control can they be given, and via which UI? The question remains open.

Between functions. The lack of consideration for FC (Functional Core) is starting to change. Indeed, in some applications it is important to recognise that the FC exists inherently, independently of the use to which it is put [Blay-Fornarino 07]. As a consequence, it is important to see beyond the division between UI and the FC. Of course, this modularity must be revisited in order for the changes in FC to be suitably taken into account, in a world in which services dynamically appear and disappear. Indeed, service and component-based software platforms simplify the dynamic assembly of components. Changes in the assemblies can have an impact on the UI by modifying, for example, the user tasks. This observation has led some studies to propose solutions which, at the level of the presentation and the interactions, automate the interpretation of adaptations of the FC, the latter being produced by means of a dynamic assembly of services. We cite the work of [Bihler 07] dealing with the creation of an UI based on a workflow of services, and also the work of [Pinna-Dery 08] who proposes a platform assisting with the coherent assembly of UI's as a function of the assembly of functional services. Such an approach favours the re-use of existing solutions. The work of [Mosser 08] transforms data workflows described at the Mashup level (autonomous part of web pages, which provides the UI and the functional part, and is generally handled by administrator pages on the web), into service orchestrations. The work of [Brel 10] reuses existing UI's for creating new applications by composition while preserving user requirements of individual original systems and keeping some of the links between the functional part and the UI part in the resulting system. At the heart of these approaches, are models and model transformations, permitting either the transition from a dataflow model to a workflow model, or the transition from an abstract UI model to a concrete UI model in which the UI's are designed as assemblies.

Between technologies. Until recently, UI's were technologically homogeneous. With the possibility of distributing a UI over several physical platforms (for example, remote control on a PDA, content on a PC), it has now become necessary to envisage hybrid UI's,

http://www.scientific-journals.org

combining different technologies: one part in html (“conventional” UI composed of text, buttons, etc.), another part in OpenGL (3D UI) or even in VoiceXML (for speech synthesis of information). Some of these parts can be generated, others can be reused. The generation can be based on conventional toolboxes, or can dynamically invoke advanced components [Demeure 08] and thus depart from the form type of UI usually generated up until now.

Figure 5 depicts a proposed classification for the models used in the engineering of interactive systems. It relies on an ARCH-like decomposition, which distinguishes the presentation part (P) from the dialog controller (DC) in the UI. The FC adaptor and the nuance between logical and physical presentations are not discussed here.

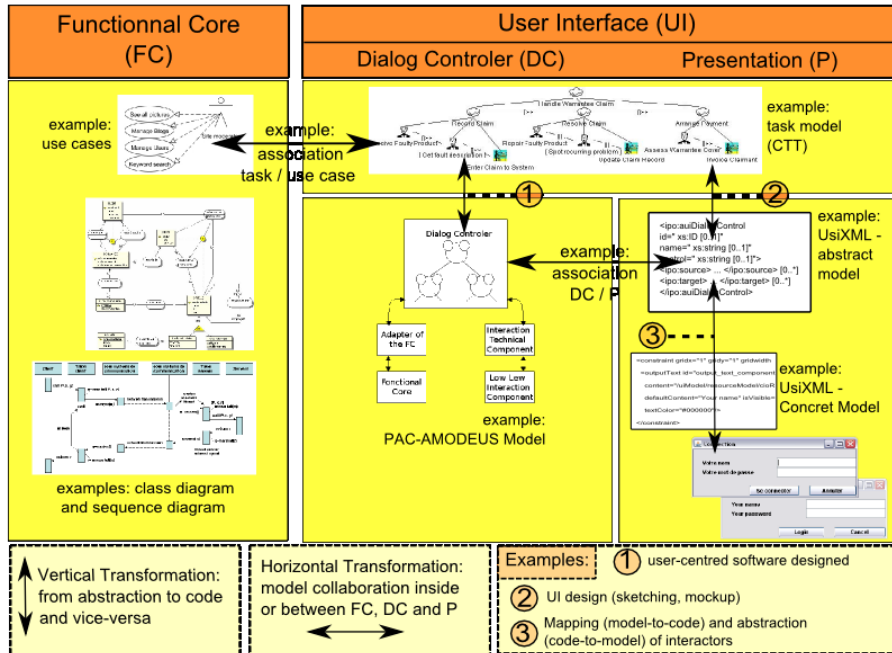


Fig 5: Model collaboration in HCI engineering.

For the FC part, one can cite as examples of conceptual models: the use cases, class diagrams or sequence diagrams, as proposed in UML (Figure 5). From the point of view of its implementation, one could apply SCA [Marino 09], Fractal [Bruneton 06] or OSGi [OSGi alliance 03]. For the DC part, the best-known models are Pétri networks [Palanque 96] or a hierarchy of PAC agents as recommended in PAC-AMODEUS [Nigay 97]. In the latter case, there are heuristic rules for deriving the DC from the external specifications. For the P part, the more conventional models are often based on abstract or concrete UI as proposed in UsiXML [Limbourg 04b], or graphical multi-platform toolboxes.

Despite the FC/DC/P breakdown, the models are not mutually exclusive. Indeed, they have an influence on one another: their definition requires mutual feedback, which is not always simple to implement, since the players are not necessarily the same. For example, the task model can be an extension of use cases manipulated in the FC (Figure 5). The use cases model the system functionalities only, without taking the UI into account, whereas the task model is based on a similar approach,

but is potentially capable of integrating functionalities as “system tasks”. When these (use case and task) models are kept mutually consistent, this can be referred to as ‘horizontal collaboration’.

Starting from the task model, a dialog model and an abstract UI model can be produced (Figure 5). Conversely, starting from an abstract or a concrete UI, the task model can be found. For example, Figure 4 is a possible task model for the UI shown in Figure 6. If the task and dialog models, the task and abstract UI models, or even the abstract and concrete UI models are kept mutually consistent, this can be referred to as ‘vertical collaboration’.

http://www.scientific-journals.org

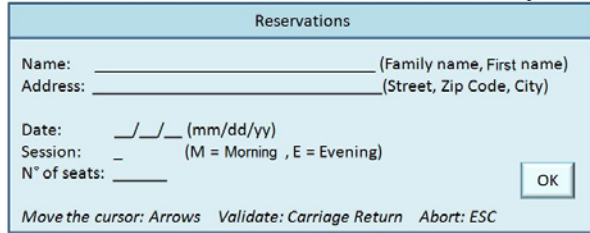


Fig 6: UI (excerpt from [Nogier 05]) for which a possible task model is that shown in Figure 4.

By preserving collaborative links during execution, the design choices can be dynamically reviewed, whenever the usage context changes. However, such a revision requires the formulation of a certain volume of know-how, i.e. transformations.

2.4 Summary

The present section has described many models in HCI engineering: user model, task model, dialog model, platform model, environment model, evolutionary model, transitional model, etc. Not all of these models are treated in the same manner. Some are treated numerically (the task model in particular), others are not (scenario models in particular). Some of these are sufficiently mature (for example, the task model) to be qualified as metamodels, whereas others are not.

Currently, some languages are thus sufficiently advanced and equipped with tools. For example, tasks may be described using the CTT language and its CTTe environment [Mori 02]. This is also the case for grammatical structures such as UsiXML [Limbourg 04b], which provide good coverage and are being standardised. Nevertheless, some models such as concrete UI still justify further improvements. This is also the case of the “mapping model”, a key model for the dispatching of vertical and horizontal collaborative links, since it links models with different levels of abstraction (multiple development paths) together, and allows links to be maintained between the different UI representations. The mapping model has strong similarities with the mapping implemented between elements of a source model and a target model, within a model transformation rule, according to a traditional MDE process.

Figure 7 shows the mapping between an interaction model and a concrete UI model. The tasks are grouped into interaction or “Layout” areas (Interaction Space), allowing them to be mutually structured. Each of these areas is mapped (arrows) with interactors (windows, tabs, etc.): the “Actions” are mapped to buttons, the “Input Elements” to text fields, ...

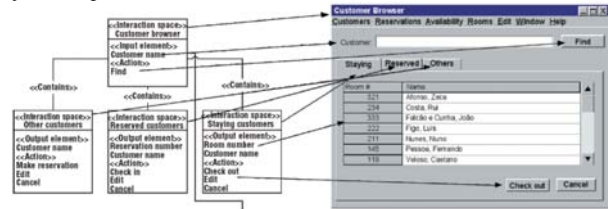


Fig 7: Example of a mapping model.

3. FORMULATION OF KNOW-HOW: TRANSFORMATION OF MODELS IN HCI ENGINEERING

HCI engineering forms a rich applicative framework for MDE, as a result of the variety of operations which can be made on the models: composition by fusion, difference or joining of models in order to re-use and compose different modelling types [Pinna-Dery 03], [Lepreux 06]. This section covers the history of the transformations and various operations carried out on the models in HCI engineering, from the standpoint of the requirements identified in section 2.

3.1 Adapting UI's to the Users

Numerous studies dealing with model transformations make use of the task model and a concrete UI to automate code generation for a given user and platform. The four levels of abstraction can then be viewed as different levels of refinement of the same UI, despite the absence of a common metamodel.

In most studies, the task model is the starting point of the transformations: the final UI is generated from a task model, without it being possible, however, to monitor the transformations or intermediate models. For example, the UI shown in Figure 6 could be generated automatically from the task model in Figure 4, by means of model transformations associating (1) one text field per leaf task, (2) an indicative text to the left of each text field, accompanied if necessary by a user guide to the right, and (3) a space to separate the information “packets” of each intermediate task (here, identification of the user specifying the seats).

The disappointing quality of generated UI's penalised this approach, and consequently the models [Myers 00]. The boom in PDA's has led to renewed interest in models for HCI engineering: the wide range of platforms indeed calls for economic solutions, to ensure ergonomic coherence between platforms (for example, a PC version and a phone version). On the assumption that business evolves less rapidly than technology, the MDA approach is dedicated to the management of the specific aspect of software dependence on an execution platform. The HCI community has thus naturally turned to MDA, which provides a response to these new preoccupations related to the generation of UI's on multiple platforms, in different languages.

3.2 Adapting UI's to Platforms

In order to deal with the multitude of existing platforms, the traditionally adopted approach of reverse engineering is giving way to other conceptions. In particular, different entry points can be selected (for example, at the level of the concrete UI, to integrate the use of layouts) and different monitored design paths (for example, reverse engineering to recover the task model, starting from an UI layout). The reference framework CAMELEON [Calvary 03] was then proposed as a methodological support, to assist with the design steps and paths. This reference framework explicitly shows that the transformations may be either manual or semi-automatic: the justification for "all automatic" is no longer applicable.

The designer can specify the model transformations to monitor the quality of the UI's produced. The specification of transformations implies integrating ergonomic criteria into the transformation process. For example, if "error protection" [Bastien 93] is a criterion considered to be important for a given UI, the transformation of a choice should privilege radio buttons, drop-down menus, or tick-boxes, rather than free text fields such as those used in Figure 6 for the selection of a morning or an evening session. Very often, a poor (i.e. non-ergonomic) transformation implies, as a result, the addition of interactors in order to provide the user with compensatory guidance (guidance is another criterion in ergonomics [Bastien 93]). In Figure 6, the text "M = Morning, E = Evening" is an example of this. Unfortunately, the text is very often ambiguous, thus leading to other ergonomic errors. For example, in Figure 6 the text "Family name, First name" is ambiguous: some users use a comma, others do not. [Sottet 08] shows how each element of an UI can be justified by means of transformations. If an element is not justified, then the ergonomic criterion of conciseness [Bastien 93] is transgressed.

When the variability and the unpredictability of the usage context are then included in the debate, the HCI community leaves MDA behind, in preference for MDE as it is known and used today.

3.3 Adapting UI's to the Usage Context

With the disappearance of the borders between design and execution in ubiquitous computing, the models become key elements, available during execution [Bencomo 06], which are indispensable for the appraisal of, and reaction to the applications' execution [Muller 07]. The different UI models (tasks, concepts, abstract UI, concrete UI, etc.) thus become observation points of the interactive system, enabling the decision to be made to suitably adapt UI's whenever the usage context changes.

[Demeure 11] proposed to include a model flow diagram (Figure 8, aka "Graph of models") in the execution, to support dynamic adaptation. The model flow diagram explicitly represents the transformations and collaborative links between models, relating to:

- Interactive systems: the levels of abstraction of the UI, the FC and its connection to the UI (relationship with the concepts and the tasks);
- Usage context according to its three constituents < User, Platform, Environment > and their inter-relationships: user and platform position in the environment (respectively Rel U-Env and rel Ptf-Env) as well as the use of the platform's I/O devices (Rel U Input/output) by the user;
- Deployment of the interactive system on the platform (Rel NF-Ptf and Rel Ptf- CUI, the latter being refined to Rel Inputs/Outputs-CUI).

The strength of the model flow diagram lies in the fact that it preserves the models (but does not consume them) and makes the transformations between models explicit. The relationships between models can thus be seen as causal links, since they exist between a system and the models at runtime to which it is related. The loss of information was known as the "mapping problem" [Clerckx 04] [Griffiths 01] [Limbourg 04a]. However, the preservation of relationships between models at runtime reduces the impact of this loss.

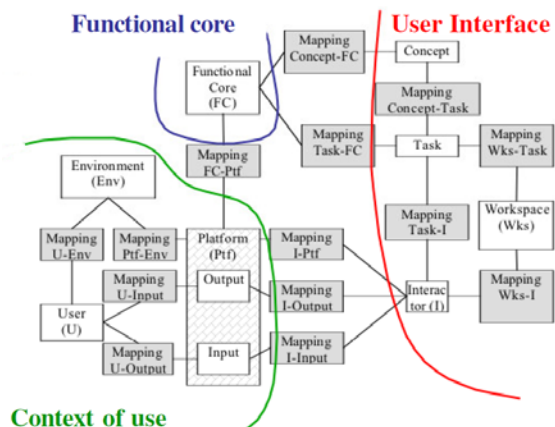


Fig 8: Model flow diagram. The (white and grey) boxes represent models. The grey boxes indicate relationships (Rel) between the models.

The Ecosystem allows the impact of a modification on a model to be analysed. For example, the modification of a task will imply changes to all of the interactors to which it is related. Another example is given by the movement of the users or rendering devices in the environment, which could cause a given task to become inaccessible (for example when searching for information) if the corresponding interactors are not enlarged or replaced by more suitable ones.

3.4 Summary

The absence of some metamodels means that some of the models are exploited manually by those who develop or use the interactive systems, whereas other models can be exploited automatically by means of MDE

tools. Here, we are interested in the types of transformation which have been experimented with and used in HCI engineering:

- Model-to-model, such as the transformation of a task model into a presentation model. In the CTTE environment [Mori 02], the task model is represented by a tree whose nodes are the user's objectives, with the links between the nodes being dependence relationships between these objectives. It is then possible to transform the task model firstly into a dialog model representing the series of tasks such that the user can accomplish them, and then into an intermediate presentation model. This link between the dialog controller (DC) and the abstract presentation model is a case of horizontal transformation (cf. Figure 5).
- Model-to-code for the automatic generation of the final UI, starting from the abstract or concrete UI. This is the case for MXML transformations (concrete UI) into a Flash executable in the Flex environment (<http://www.adobe.com/fr/products/flex/>). The transformations within UsiXML are further examples of this. Both cases correspond to vertical transformations (cf. Figure 5).
- Code-to-model for the serialisation of UI's. Both in the case of UsiXML and ALIAS [Occello 10], the aim here is for the framework computation not to depend on the final UI description, and for it to remain independent of the execution platform. The model thus obtained can then be used when changing platform, or for UI compositions.

Ultimately, the operationalisation of all models, metamodels and transformations will open up possibilities for true fast prototyping, providing support for creativity. This creativity is essential in design, as well as in evaluation. Indeed, [Tohidi 06] has demonstrated the strength of a comparative assessment, with respect to an absolute one: the ability to generate alternative UI's at a low cost offers attractive perspectives.

4. WHAT IS THE FUTURE OF HCI ENGINEERING AND MDE?

This section proposes to take a step back from the usage of models in HCI engineering. It shows how research workers in HCI engineering and MDE can try to solve, together, problems relevant to their respective domains.

4.1 Promising Studies in MDE for HCI Engineering

Some needs in HCI engineering (sections 2 and 3) have found a partial response in the progress made in the field of model engineering. The requirements in terms of HCI engineering and "model-oriented" solutions have evolved more or less consistently and separately, and

provide evidence of fruitful convergence between these two domains. Other forms of collaboration could be imagined, in view of the extensive range of MDE applicability today. Indeed, this is the result of the evolution of a set of approaches centred on the automation of the software development process, which is likely to provide solutions for other known requirements in HCI engineering, such as product lines (Software Product Lines or SPL) [Dikel 97], Model Integrated Computing (MIC) [Sztipanovits 97], Domain Specific Modelling (DSM) [Pohjonen 02] and Aspect Oriented Modelling (AOM) [France 03].

Assuming software can be grouped into product families, sharing a set of functionalities which satisfy well defined requirements, the SPL approach emphasises modelling and the processing of the variability between different software packages within the same family [Perez 06]. This approach would then allow the variability of an UI to be managed as a function of the usage context (a typical example: the different versions of Google adapted to desktop computers and mobile phones).

If the definition of a software product can be broken down into a set of requirements (task model in HCI engineering), an architecture (MVC model, or PAC in HCI engineering), and an environment (platform model), and these three aspects are required to evolve together, then the MIC is based on a multi-model concept and manages the composition and evolution of these models. This approach is all the more relevant since the different models used in HCI engineering today evolve rapidly, and need to maintain coherent links (cf. model flow diagram shown in section 3.3).

With the advent of MDE, the focus on models has shifted, and code-related concepts such as the aspects or languages specific to this domain have given rise to model approaches based on these same concepts: DSM and AOM. The DSM approach is of interest when proposing suitable modelling tools for each type of model (low and high fidelity mock-ups, interaction and dialog models, task models, software architecture model) and its associated player (designer, ergonomist, software architect, developer, or even the end user). This approach thus ensures the continuity of refinement and traceability between these models through the use of a modelling tool chain. The AOM approach can be used in work related to UI composition, in order to facilitate inter-operation of the different models involved.

4.2 Promising Studies in HCI Engineering for MDE

Exploration of the use of MDE in HCI engineering has revealed promising opportunities for the application of key MDE concepts, i.e. models, metamodels and transformations. Here, HCI engineering has a clear advantage with respect to other applicative domains: the solid know-how developed in models and transformations. Now that this tandem has been discovered, and a common basis has been established, the next step is to refine MDE usage in greater detail. Here, we have identified various leads confirming that this

discovery is still in its early days, and that HCI engineering can offer a true potential for the development of MDE.

Relevance and quality of (meta)models. In HCI engineering, the identification of metamodels has been based on the past, re-adopting traditional task levels, and abstract, concrete and final UI. However, have the correct models been defined, in other words are the appropriate variables used in this reflection? Could a method be defined for the identification of these variables?

With models, it is important to describe their quality. This has been shown to be crucial for UI transformations, but can clearly be generalised to other models. In HCI engineering, transformations are operations in which ergonomics become relevant [Sotest 08a]. Indeed, if the task model includes part of the ergonomics (the UI is useful), there is another component of ergonomics which it does not take into account: its usability. The latter can be expressed by ergonomic criteria, such as those defined in [Bastien 93]. The ergonomic properties of the transformation need to be formulated during its adaptation, in order for the best transformation, i.e. that which is likely to best satisfy the user's expectations (for example, a low work-load), to be selected.

Properties of the model flow diagram. One of the hypotheses proposed today is that the model flow diagrams are complete and correct (as far as consistency is concerned): an UI is described from all points of view (tasks, structure, interactors, program) and these aspects are mutually consistent. In reality, these two properties might not be achieved:

- Incompleteness. In the case of an open approach, the dynamic adaptation of UI can be achieved by recruiting components. It can be expected that these components will not all be described in accordance with all of these aspects. The model flow diagram can thus be incomplete. Two approaches are possible: either complement the descriptions using automatic model recovery (model recovery through reverse engineering) perhaps with an associated confidence level with respect to the inferred models; or reason on the basis of incomplete information by installing "stoppers". If these approaches seem reasonable for the models used today, this is less true for more informal models which have not yet been considered, since they are far from being operational: this is the case for the user model. Nevertheless, this model is fundamental to the construction of an UI. How can it be integrated?
- Incorrectness. On the model flow diagram (Figure 8) expressing the "why" of design (design rationale), it is important that the model flow diagram be consistent, since there would otherwise be a risk of incorrectly steering the transformations. Inconsistencies could be

introduced by the designers, or be produced by incorrect descriptions. Whatever their origins, it is important for such inconsistencies to be detected and corrected.

UI of the model flow diagram. Since models, metamodels and transformations can now be placed under the control of the user (today, the designer, in time, perhaps, the user), the problem of the fractal nature of the description is raised: the UI's of these (meta)models can themselves be modelled, leading to new model flow diagrams. It is thus important, from the methodological point of view, to correctly circumscribe the studied object and to understand in which cases it is relevant to produce an UI for these (meta)models. And what sort of UI's? For example, thanks to MDE, UI's are potentially auto-explanatory, of the type "Why and Why not ?" [Myers 06]: how and when should such explanations be given? Presentation is important. It is known, for example, during the design phase, that low fidelity mock-ups provide a more active support for the exploration process [Hong 01]: the inaccuracy of the contours indeed allows the users to be more freely critical of the system. It can thus be productive to artificially deteriorate the quality of the general contour, to give the impression of an early design stage, and an approximate mock-up (Figure 9). The model flow diagram's representation could thus be adapted according to the life-cycle instant under consideration.

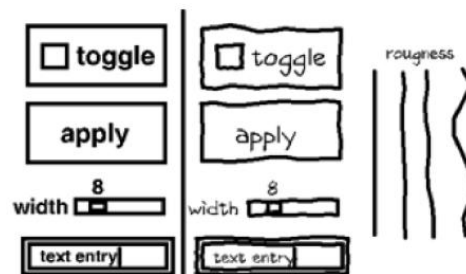


Fig 9: Fidelity levels. Excerpt from [Hong 01].

5. CONCLUSION

Through the multitude of models and transformations used in HCI engineering, the present paper has shown that HCI engineering constitutes a fertile terrain for experimentation in MDE. It has also shown that MDE has been able to provide answers to the needs of the HCI community. Some of the methods, concepts and techniques used in MDE have been, and continue to be, successfully used in HCI research projects: MDE provides the theoretical and technical foundations, bringing about rigor and knowledge capitalization through UI models, as well as know-how through the transformation of UI models.

However, as has been emphasised earlier in this paper, the relevance of the models or their representations used in HCI engineering should perhaps be queried, in particular in the context of end user programming [Smith 77], where these models or their representations should ultimately be manipulated by the end user. Similarly,

since it appears desirable to reinforce the links between the functional cores (FC) and their UI, for reasons of modularity and recomposition, the models used in HCI engineering, which have separated FC from UI, should be revisited.

ACKNOWLEDGEMENTS

We wish to thank the ANR CONTINUUM 2009_2012 project for supporting this study. We also wish to thank the members of the CESAME workshop from the GDR 13 "Design and evaluation of interactive systems capable of adapting to their usage context in an evolving world" for highly fruitful exchanges on the topic of the relationships between HCI engineering and MDE.

REFERENCES

- [1] [Avgeriou05] Avgeriou, P., Zdun, U. "Architectural patterns revisited - a pattern language". Proceedings of 10th European Conference on Pattern Languages of Programs, Isee, Germany, pp. 1-39, 2005.
- [2] [Bastien93] Bastien, J.M.C., Scapin D. Ergonomic Criteria for the Evaluation of Human-Computer Interfaces, Rapport technique INRIA, N°156, Juin 1993.
- [3] [Beyer98] Beyer, H., Holzblatt, K. Contextual Design, Kaufmann, 1998.
- [4] [Blay-Fornarino07] Blay-Fornarino, M., Hourdin, V., Joffroy, C., Laviotte, S., Mosser, S., Pinna-Déry, A.-M., Renevier, P., Riveill, M., Tigli, J.-Y. "Architecture pour l'adaptation de Systèmes d'Information Interactifs Orientés Services" in Revues des Sciences et Technologies de l'Information (RSTI), pp. 93-118, Lavoisier, 2007.
- [5] [Bencomo06] Bencomo, N., Blair, G., France, R., "Summary of the Workshop Models@run.time at Models 2006" in Lecture Notes in Computer Science, Satellite Events at the Models 2006 Conference, Springer-Verlag, pp. 226-230, 2006.
- [6] [Bihler07] Bihler P., Kniesel G. "Seamless Cross-Application Workflow Support by User Interface Fusion" In proceedings of Workshop on Ubiquitous User Interface Design, Aarhus, Denmark, pp. 5-8, 2007.
- [7] [Blomkvist02] Blomkvist, Stefan. "The User as a Personality. Using Personas as a Tool for Design. Topic 4 The blurred user" in Workshop Theoretical perspectives in Human-Computer Interaction (HMI656) at IPLab, KTH, 2002.
- [8] [Brel10] Brel C., Renevier P., Occello A., Pinna-Déry A.-M., Faron-Zucker C., Riveill M. "Application Composition Driven By UI Composition" in Proceedings of the Human Computer Software Engineering 2010 (HCSE 2010), IFIP International Federation for Information Processing, pp. 198-205, LNCS, 2010.
- [9] [Bruneton06] Bruneton, E., Coupaye, T., Leclercq, M., Quéma, V., Stefani, J.-B. "The Fractal component model and its support in Java: Experiences with auto-adaptive and reconfigurable systems", Software—Practice & Experience , 36(11-12), pp. 1257-1284, 2006.
- [10] [Calvary03] Calvary, G., Coutaz J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonck, J. "A unifying reference framework for multi-target user interfaces", Interacting With Computers, Vol. 15/3, pp 289-308, 2003.
- [11] [Clerckx04] Clerckx, T., Luyten, K., Coninx, K. "The Mapping Problem Back and Forth: Customizing Dynamic Models while Preserving Consistency" in Proc. of the 3rd Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2004 (Prague, November 15-16, 2004). ACM Press, New York, (2004), pp. 33-42.
- [12] [Cooper99] Cooper, A. The inmates are running the asylum. Macmillan, 1999.
- [13] [Coutaz87] Coutaz, J. "PAC, an Object Oriented Model for Dialog Design" in Proceedings Interact'87, North Holland, pp. 431-436, 1987.
- [14] [Coutaz01] Coutaz, J. Architectural Design for User Interfaces; The Encyclopedia of Software Engineering, J. Marciniak Ed., Wiley & Sons Publ., second edition, 2001.
- [15] [Demeure08] Demeure, A., Calvary, G., Coninx, K. "COMET(s), A Software Architecture Style and an Interactors Toolkit for Plastic User Interfaces" in Design, Specification, and Verification, 15th International Workshop, DSV-IS 2008, T.C.N. Graham & P. Palanque (Eds), Lecture Notes in Computer Science 5136, Springer Berlin / Heidelberg, Kingston, Canada, pp. 225-237, 2008.
- [16] [Demeure11] Demeure, A., Masson, D., Calvary, G. "Graphs of models for exploring design spaces in the engineering of Human Computer Interaction" in Proceeding of the 2nd SEMAIS workshop of the IUI'11 conference, Springer HCI, 2011.
- [17] [Dikel97] Dikel, D., Kane, D., Ornburn, S., Loftus, W., Wilson, J., "Applying Software Product-Line Architecture", IEEE Computer, pp. 49-55, 1997.
- [18] [Fischer04] Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N. "Meta-design: a

<http://www.scientific-journals.org>

- manifesto for end-user development", Communications of the ACM, Volume 47, Issue 9 (September 2004), End-user development: tools that empower users to create their own software solutions, Special Issue: End-user development, ACM Press, pp. 33-37, 2004.
- [19] [France03] France, R., Georg, G., Ray, I., "Supporting Multi-Dimensional Separation of Design Concerns" in AOSD Workshop on AOM: Aspect-Oriented Modeling with UML, 2003.
- [20] [Greenbaum91] Greenbaum, J., Kyng, M. Design At Work - Cooperative design of Computer Systems, Lawrence Erlbaum, 1991.
- [21] [Griffiths01] Griffiths, T., Barclay, P.J., Paton, N.W., McKirdy, J., Kennedy, J.B., Gray, P.D., Cooper, R., Goble, C.A., da Silva, P. "Teallach: a Model-Based User Interface Development Environment for Object Databases". Interacting with Computers, 14, 1, pp. 31-68, 2001.
- [22] [Hix93] Hix D., Hartson R., Developing user interfaces, ensuring usability through product & Process, Wiley, 1993.
- [23] [Hong01] Hong, J.I., Li, F.C., Lin, J., and Landay, J.A. "End-User Perceptions of Formal and Informal Representations of Web Sites" in Extended Abstracts of Proc. of ACM Conf. on Human Factors in Computing Systems CHI'2001 (Seattle, March 31-April 5, 2001). ACM Press, New York, pp. 385-386, 2001.
- [24] [Johnson93] Johnson, P., Wilson, S., Markopoulos, P., Pycok, J. "ADEPT - Advanced Design Environment for Prototyping with Task Models", Proceedings of InterCHI'93, Amsterdam, The Netherlands, pp. 56-57, 1993.
- [25] [Lepreux06] Lepreux, S., Vanderdonckt, J. "Towards Supporting User Interface Design by Composition Rules", Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces (CADUI'06), Chapter 19, Springer-Verlag, Berlin, pp. 231-244, 2006.
- [26] [Lim94] Lim, K.Y., Long, J. "The MUSE Method for Usability Engineering", Cambridge University Press, 330 pages, 1994.
- [27] [Limboung04a] Limbourg, Q., Vanderdonckt, J. "Addressing the Mapping Problem in User Interface Design with UsiXML" in Proc. of the 3rd Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA'2004, ACM Press, New York, pp. 155-163, 2004.
- [28] [Limboung04b] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Florins, M., Trevisan, D., "UsiXML: A User Interface Description Language for Context-Sensitive User Interfaces" in Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages", Luyten, K., M. Abrams, Limbourg, Q., Vanderdonckt, J. (Eds.), Gallipoli, pp. 55-62, 2004.
- [29] [Lyytinen02] Lyytinen, K., Yoo, Y. "Issues and challenges in ubiquitous computing". Communications of the ACM, Volume 45, Issue 12, pp. 62-65, 2002.
- [30] [Marino09] Marino, J., Rowley, M. Understanding SCA (Service Component Architecture), Part of the Independent Technology Guides series, Addison-Wesley Professional, 2009.
- [31] [Mori02] Mori, G., Paternò, F., Santoro C. "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design", IEEE Transactions on Software Engineering, pp. 797-813, 2002.
- [32] [Mosser08] Mosser S., Chauvel F., Blay-Fornarino M., Riveill M. "Web Service Composition: Mashups Driven Orchestration Definition" (long paper) in Proceedings of the International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC'08), IEEE Computer Society, Vienna, Austria, 2008.
- [33] [Muller07] Muller, P.-A., Barais, O. "Control-theory and models at runtime", Proceedings of the Models Workshop on Models@Runtime, Nashville, USA, 2007.
- [34] [Myers00] 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 (March 2000), Special issue on human-computer interaction in the new millennium, Part 1, pp. 3-28, 2000.
- [35] [Myers06] Myers, B., Weitzman, D.A., Ko, A.J., Chau, D.H. "Answering why and why not questions in user interfaces", Proceedings of the SIGCHI conference on Human Factors in computing systems CHI'06, Montréal, Québec, Canada, pp. 397-406, 2006.
- [36] [Nielsen93] Nielsen, J. Usability Engineering, Academic Press Professional, 362 pages, 1993.
- [37] [Nigay97] Nigay L., Coutaz J. "Software architecture modelling: Bridging Two Worlds using Ergonomics and Software Properties". Book Chapter, Formal Methods in Human-Computer Interaction, Palanque P., Paterno F. Eds., Springer-

<http://www.scientific-journals.org>

- Verlag: London Publ., ISBN:3-540-76158-6, pp. 49-73, 1997.
- [38] [Nogier05]Nogier, J.F. Ergonomie du logiciel et design Web : le manuel des interfaces utilisateur, 3^{ème} édition, Dunod, 272 pages, 2005.
- [39] [Ocellolo10] Ocellolo A., Joffroy C., Pinna-Déry A-M, Renevier-Gonin P., Riveill M. "Experiments in Model Driven Composition of User Interfaces" in Proceedings of the 10th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'10), vol. 6115, Eliassen F., Kapitza R. Eds., Springer-Verlag, pp. 98-111, 2010.
- [40] [OMG01] OMG, Model Driven Architecture, OMG Document ormsc/2001-07-01, 2001.
- [41] [OSGi alliance 03] OSGi alliance, OSGi Service Platform, Release 3, IOS Press, 2003.
- [42] [Palanque96] Palanque, P., Bastide, R. "Time modelling in Petri nets for the design of interactive systems". Revue SIGCHI bulletin, ACM Vol. 28, n°2, pp. 43-47, 1996.
- [43] [Paterno97] Paterno, F., Mancini, C., Meniconi, S. "ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models", Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction Pages, Interact'97, pp. 362-369, Sydney, ISBN:0-412-80950-8, Chapman & Hall, 1997.
- [44] [Pérez06] Pérez, J., Laguna, M. A., Crespo, Y., González-Baixaui, B., "Requirements Variability Support through MDD and Graph Transformations", International Workshop on Graph and Model Transformation (GraMoT05), Tallinn, Estonia. ISSN: 1571-0661, Volume 152, pp. 161-173, 2006.
- [45] [Perfetti07] Perfetti, Christine. Goal-Directed Design: An Interview with Kim Goodwin. User Interface Engineering, http://www.uie.com/articles/goal_directed_design/, 2007.
- [46] [Pinna-Déry03] Pinna-Déry, A.-M., Fierstone, J. "Component model and programming: a first step to manage Human Computer Interaction Adaptation" in Proceedings of the 5th International Symposium on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI), L. Chittaro (Ed.), vol. LNCS 2795, pp. 456-460, Springer Verlag, Udine, Italy, 2003.
- [47] [Pinna-Déry08] Pinna-Déry A.-M., Joffroy C., Renevier P., Riveill M., Vergoni C. "ALIAS: A Set of Abstract Languages for User Interface Assembly" in Proceedings of the 9th IASTED International Conference Software Engineering and Applications (SEA'08), IASTED, pp. 77-82, ACTA Press, Orlando, Florida, USA, 2008.
- [48] [Pohjonen02] Pohjonen, R., Kelly, S., Domain-Specific Modeling, Dr. Dobb's Journal, 2002.
- [49] [Rasmussen83] Rasmussen, J., "Skills, rules, knowledge; signals, signs, and symbols, and other distinctions in human performance models". IEEE Transactions on Systems, Man and Cybernetics, 13, pp. 257-266, 1983.
- [50] [Reenskaug79] Reenskaug, T. M. H., MVC XEROX PARC 1978-1979, <http://heim.ifi.uio.no/trygver/themes/mvc/mvc-index.html>, 1979.
- [51] [Renevier04] Renevier, P., Nigay, L., Bouchet, J., Pasqualetti, L. "Generic interaction techniques for mobile collaborative mixed systems" in Proceedings of CADUT'04, pp. 307-320, 2004.
- [52] [Rodden04] Rodden, T., Crabtree, A., Hemmings, T., Koleva, B., Humble, J., Akesson, K.P., Hansson, P., "Configuring the Ubiquitous Home" in Proc of the 2004 ACM Symposium on Designing Interactive Systems, Cambridge, Massachusetts: ACM Press, 2004.
- [53] [Rosson02] Rosson, M.B. and Carroll, J.M. Usability Engineering: Scenario-based Development of Human-Computer Interaction. London Academic Press, 2002.
- [54] [Scapin89]Scapin, D.L., Pierret-Golbreich, C. "Une méthode analytique de description des tâches". Colloque sur l'ingénierie des Interfaces Homme-Machine, Sophia Antipolis, pp. 131-148, 1989.
- [55] [Schmidt06] Schmidt, D. C. "Model-Driven Engineering", IEEE Computer, 39(2), pp. 25-31, 2006.
- [56] [Scholtz99] Scholtz J., Muller M., Novick D., Olsen D.R., Schneiderman B., Wharton C. "A Research Agenda for Highly Effective Human-Computer Interaction: Useful, Usable, and Universal", SIGCHI bulletin, ACM/SIGCHI, Volume 31, Number 4, pp. 13-16, 1999.
- [57] [Schuler93] Schuler, D., Namioka, A. Participatory Design: Principles and Practices, Lawrence Erlbaum, 1993.
- [58] [Smith77] Smith, D. C., Pygmalion: A Computer Program to Model and Stimulate Creative Thought. Basel, Stuttgart, Birkhauser Verlag, 1977.

<http://www.scientific-journals.org>

- [59] [Sottet08] Sottet, J.S., Calvary, G., Coutaz, J., Favre, J.M. "A Model-Driven Engineering Approach for the Usability of Plastic User Interfaces" in the proceedings of Engineering Interactive Systems 2007, University of Salamanca, Spain, J. Gulliksen et al. (eds), LNCS 4940, pp. 140-157, 2008.
- [60] [Szekely96] Szekely P., "Retrospective and Challenges for Model-Based Interface Development, Design, Specification and Verification of Interactive Systems'96", Proceedings of the Eurographics Workshop in Namur, Belgium, F. Bodard, J. Vanderdonck (eds), 1996.
- [61] [Sztipanovits97] Sztipanovits, J. , Karsai, G., "Model-Integrated Computing", Computer, vol. 30, no. 4, pp. 110-111, 1997.
- [62] [Thevenin01] Thevenin, D., Calvary, G., Coutaz, J. "A Development Process for Plastic User Interfaces" in Proceedings of CHI'01 workshop, Seattle,WA, USA, 2001.
- [63] [Tohidi06] Tohidi, M., Buxton, W., Baecker, R., Sellen, A. "Getting the right design and the design right", Proceedings of the SIGCHI conference on Human Factors in computing systems, Montréal, Québec, Canada, pp. 1243-1252, 2006.
- [64] [UIMS92] UIMS, "A Metamodel for the Runtime Architecture of an Interactive System", The UIMS Tool Developers Workshop, SIGCHI Bull., ACM, 24, 1, pp. 32-37, 1992.
- [65] [van Deursen00] van Deursen, A., Klint, P., Visser, J. "Domain-specific languages : An annotated bibliography", SIGPLAN Notices, 35(6), pp. 26-36, 2000.
- [66] [Wegner97] Wegner, P. "Why interaction is more powerful than algorithms". Communications of the ACM, 40(5), pp. 80-91, 1997.
- [67] [Weiser91] Weiser, M. "The computer of the 21st century". Scientific American, 265(3), pp. 66-75, 1991.