5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010)

Bridging between User Experience and UI Engineering

Proceedings

April 10, 2010

Atlanta, Georgia, USA

organized at the 28th ACM Conference on
Human Factors in Computing Systems (CHI 2010)

Editors:

Jan Van den Bergh (Hasselt University, Belgium)

Stefan Sauer (s-lab, University of Paderborn, Germany)

Kai Breiner (Fraunhofer IESE, TU Kaiserslautern, Germany)

Heinrich Hußmann (University of Munich, Germany)

Gerrit Meixner (German Research Center for Artificial Intelligence, Germany)

Andreas Pleuss (Lero, University of Limerick, Ireland)

Table of Contents

Preface

In the history of software development, the abstraction level on which software is described has been increasing all the time. The latest trend is to specify software using (platform-independent) models, which are then gradually and (semi-) automatically transformed into executable applications for different platforms and target devices. The goal of the 5th Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI 2010) is to discuss the use of (semi-) formal models in user interface development. What is the right way of creative design and interaction design working with user interface models? Can models help in coping with innovative interaction techniques?

The workshop is a platform for discussing the modelling of advanced user interfaces, such as interfaces supporting complex interactions, visualizations, multimedia representations, multimodality, adaptability or customization. It is intended to contribute to a better integration of knowledge from the Human-Computer Interaction community and the Software Engineering community. We are striving for development methods which support the model-driven development of user interfaces with great user experience and optimal usability. Which kinds of models are required to achieve this goal? How can different kinds of models be flexibly combined? How can individual and informal design skills be integrated into the process? Can (and shall) we integrate the tools designers are accustomed to use nowadays? Is it possible to do user-centred design based on models?

The MDDAUI program committee has selected 13 high-quality papers from the submitted contributions in a selective peer review process. These papers present novel, innovative, and exciting work in this dynamic area of research and development. They are intended to present the current state of the art, fuel discussions at the workshop, and possibly raise new research directions.

We produced and included a workshop report in these proceedings that summarizes the vivid discussions at the workshop. We also included the workshop poster that was presented at the CHI 2010 conference.

We thank all workshop participants for their valuable contributions to an interesting MDDAUI workshop at CHI 2010. We are looking forward to the next edition of our workshop series.

Jan Van den Bergh
Stefan Sauer
Kai Breiner
Heinrich Hußmann
Gerrit Meixner
Andreas Pleuss

Workshop Organizers

Jan Van den Bergh (Hasselt University, Expertise Centre for Digital Media, Belgium).

Stefan Sauer (University of Paderborn, s-lab – Software Quality Lab, Germany)

Kai Breiner (Fraunhofer-Institute for Experimental Software Engineering (IESE) and TU Kaiserslautern, Software Engineering Research Group, Germany)

Heinrich Hußmann (University of Munich, Media Informatics Group, Germany)

Gerrit Meixner (German Research Center for Artificial Intelligence, Center for Human-Machine Interaction, Germany)

Andreas Pleuss (Lero, University of Limerick, Ireland)

Program Committee

Kai Breiner, Fraunhofer-Institute for Experimental Software Engineering, Germany
Gaelle Calvary, University Joseph Fourier, France
Peter Forbrig, University of Rostock, Germany
Phil Gray, University of Glasgow, GB
Heinrich Hußmann, University of Munich, Germany
Youn-kyung Lim, KAIST, South-Korea
Kris Luyten, Hasselt University, Belgium
Gerrit Meixner, German Research Center for Artificial Intelligence, Germany
Philippe Palanque, University Paul Sabatier, France
Fabio Paternò, C.N.R. Pisa, Italy
Andreas Pleuss, Lero, Ireland
Angel Puerta, RedWhale Corp., USA
Harald Reiterer, University of Konstanz, Germany
Stefan Sauer, University of Paderborn, Germany
Orit Shaer, Wellesley College, USA
Gerd Szwillus, University of Paderborn, Germany
Jan Van den Bergh, Hasselt University, Belgium
Jean Vanderdonckt, Université Catholique de Louvain, Belgium
Detlef Zuehlke, German Research Center for Artificial Intelligence, Germany

Additional Reviewers

Florian Geyer
David Navarre
Dawid Ostrowski
Marco Winckler

Self-Explanatory User Interfaces by Model-Driven Engineering

Alfonso García Frey, Gaëlle Calvary and Sophie Dupuy-Chessa
University of Grenoble, CNRS, LIG
385, avenue de la Bibliothèque, 38400, Saint-Martin d'Hères, France
{Alfonso.Garcia-Frey, Gaelle.Calvary, Sophie.Dupuy}@imag.fr

ABSTRACT
Modern User Interfaces (UI) must deal with the increasing complexity of applications as well as new features such as the capacity of UIs to be dynamically adapted to the context of use. The complexity does not necessarily imply a better quality. Thus, it becomes necessary to make users understand the UIs. This paper describes an on-going research about Self-Explanatory User Interfaces (SE-UI) by Model-Driven Engineering (MDE). Self-explanation makes reference to the capacity of a UI to provide the end-user with information about its rationale (which is the purpose of the UI), its design rationale (why is the UI structured into this set of workspaces?, what's the purpose of this button?), its current state (why is the menu disabled?) as well as the evolution of the state (how can I enable this feature?). Explanations are provided by embedded models. We explore model-driven engineering to understand why and how this approach can lead us to overcome shortcomings of UI quality successfully.

Author Keywords
Self-Explanatory User Interfaces, UI quality, help, design rationale, model-driven engineering, model transformation.

ACM Classification Keywords
H.5.2 User Interfaces: Theory and method.

INTRODUCTION
Motivation
On the one hand, most software is too hard to use."Modern applications such as Microsft Word have many automatic features and hidden dependencies that are frequently helpful but can be mysterious to both novice and expert users" [15]. Users may require assistance while interacting with a User Interface (UI). Ideally, the UI must guide the user in accomplishing a task the application was designed for. The user can request help about functionality, features, or any information about the process of the task that is being performed. The UI must be able to provide the correct answer, giving the necessary information to the user in an appropiate format. This can take place at any time in the whole interaction process between both the user and the UI. However, modern applications cover only a few questions the user may have, or provide a general help instead of a clear and concise answer to a given question. Furthermore, help is created ad-hoc, this is, it has been previously generated and it's not able to cover new questions at run-time because they were not considered by the designers. UI design problems are not covered at all because the designers are not aware of them.

Moreover, the UI must deal with users having different levels of expertise. Even many long-time users never master common procedures [6] and in other cases, users must work hard to figure out each feature or screen [6].

The problem is greater for Plastic UIs [5, 19]. Plastic UIs demand dynamic adaptation also for help systems because from now on, developers can't afford to consider all the different contexts of use one by one coding all possible ad-hoc solutions by hand. This complicates the prediction of the result and the final quality, making difficult the design choices.

As a result, dynamic solutions are required also for help systems. These help systems must now be aware of the context of use (user, platform and environment), the task, the structure and presentation of the UI.

MDE and MB-UIDE approaches
On the other hand, Model-Driven Engineering (MDE) exists since long time ago and its recently applied to the engineering of UIs. It consists in describing different features of UIs (e.g., task, domain, context of use) in models from which a final UI is produced [18] according to a forward engineering process. MDE of UI is assumed to be superior to the previous Model-Based User Interface Development Environment versions since it makes the UI design knowledge explicit, and external for instance as model-to-model transformations and model-to-code compilation rules [2]. However, neither Model-Based User Interface Development Environment automatic generated UIs nor final UIs produced by MDE have enough quality, forcing designers to manually tweak the generated UI code [2]. Design knowledge can not be always explicitly represented into the models, but it has a potential to help final users. Some models as for instance the task model have this potential explicitly represented, and they can contribute also to guide and help the user.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

This research will study how Self-Explanatory User Interfaces (SE-UIs) can be built using the MDE. A SE-UI is a UI with the capacity of understanding its own rationale and consequently having the abilities of answer questions about it. We aim to provide a method for creating SE-UIs analyzing the relations between the different levels of abstraction in our MDE-compliant approach for developing UIs as well as the different models presented into the UsiXML specification and their relations. Complementary views of the UI are also considered into this research.

The rest of the paper presents the related work and our contribution to the field.

RELATED WORK

The two major areas involved in our Self-Explanation approach are MDE and UI quality. The related works of the next two sections allow us to set up the bases of our contribution.

MDE

The Cameleon Reference Framework [4] presented a MDE-compliant approach for developing UIs consisting of four different levels of abstraction: Task Model, Abstract User Interface, Concrete User Interface and Final User Interface. These levels correspond, in terms of MDE, to Computing-Independent Model (CIM), Platform-Independent Model (PIM), Platform-Specific Model (PSM) and the code level respectively. In the Model-Driven Development (MDD) many transformation engines for UI development have been created. Several researches have addressed the mapping problem for supporting MDD of UIs: Teresa [14], ATL [10], oAW [10] and UsiXML [17] among others. A comparative analysis can be found in [9]. Semantic Networks have been also covered for UIs [8]. The Meta-UI concept was firstly proposed in [7] and deeply explored later in many other works. In one of them [16], the concept of Mega-UI is studied introducing Extra-UIs, allowing a new degree of control by the use of views over the (meta-)models. We will focus on it later as these views are relevant for the explanation of the UI and consequently for the end-user's comprehension.

UIs Quality

Help systems have been extensively studied. One of the most relevant works is the Crystal application framework [15]. Inspired by the Whyline research [11], "Crystal" provides an architecture and interaction techniques that allow programmers to create applications that let the user ask a wide variety of questions about why things did and did not happen, and how to use the related features of the application without using natural language [15]. Even if this approach does not cover the capacity of adaptation to different contexts of use, it represents an important improvement in quality for the end-user in terms of achieved value. Quality can be improved regarding not only the *achieved value*, but also from the perspectives of *software features* and *interaction experiences* [12]. The integration of Usability Evaluation Methods (UEM) [13] into a MDA process has been proved to be feasible in [1]. In particular, the evaluation at the PIM or PSM should be done in an interactive way until these models have the required level of usability. Different UEMs (e.g., heuristic evaluation, usability test, etc) can be applied iteratively until the concerned models have the required level of usability. A set of ergonomic criteria for the evaluation of Human-Computer Interaction (HCI) can be found in [3].

This research improves quality of help systems allowing a new range of questions. Adaptation to the context of use is now considered since SE-UIs understand their own rationale.

CONTRIBUTION

End-User's point of view

The goal of this work is to study how SE-UI can be built by MDE. One of the ways to explore SE-UI involves the task model and its rationale. A task model describes the user's task in terms of objectives and procedures. Procedures recursively decompose tasks into subtasks until one or more elementary tasks are reached, i.e., tasks which would be decomposable into physical actions only ("press the button"). A task model is well-defined then by the following terms:

Nodes Containing abstract tasks

Leaves Special nodes containing elementary tasks

Branches Expressing logical and temporal relations between tasks, subtasks and elementary tasks

The explicit information contained into the branches can help and guide the end-user answering questions related to different aspects of the UI. For instance, regarding the rationale of the UI questions like *which is the purpose of the UI?* can be successfully answered; also, questions as *why is the UI structured into this set of workspaces?* or *what is the purpose of this button?* can be explained understanding the relations of the design rationale. The current state of the UI and consequently the state of the application, can trigger a different kind of questions to the end-user as for instance *why is the menu disabled?*, as well as questions related to the overall progress of a task or questions about the evolution of the current state of the application as for example *how can I enable this feature?* Answers for all of them can be obtained exploring tasks and subtasks (nodes), elementary tasks (leaves) and relations between them (branches) in the task model.

This work will study also how different views of the model centered in extra-UIs, can help the end-user to understand the UI. A extra-UI [16] is a UI which represents and gives the control of a UI through a model. It is in a sense the UI of the configuration of a UI. These views can improve the end-user's comprehension as they are relevant for the explanation of the UI. Extra-UIs provide a new degree of control over the (meta-)models of the UI; both designer and end-user can see and understand how tasks are decomposed and how tasks are represented in a specific UI. In other words, how the UI is interfacing the interaction between the application and the own user. Designers can express this interaction in the form of relations between tasks and elements of the final UI with the method explained in the next section.

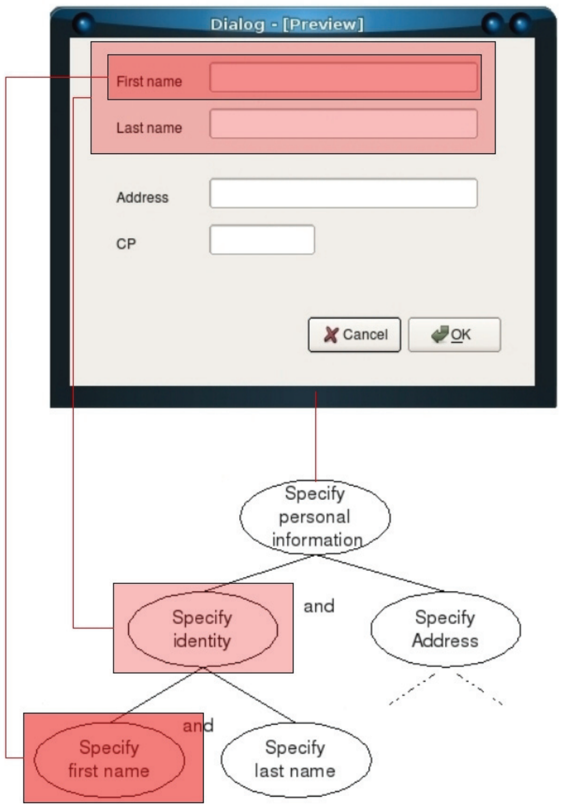


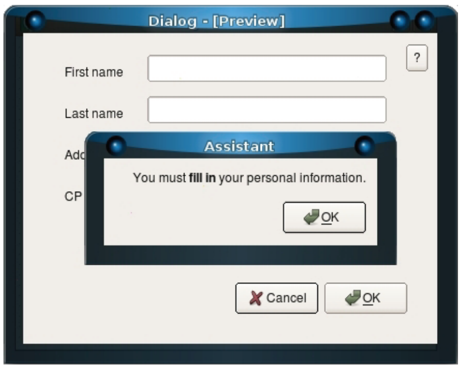
Figure 1. Association between UI and a task model.

Designer's point of view

This work will explore a method to provide designers with a technique to add Self-Explanation to final UIs, specifying how end-user's tasks are directly related to the final UI level. The method consists in four steps:

1. Specify the final UI of the model-compliant application that it will be extended with SE functionality.
2. Define the task model of the application.
3. Specify the relations between both the task model and the final UI.
4. A new final SE-UI will be generated from these relations, adding SE functionality in real-time.

To support this method, we will supply designers with an editor in which tasks models and final UIs can be created. Both of them will coexist at the same time into the same workspace inside this editor. Once the task model and the final UI are represented, the designer will draw direct connections between elements of the task model and elements of the final UI, linking for instance, widgets with subtasks, as we can see in figure 1. Here, the task called *Specify identity* is visually connected to a group of widgets, containing two labels and two input fields. Then, the elementary task *Specify first name* which is also a subtask, is connected to a new subgroup of two widgets, one label and one input field.



You must **fill in** your personal information.

Figure 2. Help message derived from connections in Figure 1.

The purpose of the method is to allow designers to specify direct relations between tasks and different elements of the final UI. The main advantage for designers is that from now on, there is no need of a deeply comprehension of all the model-to-model and model-to-code transformations between all the four levels of MDE. A visual representation gives direct information about these relations because connections are explicitly represented in a visual render, in which the final UI and the task model levels share the same workspace.

To allow end-user questions this study will consider a help button (figure 2) as a first approach. Other approaches can be considered as well. By clicking this help button, the application enters in a help mode where the end-user can ask about different elements of the UI just by clicking on them. Answers will be generated in real-time in different ways. The following section illustrates an example of this procedure.

Answering questions

This work will study also how different questions can be answered. The first approach will associate a description to each element (tasks, relations, widgets, etc.) of figure 1. Other approaches like semantic networks [8] can be considered in the future. If the end-user asks himself, for instance, *Why is the OK button disabled?*, by clicking on this button using the special help mode, the system can say that the task is not completed. In figure 2 the message is dynamically derived from the relations of figure 1. For an edit box, the application can say *You must fill in* + *Description of the task*, where *your personal information* is the description. A more specific information can be generated exploring the task model. For instance, we can travel all the subtasks of the uncompleted task. In the example before, we can answer also that the user needs to fill in the first name and the last name, because these subtasks are both uncompleted.

CONCLUSION

This research takes a significant step forward in the development of high quality UIs. It explores MDE of UIs to provide Self-Explanation at run-time, analysing the four levels of the MDE-compliant approach for developing UIs and the different models presented into the UsiXML specification and their relations. Complementary views of the UI are explored in order to exploit these models, explaining the UI itself and giving to the user a new dimension of control by these views. This opens the work on End-User programming.

ACKNOWLEDGMENTS

This work is funded by the european ITEA UsiXML project.

REFERENCES

1. S. Abraho, E. Iborra, and J. Vanderdonckt. *Maturing Usability*, chapter Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool, pages 3–32. Human-Computer Interaction Series. Springer-Verlag, 2008.

2. N. Aquino. Adding flexibility in the model-driven engineering of user interfaces. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 329–332, New York, NY, USA, 2009. ACM.

3. J. C. Bastien and D. L. Scapin. Ergonomic criteria for the evaluation of human-computer interfaces. 0 RT-0156, INRIA, 06 1993.

4. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. *Interacting With Computers Vol. 15/3*, pages 289–308, 2003.

5. B. Collignon, J. Vanderdonckt, and G. Calvary. Model-driven engineering of multi-target plastic user interfaces. In *Proc. of 4th International Conference on Autonomic and Autonomous Systems ICAS 2008*, pages 7–14, 2008. D. Greenwood, M. Grottke, H. Lutfiyya, M. Popescu (eds.), IEEE Computer Society Press, Los Alamitos, Gosier, 16-21 March 2008.

6. M. Corporation. Microsoft inductive user interface guidelines, 2001. http://msdn.microsoft.com/en-us/library/ms997506.aspx.

7. J. Coutaz. Meta-user interfaces for ambient spaces. In *Tamodia'06*, 2006. 8 pages.

8. A. Demeure, G. Calvary, J. Coutaz, and J. Vanderdonckt. Towards run time plasticity control based on a semantic network. In *Fifth International Workshop on Task Models and Diagrams for UI design (TAMODIA'06)*, pages 324–338, 2006. Hasselt, Belgium, October 23-24, 2006.

9. J. González Calleros, A. Stanciulescu, J. Vanderdonckt, D. J.P., and M. Winckler. A comparative analysis of tranformation engines for user interface development. In *Proc. of the 4th International Workshop on Model-Driven Web Engineering (MDWE 2008)*, pages 16–30, Tolouse, France, 2008. CEUR Workshop Proceedings.

10. F. Jouault and I. Kurtev. Transforming models with atl. In *Satellite Events at the MoDELS 2005 Conference*, volume 3844 of *Lecture Notes in Computer Science*, pages 128–138, Berlin, 2006. Springer Verlag.

11. A. J. Ko and B. A. Myers. Designing the whyline: a debugging interface for asking questions about program behavior. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 151–158, New York, NY, USA, 2004. ACM.

12. E. Lai-Chong Law, E. T. Hvannberg, and G. Cockton. *Maturing Usability. Quality in Software, Interaction and Value*. Human-Computer Interaction Series. Springer-Verlag, 2008.

13. E. L. Law, E. T. Hvannberg, G. Cockton, P. Palanque, D. Scapin, M. Springett, C. Stary, and J. Vanderdonckt. Towards the maturation of IT usability evaluation (MAUSE). In *Human-Computer Interaction - INTERACT 2005*, pages 1134–1137. 2005.

14. G. Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520, 2004.

15. B. A. Myers, D. A. Weitzman, A. J. Ko, and D. H. Chau. Answering why and why not questions in user interfaces. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 397–406, New York, NY, USA, 2006. ACM.

16. J.-S. Sottet, G. Calvary, J.-M. Favre, and J. Coutaz. *Megamodeling and Metamodel-Driven Engineering for Plastic User Interfaces: Mega-UI*. 2007.

17. J. Vanderdonckt. A MDA-Compliant environment for developing user interfaces of information systems. In *Advanced Information Systems Engineering*, pages 16–31. 2005.

18. J. Vanderdonckt. Model-driven engineering of user interfaces: Promises, successes, failures, and challenges. In *Proc. of 5th Annual Romanian Conf. on Human–Computer Interaction ROCHI'2008, (Iasi, 18–19 September 2008), pp. 1—10. Matrix ROM, Bucarest*, 2008.

19. J. Vanderdonckt, J. Coutaz, G. Calvary, and A. Stanciulescu. *Multimodality for Plastic User Interfaces: Models, Methods, and Principles*, chapter 4, pages 61–84. 2008. D. Tzovaras (ed.), Lecture Notes in Electrical Engineering, Springer-Verlag, Berlin, 2007.

A Multimodal User Interface Model For Runtime Distribution

Dirk Roscher, Marco Blumendorf, Sahin Albayrak
DAI-Labor, TU-Berlin
Ernst-Reuter-Platz 7, D-10587 Berlin, Germany
{Dirk.Roscher, Marco.Blumendorf, Sahin.Albayrak}@DAI-Labor.de

ABSTRACT

Smart environments provide numerous networked interaction resources (IRs) allowing users to interact with services in many different situations via many different (combinations of) IRs. In such environments it is necessary to adapt the user interface dynamically at runtime to each new situation to allow an ongoing interaction in changing contexts. Our approach allows to dynamically select the combination of IRs that are suitable for the interaction in the current context at any time. The decision is based on information from a user interface model executed at runtime and a context model gathering information about the environment. The user interface model supports the CARE properties to specify flexible multimodal interaction.

Author Keywords

User interface distribution, model-based development, executable models

INTRODUCTION

Smart environments provide numerous networked interaction resources (IRs) allowing users to interact with services in many different situations via many different (combinations of) IRs. In such environments user interfaces (UIs) need to take a changing context of use into account [6]. This makes the alteration of the used IRs and thus the dynamic (re-) combination of the resources at runtime an important aspect.

In this work, we first present the requirements to dynamically adjust the used IRs at runtime (section 3). Afterwards, our model-based approach targeting the requirements is described. A UI model reflects the various UI elements as well as the relations between them in terms of CARE properties [7] to achieve multimodal interaction (section 4). At runtime, the modeled UI description in combination with information about the available IRs from an additional context model is used to continuously adjust the UI distribution according to the current situation (section 5). Before we describe our approach, related work is presented in the next section.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

RELATED WORK

Model-based development is a promising approach in the context of multimodal UIs. According to the classification of the Cameleon Reference Framework proposed in [6] UI models feature four levels of abstraction: Concepts and Task Model, Abstract, Concrete and Final User Interface. Based on this general framework, several User Interface Description Languages (UIDLs) have been designed. The most relevant with respect to the goals of this work are UsiXML [11] and TERESA [2]. Their goal is to develop multimodal UIs but they only support a fixed set of IRs, whereas we aim to support sets of IRs changing at runtime.

The distribution of UIs has also been a topic of various research activities, ranging from the characterization of distributed UIs [8] to development support for specifying distributed UIs [12]. The approach of Elting and Hellenschmidt [9] supports simple conflict resolution strategies when distributing output across graphical UIs, speech syntheses and virtual characters. The main goal is the semantic processing of input and output in distributed systems. The dynamic redistribution and definition of dynamic UI models has thus not been the focus of the approach. The I-AM (Interaction Abstract Machine) system [1] presents a software infrastructure for distributed migratable UIs. It provides a middleware for the dynamic coupling of IRs to form a unified interactive space. The approach supports dynamic distribution across multiple heterogeneous platforms but does not support the arbitrary recombination of IRs and is limited to graphical output as well as mouse and keyboard input.

Our approach utilizes the modeled design information at runtime to dynamically adjust the combination of the used IRs. In the following we first describe the requirements that need to be fulfilled to allow the distribution of multimodal UIs at runtime. Afterwards, we show how these requirements are implemented by our approach.

DYNAMIC UI DISTRIBUTION

To support UI (re-) distribution at runtime several requirements needs to be fulfilled, that are derivated from the abstract architecture depicted in Figure 1:

1. A user interface description is needed that supports different variants of multimodal interaction and benefits from the advantages of specific modalities and modality combinations. Information about the supported modality combinations need to be available at runtime.

2. To combine IRs from different platforms, the user interface description needs to address single IRs.

3. Environment information must be gathered and kept up to date (e.g. IRs, users and their positions).

4. An instance that incorporates information about the UI and the environment is required which determines the most appropriate combination of IRs at any time.

5. The combination of arbitrary IRs from different platforms also requires a mechanism that allows to control IRs independently from each other.

The first two requirements are fulfilled by our multimodal UI model with different presentations and input possibilities as described in the next subsection. Afterwards we show how this model is used to create multimodal UIs by selecting the most appropriate combination of IRs (requirements 3 to 5).

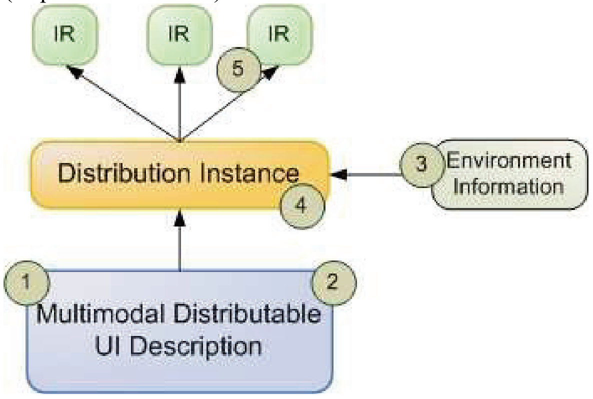


Figure 1: Abstract architecture for distributing UIs.

MULTIMODAL EXECUTABLE UI MODEL

The distribution of UIs at runtime requires certain information about the UI at runtime (part of requirement 1 and 2). To achieve this, our approach is based on the notion of executable models that combine the static design information, execution logic and runtime state information of the UI [4]. This allows to execute and observe their status at runtime as well as to access design information.

Our set of metamodels for specifying distributable multimodal UIs follows the Cameleon reference Framework [6] and thus we distinguish tasks and concepts, abstract interface, concrete interface and final interface as also done in TERESA and UsiXML (see Figure 2). To show how to develop a UI with our set of models, we explain the short example presented in Figure 2. The example is an excerpt from our cooking assistant and models a recipe selection scenario. The presentation of the recipe is possible via different modality combinations and can be confirmed via several input styles.

We first specify the workflow of the example with a task model (we use an extended version of Concurrent Task Trees for the definition [10]) and thus start with the definition of the "ConfirmSelection"-task (*T1: ConfirmRecipe*). Afterwards, the abstract interaction(s) for each task is specified by choosing between *OutputOnly*, *FreeInput*, *Choice* and *Command* (similiar to UsiXML and TERESA) or *ComplexInteractor* to aggregate several abstract interactions. So we choose one abstract interaction object for the presentation of the selected recipe (*A1:OutputOnly*) and one for the confirmation (*A2:Selection*). The abstract interactors are connected via mappings to the "ConfirmSelection"-task (see Figure 2). This is one huge difference between transformational approaches like UsiXML and the executable models approach. Because of the parallel execution of all models (task, abstract and concrete), the information from all models is available and does not need to be transformed from one model into another. Each model only contains the information of its abstraction level and information from different models is connected via mappings.

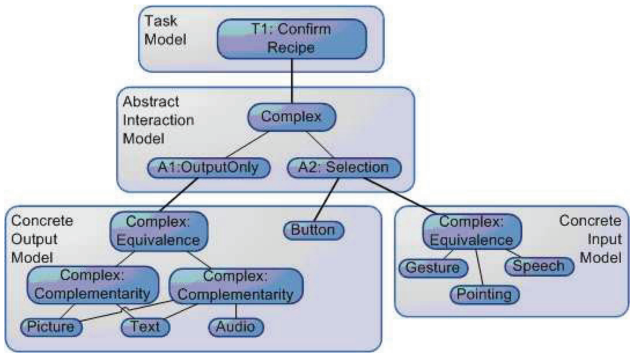


Figure 2: Model structure and interaction example.

In the next step the concrete interaction is specified by using the concrete input and concrete output model. The separation of input and output is another difference to other approaches but allows the independent addressing of single IRs (requirement 2). However, it requires to handle IRs with combined input and output like touchscreens. By utilizing the CARE properties, developers can specify their intention on how to combine the different modalities (requirement 1). Defining multimodal relations with the CARE-properties is similar to the ICARE software components [5]. In contrast to ICARE however, the components and thus the multimodal relationships are not statically related at design time but can be freely configured between arbitrary modalities through the integration in the interaction metamodel and evaluated at runtime.

To present the recipe (*A1:OutputOnly*) the developer chooses two different presentation possibilities: one for graphical output (*Picture* and *Text*) and one with additional natural language output (*Audio*). Each possibility is aggregated by a complex interactor and the Complementarity-attribute means that the children complement each other and must be presented together.

Both possibilities are also aggregated by a complex interactor with an Equivalence-attribute, marking both possibilities as equivalent. The confirmation of the recipe (*A2:Selection*) has only a graphical presentation (*Button*) which is directly mapped to the abstract interactor.

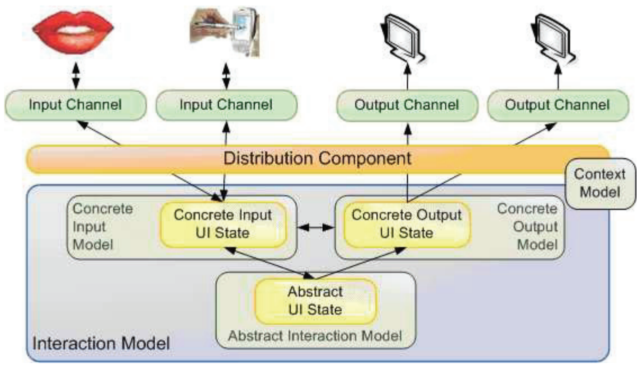


Figure 3: Overview of the runtime architecture.

Beneath the used possibilities, the concrete output model supports *SignalOutput*-elements to include more limited modalities like blinking lights or haptic feedback (vibration) and *DynamicOutput* to create multiple output, e.g. for a dynamically created number of elements.

To confirm the recipe, the developer provides three possibilities (*Gesture, Speech* and *Pointing*), which are aggregated by a complex interactor with an Equivalence-attribute, defining that they all can be used to provide the same input to the system (Figure 2). The next section describes how the modeled description is used within a runtime architecture to deliver flexible multimodal interaction.

RUNTIME DISTRIBUTION

Based on the needed components and the requirements that need to be fulfilled to realize the anticipated dynamic distribution at runtime, we developed and implemented the runtime architecture depicted in Figure 3. The different components and their behavior are described next.

On the execution of the set of models, the central task model calculates a set of active tasks. This triggers the mappings that are connected with each task and results in the activation of the mapped abstract interactors. The mappings connected to the abstract interactors are in turn triggered and the result is a set of active complex CUI elements in the concrete input and output model provided to the distribution component.

The second information provider is a context model that includes different observers to get information about the available IRs (requirement 3). IRs are connected to our runtime system via so called channels [3]. One channel is responsible for establishing and maintaining the connection to one IR (if needed via a network). This includes not only the registration within the context model but also the capability to receive and send information to the IRs. This concept allows the independent addressing of the IRs over platform borders (requirement 4).

We have implemented different channels which connect various interaction technologies, including browsers for graphical output through an AJAX-based channel implementation or connect automatic speech recognition via Dragon Natural Speaking and Text-To-Speech engines by implementing the Media Resource Control Protocol (MRCP). All models (user interface and context) are implemented with the Eclipse Modelling Framework (EMF). The direct mapping of EMF to Java supports the bridging of model and the distribution component on the implementation level which allows the distribution component to observe the models for state changes.

The distribution component is notified whenever a new set of concrete interactors is activated, and matches the supported modalities to the available modalities of the available IRs by adhering to the following goals:

Input: support as many (equivalent) input resources as possible while considering the specified CARE relations between the input elements. This aims at leaving the control about the used IRs to the user by supporting a wide range.

Output: find the most suitable combination of output resources while considering the specified CARE relations between the output elements. Distributed output thus aims at utilizing the most suitable combination of IRs to convey the UI. The selection of IRs depends on their capabilities and context information like the resource location.

The algorithm first determines the IRs that can be utilized by the user. Therefore the available IRs are queried from the context model together with information about the premises and localization and direction information. Based on the type of the IRs, the algorithm calculates if the resources are currently usable. E.g. displays are considered usable when they are within the vicinity of a user and haptical input IRs are considered usable when they are within the range of the user. The resulting set of usable IRs determines the usable modalities and thus the types of concrete interactors that can be distributed.

In the next step the algorithm analyzes the CARE relations of the active concrete interactors. The specified UI model contains trees of complex interaction elements with simple elements as leaf nodes. As only the leaf nodes have to be distributed, the relations defined by their parent complex interactors influence their distribution. The simple interactors are automatically of type "assigned" and can thus be directly distributed if a corresponding type of IR is available. Interactors combined via complex elements of type complementary or redundant must be distributed together to reflect their meaning. This means that to make an interaction, defined as redundant, available to the user, all modalities addressed by the childs of the complex interactor have to be available. The equivalence relation is used to specify different (combinations of) interactors that transport the same information in case of output or allow the user to provide the same information in case of input. This makes the system more reliable and reduces ambiguity and inconsistency. With respect to the distribution goals specified above, a different handling of the equivalency relation for input and output has been realized. For input the distribution of as many equivalent interactors as possible results in more possibilities for the user to provide the needed input. For output a selection of the most feasible interactors avoids confusion and unwanted redundancy.

Based on these interpretations of the CARE relationships the algorithm first calculates the distribution of the output interactors. The algorithm decides between the different equivalent interactor combinations by selecting the one supporting the most modalities. This is based on the assumption that the designer utilizes the advantages of each modality, so that more modalities result in a better presentation. More sophisticated extensions that consider additional context information are currently evaluated. Afterwards, the distribution of input interactors is calculated. Thereby the algorithm distributes all elements that are supported by the usable IRs to allow as many input possibilities as possible. It is crucial that during the distribution of the input interactors the algorithm pays attention to coupled input and output as e.g. in case of a touchscreen. The resulting distribution configuration consists of tuples of concrete interactors and IR references. Before sending the interactors to the channels, the presentation of the output interactors is accomplished by a layouting algorithm [11] which takes into account the spatial and temporal relationships of the interactors as well as the workflow model to not scatter related interactors.

In case of the little example, the algorithm would distribute the interactors as follows: For input the algorithm tries to support as many IRs as possible and thus determines at maximum the gesture, voice input and pointing interactors to support a keyboard, a microphone and a mouse respectively. For output a screen is required and an optional loudspeaker would be integrated if available. The algorithm would adapt the distribution accordingly, when e.g. the user is changing the position and the distribution component determines that some IRs are no longer available to the user and others just became available.

CONCLUSION

We presented an approach for dynamically selecting the IRs at runtime. Based on the modeled modality relations defined as CARE properties, which are available at runtime due to the utilization of executable models, and information about the actual context, a distribution algorithm calculates the most appropriate set of IRs.

In the future we plan to develop a multimodal widget set to ease the development of such multimodal and distributable UIs. We also want to analyze further factors that influence the distribution algorithm. Furthermore, automatic calculation raises the problem of unsatisfactory results. To overcome this issue for distribution of UIs, we started the development of a meta user interface allowing users to configure the distribution according to their needs.

REFERENCES

1. N. Barralon, J. Coutaz, and C. Lachenal. Coupling interaction resources and technical support. In *HCI International 2007*, Volume 4555 of LNCS, pages 13-22, 2007.
2. S. Berti, F. Correani, G. Mori, F. Paternò, and C. Santoro. Teresa: A transformation-based environment for designing and developing multi-device interfaces. In *CHI 2004*, volume II, pages 793-794, 2004.
3. M. Blumendorf, S. Feuerstack, and S. Albayrak. Multimodal user interaction in smart environments: Delivering distributed user interfaces. In *Constructing Ambient Intelligence*, AmI 2007 Workshops Darmstadt.
4. M. Blumendorf, G. Lehmann, S. Feuerstack, and S. Albayrak. Executable models for human-computer interaction. In *Proc. of the DSV-IS Workshop 2008*, pages 238-251, Berlin, Heidelberg, 2008.
5. J. Bouchet, L. Nigay, and T. Ganille. Icare software components for rapidly developing multimodal interfaces. In *Proc. of ICMI 2004*, pages 251-258, New York, USA, 2004.
6. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, Laurent Bouillon, and Jean Vanderdonckt. A unifying reference framework for multi-target user interfaces. In *Interacting with Computers*, 15(3):289-308, 2003.
7. J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. M. Young. Four easy pieces for assessing the usability of multimodal interaction: The care properties. In *INTERACT 1995*, pages 115-120, 1995.
8. A. Demeure, J.-S. Sottet, G. Calvary, J. Coutaz, V. Ganneau and J. Vanderdonkt. The 4c reference model for distributed user interfaces. In ICAS 2008. IEEE Computer Society Press.
9. C. Elting and M. Hellenschmidt. Strategies for self-organization and multimodal output coordination in distributed device environments. In *Workshop on Artificial Intelligence in Mobile Systems 2004.*
10. S. Feuerstack, M. Blumendorf, and S. Albayrak. Prototyping of multimodal interactions for smart environments based on task models. In *Constructing Ambient Intelligence*, AmI 2007 Workshops.
11. Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon and V. López-Jaquero. Usixml: A language supporting multi-path development of user interfaces. In *EHCI/DS-VIS*, Volume 3425 of LNCS, pages 200-220. 2004.
12. J.P. Molina, J. Vanderdonckt, P. González, A. Fernández-Caballero and M.D. Lozano. Rapid prototyping of distributed user interfaces. In Proc. of CADUI'2006, pages 151-166. Springer-Verlag, 2006.

An Interactive Process Meta Model for Runtime User Interface Generation and Adaptation

Thomas Schlegel
University of Stuttgart
Universitätsstr. 38, Stuttgart, Germany
Thomas.Schlegel@vis.uni-stuttgart.de
+49 711 7816 209

ABSTRACT

Complex and distributed interactive systems today – in this case mainly in production and supply chain management – rely strongly on defined processes but as well on flexible interaction and dynamic adaptation. We describe an interactive process model that allows recognizing and deriving interactions with users on runtime. The model can be dynamically adapted to fit new requirements or offer additional interactions in specific contexts.

Author Keywords

Interactive Processes, Process Model, Production Process Control, Runtime User Interface Generation

ACM Classification Keywords

H.5.2 User interface management systems (UIMS), H.5.3 Theory and models, H.5.m Miscellaneous, C.3 process control systems, H.4.1 workflow management

INTRODUCTION

Industry has experienced a strong development in the direction of product customization (flexibility in production and products) and globalization (flexibility in organizational structures and local differences), leading to instable processes and user interfaces with frequent adaptations and necessary changes. Industrial systems, in production and supply chain, are developing today from monolithic systems towards a complex set of interconnected systems like Service Oriented Architectures (SOA, [2]). On the one hand, these complex industrial systems like networked production systems or supply networks [13] rely on defined processes to ensure proper and controlled execution of business processes in production companies. On the other hand, build-to-order, Mass Customization [1] and rapid reconfiguration (e.g. [3]) of the factory require flexibility and adaptability of processes that go beyond what is possible today.

Therefore, we experience today **interaction with a "system of systems"** [6], integrating different users and different systems in a dynamic **Multi-Stakeholder Multi-System (MS2) System**, which adds a new level of complexity and necessary flexibility to today's networked industrial systems and their interaction layer. Interactions with such a complex system occur at all levels of the processes. Hence, processes changing dynamically on runtime lead to the requirement of adapting or generating their adjacent user interface even during their execution.

To achieve this flexibility in process modeling and execution, we shortly motivate and present the concepts of interaction in this field, explain the necessary interactive process model integrating static and dynamic aspects in one model and show a first user interface prototype for executing, controlling, adapting and testing interactive processes and interacting with the user based on this model.

INTERACTIVITY IN COMPLEX SYSTEMS

We studied e.g. the production of dishwashers at BSH, where variability is expected to be low compared e.g. to special machines. Even in this company, 1000 variants exist and changes to the process have to be applied with high frequency depending on product changes, location and many other factors, like distributing formerly integrated interaction tasks of the workers.

Users and even software developers working on such an MS2 System normally only perceive and access some local parts of the system – often without a detailed mental model of the whole system, while the system itself may additionally be changing and evolving without notification. Full oversight and control over the processes for one stakeholder or organization is often not possible, if the system is spanning across organizational and technological borders. E.g. an operator in a production cell executes some steps of a complex production process for producing a car part that is assembled by other workers in a different factory of a different company, still participating in the same overall process. This makes decentralization and flexible interaction a key issue for these systems.

The need for integrating interaction with processes has been recognized by the Workflow community, e.g. leading to additional standards like BPEL4People [4] and WS-Human Task [5], which emphasize the importance of the user in process execution. But even these specifications do not fully cover dynamically and decentrally changing interactions in interactive processes. To overcome this issue, interactions cannot be embedded or occur in processes as **pre-programmed or interaction-based user interfaces** with predefined dialogs and functions anymore.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

Flexibility is offered by **model-based user interfaces** (e.g. [11] and especially for workflows [7]), which do not rely on statically predefined functions and dialogs anymore. In an MS2-System with dynamically changing processes, even **dynamic model-based user interfaces** are needed, which additionally allow changes to the underlying workflows and interactions by users on runtime and offer automatable creation of interactions and dialogs based on changing contexts. Disadvantages of such a flexible system include that usability tests of model-based and runtime-generated systems are not fully possible and that the system and models become complex and partially unpredictable and make software development more expensive and complex. Although, for decentralized and runtime-changing infrastructures there is currently no second option available.

A SEMANTIC PROCESS MODEL

Model- and context-based generation means that the underlying process model has to provide the semantics and abilities for generating user interfaces and also mechanisms for safely adapting the process model to new needs arising while the system is already in production state. To achieve a decentralized execution of interactive processes, the underlying process model has to provide the semantics and the dynamics of the processes and the adjacent interactions.

The Process Flow Model Aspect (horizontal aspects)

Common process models like Event-Driven Process Chains (EPC) with BPEL [8] and UML Activity Diagrams [10] focus on the flow aspect of the process. Process steps with their sequence and dependencies are described in a graphical or other language-based model. The **Process Flow Model** contains these horizontal, dynamic aspects of processes necessary for correct execution. It also provides the dependencies between different elements to be able to create interactions, e.g. dialogs which consider causality of inputs. Flow relations determine where data, events and activation have to flow (process definition) or currently flow (process instance).

The Structural Model Aspect (vertical aspects)

While the flow aspect shows the sequencing, the Structural Model defines the vertical, static aspects of a process. Data type, activity types, event types and many other classifications also form an integral part of an executable and self-dependent process model.

Object-oriented inheritance mechanisms are used to create specializations of existing types of actions and interactions in order to provide rich semantics for every component in the model, like process steps, data or dialog elements. Aggregation allows for creating complex processes, data types and semantic groups of interactions, determining their meaning and integration. The inheritance mechanism allows creating specializations of complex, aggregated processes (activities) and their process steps. Figure 1 shows how the concept works for such a process: Activity A consists of Activities x, y and z. Activity B is created as a specialized process (child) of A by inheritance. While x' and z' are created by (sub-)inheritance, i.e. inherited without changes from A (gray, dashed line), y' is specialized from y as a full component (black, continuous line). Through instantiation, process instances are defined by and created from processes (types) and then executed decentrally according to the process specification, e.g. process instance c, which is an object-oriented instance of Activity C, with its components being instances of the Activities x'', y'' and z''.

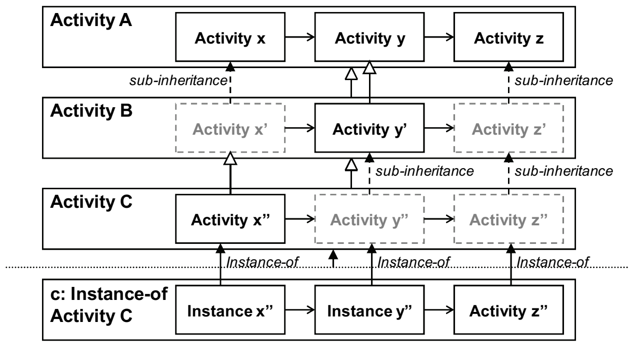


Figure 1. Specialization of aggregated activities by inheritance.

To execute interactive steps, dialog elements can be instantiated from type-compatible interaction element types. For such **explicit interactions** [12], the classification can be used to determine the most specialized interaction element available for the type of interaction requested. For **implicit interactions** [12], it can be used to determine the correct interaction elements to be used to gather the information needed from the users for the current process execution.

Integrated Semantic Process Model

One significant contribution to flexible interactive process systems is that we provide an integrated process and interaction concept, which offers the advantages of object-orientation like runtime behavior, type-safe extension and generation also for interactive processes.

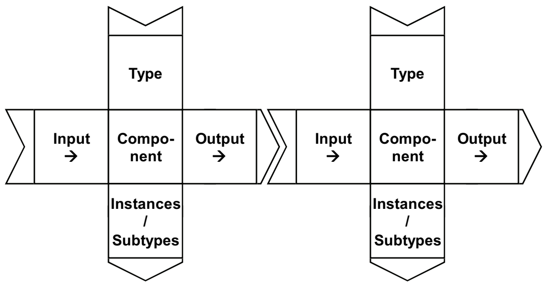


Figure 2. Integrated model containing the structural (vertical, semantic) and dynamic (horizontal, flow) aspects of processes.

To execute interactive processes in a decentralized environment, the integration of both aspects into one model framework is vital. This is done using different component and relation types for static aspects like inheritance / classification and aggregation than for the dynamic aspects like flow/activation, input and output. Figure 2 shows how two components are connected horizontally via their Input and Output interfaces. Inheritance and aggregation offer a vertical connection.

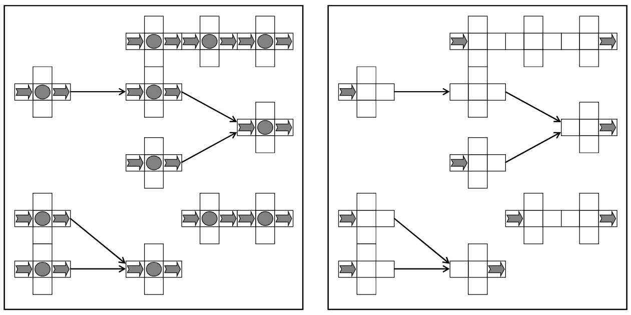


Figure 3. Interface of the component.

For each complex/aggregated component the data needed or produced can be determined semantically by identifying the "interface" of the component. Figure 3 shows how the input and output of every component (e.g. process step) can be connected either internally (arrows between components) or be aggregated as interface for the whole component (right), showing input necessary and output delivered to the next component in the process.

USER INTERFACE CREATION FROM THE MODEL

Once the elements required for execution of the process (or sub-process) have been identified, for each element it can be validated if the necessary data is available. If not, or if an interaction is explicitly foreseen to gather the data from the user [13], the contained User Information Model (UIM) is used to determine the type of the element. With this information, each peer in the system can check the User Interface Element Model (UIEM) to determine if an interpretation of the element is available for the defined context of use. The Dashboard Model is then used to integrate the elements correctly into the overall dialog set.

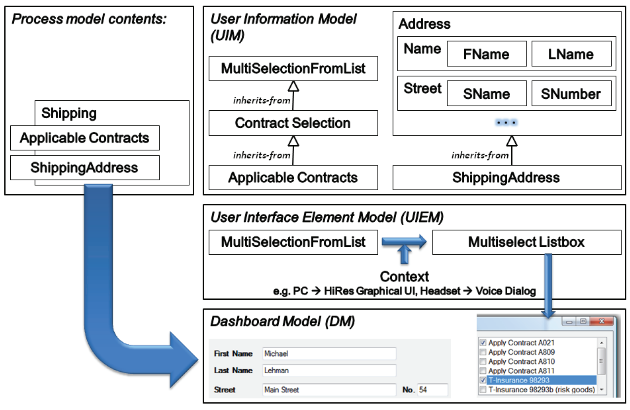


Figure 4. Example for the model-based execution of an interactive shipping process using UIM, UIEM and DM for model interpretation.

In the example in figure 4, we describe a partial view of a typical shipping process in logistics. It requires the shipping address and "Applicable Contracts" (insurance etc.) for the adjacent order and transport. From the UIM, the type can be derived. While for "Applicable Contracts" there is no interaction specification available in the UIEM, it is a specialization of MultiSelectionFromList for which the UIEM provides an implementation.

For ShippingAddress, the UIM provides all elements of it by resolving it from Address. For all partial elements like LName an implementation is derived from the UIEM. The semantic group ShippingAddress can be displayed together with the MultiselectListbox implementation of "Applicable Contracts". MultiselectListbox has been determined to be a valid implementation of MultiSelectionFromList, which in turn is the (still) most specialized generalization of "Applicable Contracts" in the UIM.

To create necessary dialogs for gathering missing information, the type of a missing element (complex or simple) is mapped via reference directly to a corresponding element in the UIM. Each possible realization of an element is listed in the UIEM and references its source element in UIM. For a specific context (e.g. WinFormsApp), the correct renderer in UIM can be identified.

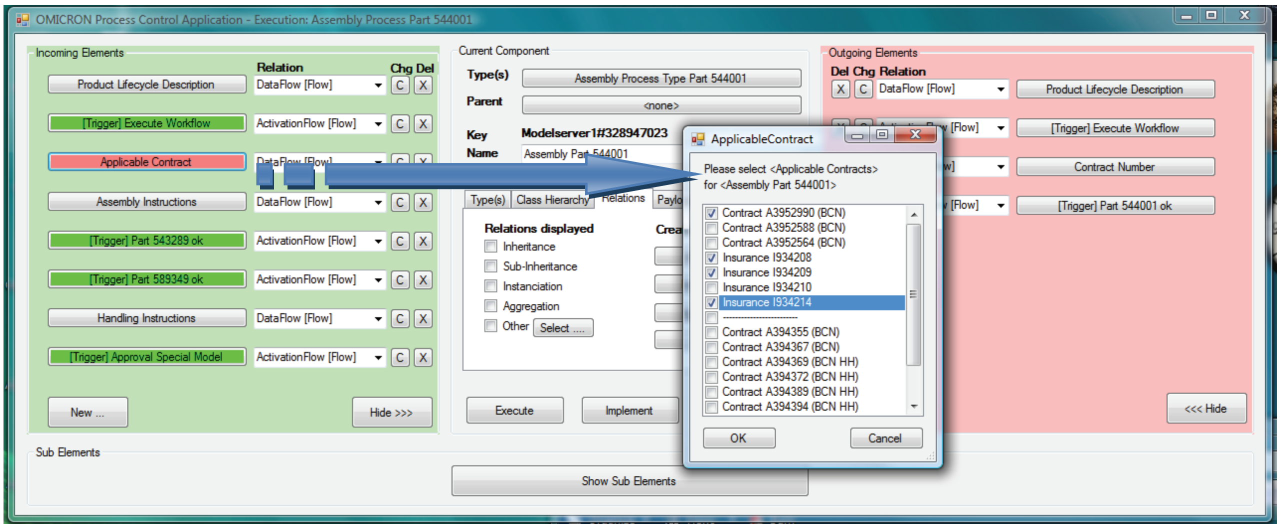
PROTOTYPE FOR MODELING AND EXECUTION

When interactive processes are executed and adapted in a decentralized system environment, it is necessary to provide a modeling and control interface, which is context-oriented. The challenges of this kind of process models are the size and the connections of the model. Each process component has sub-components like input and output and a type hierarchy above and maybe also below as well as flow connections to other components at the same level. A graphical notation like BPMN[9], EPC[8] or UML [10] needs a lot of space, has low content editing abilities and already uses two dimensions without providing the static (vertical) aspects of the model. Therefore we are using a dialog based, navigational form of working with the process model, including execution, editing and generation.

Figure 5 shows a prototype of a dialog-based interface for process execution and editing using an object-oriented process and interaction model. It offers runtime inspection, editing and execution of the interactive process. In the center, information about the currently focused component is displayed and can be edited where possible. Each component has a unique key (identity) and carries a payload and comments with it. The left part contains all incoming data and trigger relations with their types, components and indication if data is missing. The right part contains all results and triggers generated by the current component.

Where possible, elements of the current component can be edited. Also, additional components can be connected using all available relation types. New components can be created directly as input, output or part of the current component in focus.

Navigation is possible **upwards** to the type layer and along the aggregation and inheritance relations, **downwards** to the sub-elements via different relation types like



Please select <Applicable Contracts>
for <Assembly Part 544001>

- ✓ Contract A3952990 (BCN)
- ☐ Contract A3952588 (BCN)
- ☐ Contract A3952564 (BCN)
- ✓ Insurance I934208
- ✓ Insurance I934209
- ☐ Insurance I934210
- ✓ Insurance I934214
- ☐ -----------------------
- ☐ Contract A394355 (BCN)
- ☐ Contract A394367 (BCN)
- ☐ Contract A394369 (BCN HH)
- ☐ Contract A394372 (BCN HH)
- ☐ Contract A394389 (BCN HH)
- ☐ Contract A394394 (BCN HH)

Figure 5. Integrated structural (vertical, semantic) and dynamic (horizontal, flow) model.

instantiation, inheritance, aggregation and others, depending on the required aspect, **leftwards** to the predecessors of the same and next sub-level along flow relations of data and activation and **rightwards** to the successors of the same and next sub-level along flow relations of data and activation.

Additional interfaces e.g. serve for browsing the hierarchy more easily in a tree-like view. The multi-relational hierarchy of the components is that of a Directed Acyclic Graph (DAG) for each relation type used, which requires aspect-oriented (mainly relation type-oriented) navigation.

The prototype has been implemented in C# using WinForms as generator target. An implementation for WPF is currently created in addition. The screenshot also shows how missing information in the process executed is added interactively: "Applicable Contract" input is not filled by the predecessors. Clicking on the button for this data required to execute "Assembly Part 544001", a dialog opens that offers the user to interactively enter the contracts to be used with this order / process instance.

CONCLUSION AND OUTLOOK

This article has presented a concept for an interactive process model execution and modeling prototype for distributed process execution and derivation of interaction elements from the semantic structures in the model.

Further research will be carried out on model extension and specification to allow for runtime user interface generation also for complex types and extend the modeling capabilities in addition to a better integration with existing standards and concepts in an M2S System. This also includes user interface generator components and extended process modeling capabilities.

REFERENCES

1. Anderson, D.M. *Build-to-Order & Mass Customization,* Cambria CIM Press (2003).
2. Erl, T. *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall (2005).
3. Harrison, R., Colombo, A. W. Collaborative automation from rigid coupling toward dynamic reconfigurable production systems, *Proc. 16th IFAC World Congress 2005* (2006).
4. IBM et al. *WS-BPEL Extension for People (BPEL4People)*, Version 1.0, (June 2007).
5. IBM et al. *Web Services Human Task (WS-HumanTask)*, Version 1.0, (June 2007).
6. Jamshidi, M. System of systems engineering – New challenges for the 21st century, *IEEE Aerospace and Electronic Systems Magazine*, 23, 5 (2008), 4-19.
7. Guerrero García, J., Lemaigre, C., González Calleros, J.M., Vanderdonckt, J. Model-Driven Approach to Design User Interfaces for Workflow Information Systems, Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3160-3173
8. Kopp, O., Unger, T., Leymann, F. Nautilus Event-driven Process Chains Syntax, Semantics, and their mapping to BPEL, *Proc. GI EPK* (2006).
9. OMG *Business Process Modeling Notation Specification*, OMG Final Adopted Specification, dtc/06-02-01, (2006).
10. OMG *Unified Modeling Language (UML) Superstructure* Version 2.1.1. formal/07-02-05, (2007).
11. Pinheiro da Silva, P., Griffiths, T., Paton, N. Generating User Interface Code in a Model Based User Interface Development Environment, *Proc. Advanced Visual Interfaces 2000* (2000), 155-160.
12. Schlegel, T. Object-oriented interactive processes in decentralized production systems. *Proc. HCI International 2009* (2009).
13. Schlegel, T., Kirn, S. Interacting with the Supply Swarm Towards an Interactive and Visual Management of Supply Webs. *Proc. IEEE INDIN* (2009).

Automated Optimization of User Interfaces for Screens with Limited Resolution

Sevan Kavaldjian, David Raneburger, Roman Popp, Michael Leitner, Jürgen Falb, Hermann Kaindl
Institute of Computer Technology
Vienna University of Technology, Austria
{kavaldjian, raneburger, popp, leitner, falb, kaindl}@ict.tuwien.ac.at

ABSTRACT
More and more devices with different screen resolutions are used to run the same application. In order to reduce usability problems, user interfaces (UIs) specific to resolution are needed, but it is time consuming and costly to implement all the different UIs manually. Automated generation of UIs has the potential to reduce time and costs in case of many such devices. Still, model-driven generation of UIs may not be flexible enough to include optimization for various resolutions.

We extended the straight-forward approach to model-driven generation by including optimization according to maximum usage of available space for a given resolution, minimum amount of clicks, and minimum scrolling. For these optimizations, we also use automated layouting and calculate the space needs of the possible variants. In effect, our new approach generates UIs optimized for screens with limited resolution, in order to reduce related usability problems.

INTRODUCTION
Automated generation of UIs has certainly advanced in recent years, especially based on model-driven approaches. Still, such generated UIs pose many usability problems. We think that this is partly due to insufficient flexibility of the current generation approaches.

In particular, straight-forward model-driven generation only allows for matching a single transformation rule for each source pattern. We extend this approach by taking up means from rule-based programming, that have been around for a long time. We allow matching of several transformation rules for any source pattern, and we use so-called conflict resolution to determine which rule to apply (fire). Based on that, we implement a simple form of optimization in the context of model-driven UI generation.

It allows us to maximize the amount of information to be displayed on a screen with limited resolution, to minimize the number of navigation clicks, and to minimize scrolling. All this is important for reducing usability problems. Since more and more devices with different screen resolutions are used to run the same application, we can automatically optimize the generated UI for the given (limited) resolution.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

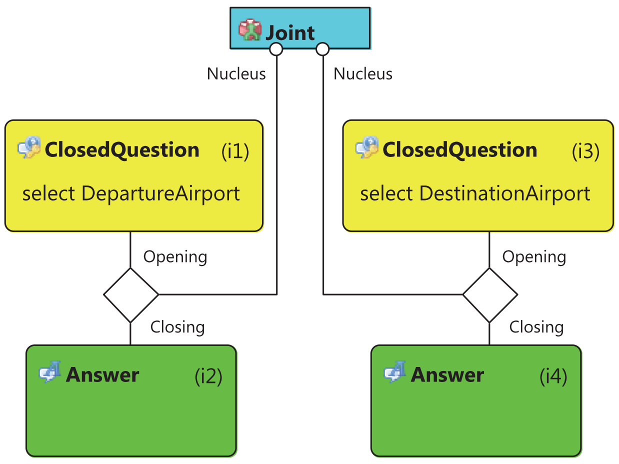
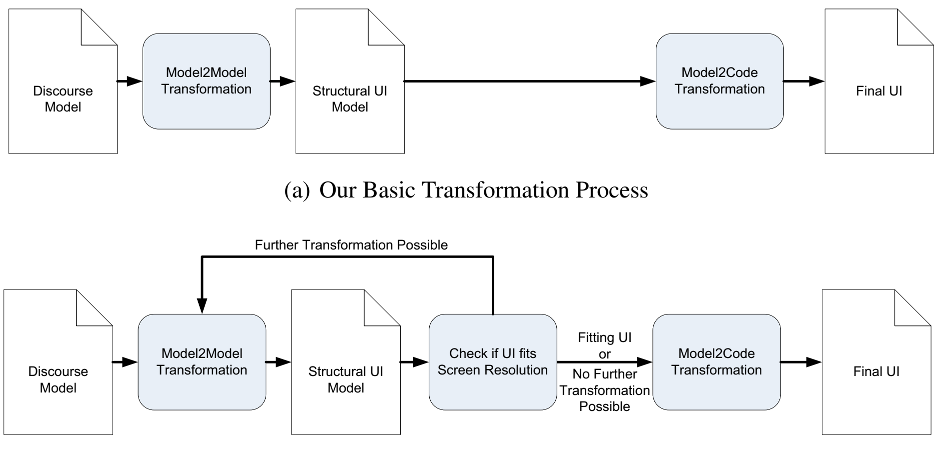


Figure 1. A discourse model excerpt

BACKGROUND
The input for our UI generation approach is a discourse model [2]. Such a discourse model serves as an interaction design on a high level of abstraction based on concepts of human language theories. A small excerpt of a larger discourse model for flight booking is shown in Figure 1. We use this discourse model as a running example throughout the remainder of this paper.

Our Discourse Models
The main ingredients of our discourse models are communicative acts derived from speech acts [7]. A communicative act is represented as a rounded rectangle and models an utterance of one of the communication partners. In our example, the application asks the customer *closed questions*, while the customer provides *answers* to the questions. In our example, the yellow (or light gray) communicative acts are uttered by the application and the green (or dark gray) ones are uttered by the customer.


Discourse Model
Model2Model Transformation
Structural UI Model
Model2Code Transformation
Final UI
Further Transformation Possible
Check if UI fits Screen Resolution
Fitting UI or No Further Transformation Possible

(a) Our Basic Transformation Process

(b) Our Extended Transformation Process

Figure 2. Our Transformation Process

Additionally, some communicative acts, like *Question* and *Answer* form a so-called adjacency pair which is represented by a diamond in our discourse models and defines the turn-taking and thus the sequence of utterances.

Adjacency pairs can be related with each other to build a tree structure. In our example, two Question–Answer pairs are related by a *Joint* relation. This *Joint* relation states that the Question–Answer pairs in both nucleus branches are of equal importance. Further, it does not imply a temporal order per se. For instance, both pieces of information can be presented in parallel if there is enough space on the screen. Otherwise they can be uttered in sequence.

Our Basic Transformation Process

We have developed a user interface generation process [5] that transforms such discourse models into WIMP-based graphical user interfaces (Windows, Icons, Menu and Pointers). Our basic user interface generation process is illustrated in Figure 2(a) and consists of two steps.

The first step transforms a discourse model into a Structural UI Model [4] by applying transformation rules to discourse model elements. The resulting Structural UI Model represents the user interface's widgets and their structure, but still abstracts from details of the final UI. We do not use a common UI description language (e.g. UsiXML[1]) because our runtime environment is based on the exchange of Communicative Acts. In our running example, the following transformation rules are applied to elements of the discourse model excerpt in Figure 1 for generating a model representing the structure of the final UI in Figure 3(a).

First, a *Joint Rule* gets applied that matches the Joint relation and adds a panel to the Structural UI Model. This panel acts as a container for the Radio Button lists in Figure 3(a), which correspond to the two nucleus branches of the Joint relation.

Second, a *Closed Question Rule* gets applied twice that matches each of the two Question–Answer adjacency pairs. For each adjacency pair a panel containing a label for a heading, a list of radio buttons together with item labels, and a submit button on the bottom is added to the Structural UI Model.

[1]http://www.usixml.org

In the second step this Structural UI Model is used to generate source code for a particular target platform, e.g., Java Swing in our running example.

OPTIMIZED RENDERING FOR LIMITED RESOLUTIONS

The problem tackled by our extended approach is to fit a given amount of information optimally (in the following sense) into screens with limited resolution.

Optimization Objectives and Approach

We assume that the following optimization objectives improve the usability of the generated user interfaces:

- maximum use of the available space for the given resolution,
- minimum amount of navigation clicks, and
- minimum scrolling (except list widgets).

Whenever the given information to be displayed does not fit into a single screen with default widgets, we try to display it with widgets that use less space. If it still does not fit into a single screen, then we split its display to two or more screens. Splitting increases the number of navigation clicks but it minimizes scrolling. We exclude list widgets from this last optimization objective because the number of list entries can vary extremely at runtime and determines whether the list is scrollable or not. This information is not known during our rendering process.

Our Extended Transformation Process

Our basic transformation process looks like straight-forward model-driven generation that only allows for matching a single transformation rule for each source pattern. We are not aware of any optimization strategy in such a context. Therefore, we extend the straight-forward approach by allowing that several transformation rules may match for each source pattern, and by applying so-called conflict resolution to select which rule to apply (fire) in the next model-to-model transformation. Our extended generation process shown in Figure 2(b) illustrates the resulting possibility of trying out several rules for optimization purposes. In this approach, the rules need not to be specifically designed for a particular screen resolution. It is rather the way the rules are applied that achieves the given optimization objectives.

In order to implement such an optimization, the conflict resolution mechanism needs to select the rules in a certain defined order. For achieving the optimization objectives given above, this selection order is according to the space that the widgets the rule creates occupy in the final UI. Therefore, all rules matching the same discourse element for transformation have to be ranked by the designer according to this space need.

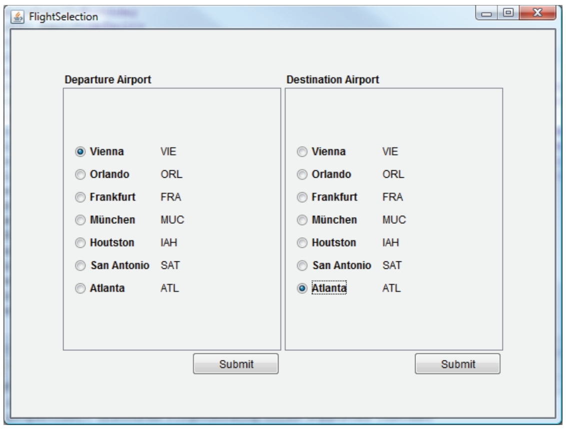
Each target device we render for has an abstract device specification that contains all style data used by the transformation rules. These data specify default sizes for all input and output widgets on the target device that can be overwritten in a transformation rule. They are used to set the size for each final UI element and allow us to calculate the exact size of each container (e.g., panel). For example, we set the size of the list widget explicitly. This makes it independent from the number of entries. If the list widget is not able to display all entries, it becomes scrollable.

After the size calculation we try to layout each generated screen to fit into the given resolution. However, we modify only the arrangement of the widgets that has not been fixed explicitly in a transformation rule. Therefore, we do not change the layout specified by the *Closed Question Rule* (i.e., the layout of the heading label, the radio button list and the submit button in Figure 3(a)). In this example, we modify the position of the complete radio button lists in the panel created by the Joint relation, since the *Joint Rule* does not contain any layout information.

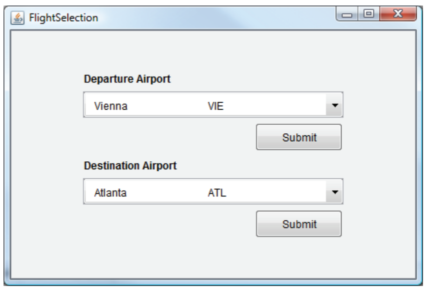
Now let us explain how to apply this approach to automatically generate user interfaces for three target resolutions. As input we use the discourse model excerpt shown in Figure 1.

Our first GUI is rendered for the resolution 640×480. The first cycle of the model-to-model transformation uses the highest ranked rules (i.e., the ones with the highest space need) for each discourse element. These are the same rules that have been applied in our basic transformation process. After the first transformation cycle we calculate the size for each panel in the corresponding Structural UI model. We can place them next to each other without exceeding the screen resolution. So, this is a fitting UI and we trigger the model-to-code transformation. The result is shown in Figure 3(a).

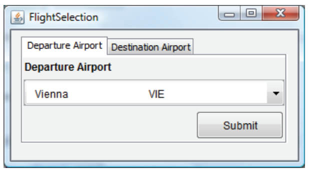
Next we generate a UI for the resolution 480×320. This time, the UI resulting from application of the highest ranked



(a) 640×480



(b) 480×320



(c) 320×180

Figure 3. Generated User Interfaces

rules does not fit. As long as a lower ranked rule can be applied, we initiate another generation cycle. First, the *Small Closed Question Rule* is used in our example. This rule matches the same source element (Question–Answer adjacency pair) as the *Closed Question Rule* but it creates a UI structure which occupies less space on the screen. A combo box element presents the content of the Closed Question communicative act to the user and a submit button is generated to confirm the selection of the user. The user interface shown in Figure 3(b) is the result of two more cycles, because in each cycle only one lower ranked rule is applied. The resulting UI fits and still presents the same information, but using widgets with less space needs (combo boxes instead of radio buttons). However, the list widgets do not fit next to each other and the layouter arranges them vertically.

In a third run, we generate a user interface for the resolution 320×180. Even after all rules for widget selection have been tried out, the generated GUI still does not fit the given resolution. Therefore, we start using rules that split the screen in order to increase the number of navigation clicks before we make use of scrolling. In our example, this means that in the next cycle the *Small Joint Rule* is applied instead of the *Joint Rule*. The *Small Joint Rule* matches the same source element (Joint relation) but creates a different UI structure (a tabbed pane element instead of a panel). Figure 3(c) shows the outcome for the resolution 320×180. The *Small Joint Rule* and the *Small Closed Question Rule* have been applied and a fitting UI has been generated after a third cycle of rule application. This time no layout modifications are necessary because each tab contains only one panel.

The worst case in our extended generation process occurs if and when no more rules are available and the generated screen still does not fit the given screen resolution. In this case, we stop the optimization loop and rely on scrolling.

RELATED WORK

A transformation system that fits web pages automated and on-the-fly to screens of small devices is presented in [8]. The transformations are performed in order to minimize navigation and scrolling like in our approach. In contrast, however, this process alters an already existing UI.

Declarative user interface specifications are used as input for multi-target UI generation in [3]. The user interface adaption is treated as an optimization problem based on a user- and device-specific cost function. Compared to such user interface specifications, our interaction models are on a higher level of abstraction.

The model-driven approach for engineering multi-target UIs presented in [1] supports switching between predefined presentations during runtime. Our approach, in contrast, is intended to automatically generate presentations for different resolutions from a single discourse model.

An advanced approach for generating multi-device UIs is based on task models [6]. Such a Task Model specifies the temporal relations among tasks and has to be adapted according to the screen space available on the target device. Therefore, any optimization and screen splitting has to be done explicitly during the creation of the Task Model.

We are not aware of any other approach that performs optimization in the course of model transformations. Neither are we aware of any model-driven GUI transformation process that takes the resolution for transformation rule selection into account.

CONCLUSION

Our new and extended approach introduces an optimization technique into model-driven generation of UIs to reduce usability problems. However, we only deal with relatively simple usability aspects (minimum amount of clicks and minimum scrolling). We do not (yet) optimize layout according to, e.g., aesthetic criteria. Therefore, our optimization approach as presented above is not suitable for large screens with high resolution.

Overall we introduce a UI generation process that allows the same rule set to be used for generating UIs for devices with different resolutions. Through the automatic calculation of space need, it may even have an advantage in this respect as compared to a human interface designer. We implemented a simple optimization approach that allows us to optimize generated UIs for devices with limited resolution in such a way as to utilize the given space and to minimize navigation and scrolling. This should pave the way to optimized multidevice UI generation.

ACKNOWLEDGMENTS

This research has been carried out in the CommRob project (`http://www.commrob.eu`), partially funded by the EU (contract number IST-045441 under the 6th framework programme).

REFERENCES

1. B. Collignon, J. Vanderdonckt, and G. Calvary. Model-driven engineering of multi-target plastic user interfaces. In *Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems (ICAS 2008)*, pages 7–14, Washington, DC, USA, 2008. IEEE Computer Society.

2. J. Falb, H. Kaindl, H. Horacek, C. Bogdan, R. Popp, and E. Arnautovic. A discourse model for interaction design based on theories of human communication. In *Extended Abstracts on Human Factors in Computing Systems (CHI '06)*, pages 754–759, New York, NY, USA, 2006. ACM Press.

3. K. Gajos and D. S. Weld. SUPPLE: Automatically generating user interfaces. In *Proceedings of the 9th International Conference on Intelligent User Interface (IUI '04)*, pages 93–100, New York, NY, USA, 2004. ACM Press.

4. S. Kavaldjian, C. Bogdan, J. Falb, and H. Kaindl. Transforming discourse models to structural user interface models. In *Models in Software Engineering, LNCS 5002*, volume 5002/2008, pages 77–88. Springer, Berlin / Heidelberg, 2008.

5. S. Kavaldjian, J. Falb, and H. Kaindl. Generating content presentation according to purpose. In *Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC2009)*, San Antonio, TX, USA, Oct. 2009.

6. F. Paternò, C. Santoro, and L. D. Spano. Model-based design of multi-device interactive applications based on web services. In *INTERACT (1)*, pages 892–905, 2009.

7. J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, England, 1969.

8. X. Xiao, Q. Luo, D. Hong, H. Fu, X. Xie, and W.-Y. Ma. Browsing on small displays by transforming web pages into hierarchically structured subpages. *ACM Transactions on the Web*, 3(1):1–36, 2009.

Evaluation of user interface adaptation strategies in the process of model-driven user interface development

Kai Breiner, Volkmar Gauckler
University of Kaiserslautern
Software Engineering Research Group
Gottlieb-Daimler Str.
67663, Kaiserslautern, Germany
breiner@cs.uni-kl.de,
volkmar.gauckler@gmx.net

Marc Seissler
University of Kaiserslautern
Center for Human-Machine-Interaction
Gottlieb-Daimler Str.
67663, Kaiserslautern, Germany
Marc.Seissler@mv.uni-kl.de

Gerrit Meixner
German Research Center for Artificial Intelligence (DFKI)
Trippstadter Str. 122
67663, Kaiserslautern, Germany
Gerrit.Meixner@dfki.de

ABSTRACT
In this paper, we describe the evaluation of our prototype Universal Control Device (UCD), which enables the control of various devices in modern dynamic production environments, while being able to adapt itself to the current configuration of the environment. While it is hard to apply traditional user interface design heuristics in recent paradigms – such as Ambient Intelligence – there is a demand for suitable compensation strategies addressing usage errors, which can be met by applying an adequate adaptation strategy. In a pilot study, we gained experience regarding differences in the performance of selected adaptation strategies in the case of our demonstrator.

Author Keywords
Universal Control Device, Adaptation Strategies, SmartFactory.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): Evaluation/methodologie, Prototyping, User-centered design.

INTRODUCTION

The ongoing technological development of microelectronics and communication technology is leading to more pervasive communication between single devices or entire pervasive networks of intelligent devices (smart phone, PDA, netbook, etc.). Especially industrial devices and applications can take advantage of modern smart technologies, e.g., based on ad-hoc networks, dynamic system collaboration, and context-adaptive human-machine interaction systems. The vision of Mark Weiser [1] concerning ubiquitous computing – also in production environments – is becoming a reality [10].

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

In today's production environments, technical devices often stem from multiple vendors using heterogeneous user interfaces that differ in terms of complexity, look&feel, and interaction styles. Such highly complex and networked technical devices or systems can provide any information at any time and in any place. This advantage can turn out to be a disadvantage when information is not presented properly according to the users' needs. This leads to problems, especially concerning the usability of the user interface. The level of acceptance of a user interface largely depends on its ease and convenience of use. A user can work with a technical device more efficiently if the user interface is tailored to the users' needs, on the one hand, and to their abilities on the other hand. Therefore, providing information in a context- and location-sensitive manner (depending on user, situation, machine, environmental conditions, etc.) has to be ensured.

Hence, in the following we will give a short introduction to the SmartFactory^{KL}, which serves as a demonstration environment for future intelligent production environments, and a Universal Control Device (UCD), which provides holistic control to various devices in such environments. Further, we give a brief introduction to user interface adaptation, usage errors, as well as to their compensation. After presenting our idea of how to approach compensation by using adaptation strategies, we describe the set-up of the corresponding controlled experiment and the preliminary results of the pilot study conducted. We conclude with the interpretation of how the results contribute towards our hypothesis.

SmartFactory^{KL}

Besides these aspects, modern production environments are characterized by a modular layout. Entire modules can be replaced or reorganized. Furthermore, these environments are able to react to errors occurring in the production process (e.g., device malfunction) and to dynamically reorganize parts of the process in order to ensure the production process. Thus, this also affects the user who interacts with the individual devices – the user's workflow will change, or devices will not be available anymore.

Serving as a demonstration environment, the SmartFactory^{KL} in Kaiserslautern, Germany is able to simulate such a process. In previous work, we developed a UCD, which is able to provide access to various devices of the SmartFactory^{KL} [3][4].

Universal Control Device (UCD)

As a result of a model-driven process, the user interface of the UCD is being generated at run-time [2]. Starting with a topological description of the environment, enriched with user tasks on the single devices and information about how to address the single devices, this information is sufficient for generating a functional user interface [2][4]. In order to remain functional to the user, this user interface has to correspond to the current configuration of the environment and is additionally restricted to the functionality as specified in the underlying model. During field studies, faced by the need to adapt the user interface, we encountered the demand for a systematic strategy that would support the user as much as possible [3]. This was the trigger for a study on adaptation, which we will describe in the following.

ADAPTATION

After giving a brief overview of different types of adaptation – their properties as well as their impact on the users' workflow – we will elaborate types of usage errors resulting from static user interfaces and how adaptive user interfaces contribute to the compensation of such errors.

Static versus Dynamic User Interfaces

On the one hand, one important usability quality attribute is *memorizability*. The ease to remember helps users to speed up the process of interacting with the user interface [5][6]. Since humans have a very visual memory, the way a certain user interface is structured is essential for finding items faster. After a while, users form a coherent model of the user interface and are able to recall how to execute their workflow. If the system (and therefore the user interface) changes frequently, the user will not be able to form such a model [7][8]. On the other hand, there are user interface heuristics demanding that the user interface matches the real world [6]. In order to provide control over devices in dynamic environments – such as the SmartFactory^{KL} – it is extremely vital to match the user interface to the real world in order to remain functional.

These simple rules about how to develop a user interface appear to be contradictory for future information systems in our application domain. Hence, there is a need for an adaptation strategy that does not violate usability quality attributes leading to usage errors.

Usage Errors

In general, for each kind of usage error, there is a basic cause (see Figure 1). Using the right compensation technique – such as adaptation of the user interface – the users can be actively supported in preventing usage errors. Basically, there are two kinds of operating errors that may lead to a failure of the system.

In the first case, there are *slips*. Here, the user had the correct intention but executed the task in the wrong way. Most often this is caused by poor physical skills, or by the user interface just being just inadequate for use (e.g., buttons too small). In the second case – which is more interesting in our example – there are *errors*. Errors are characterized by the user having the wrong intention while executing the task. This is caused by a misunderstanding of the user interface (e.g., if the user interface is offering control over devices that are not available physically).

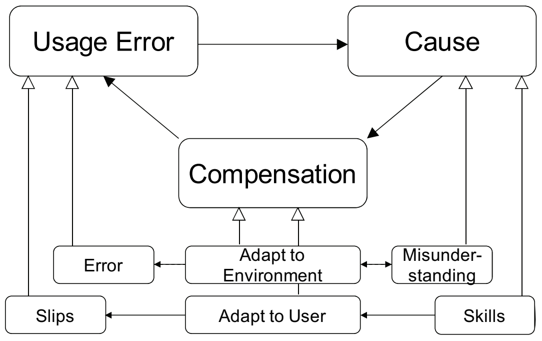


Figure 1. Relationship between usage errors, their causes, and compensation.

Compensation

Depending on the type of usage error, there are different ways to compensate in order to minimize the effect. *Slips*, which are caused by poor skills of the user, can be prevented by adapting the user interface to the user. In case of our application domain – intelligent production environments – we are dealing with predefined user roles and user groups and are therefore able to tailor the user interface to the needs of these user groups.

In dynamic environments, *errors* can be prevented by adhering to design heuristics as mentioned earlier. The system has to represent the current configuration of the user interface, while supporting the development of a mental model. Hence, a method of adaptation needs to be chosen that contributes to these heuristics.

Types of Adaptation

There are different methods of how an adaptation of the user interface can be executed. These methods differ in their way of execution as well as in their degree of intrusion into the users' workflow. In case of the UCD, a simple adaptation scenario would be the appearance or disappearance of devices in the device selection list. In the following, we will refer to this example while giving details about the individual methods.

The first method – which was implemented initially and led to the investigation described in this paper – is *ad-hoc* adaptation. Here, devices are added or removed from a device list according to the physical status of the respective devices. Unless a regular user permanently observes the device list, he or she will not notice any change. Furthermore, this will be distracting for the user, because the structure of the user interface changes without notification.

Providing more information about the system state leads to the second adaptation method – *notification*. Now, the user interface provides information about the change and therefore supports the user in adapting his or her mental model. In case of the device list example, we implemented this method by applying the so-called instant-messenger metaphor [9]. This means that new or defective devices will be emphasized graphically, which provides information for the user to understand the current state of the system. However, the user may just overlook the notifications if he or she is distracted, and then the system cannot provide further support.

Thus, we implemented a third adaptation strategy, which aims at *confirmation* by the user. Here, the user has to actively confirm the change to the user interface. Referring to the device list of the example above, this strategy was implemented in terms of a dialog box asking for confirmation from the user that he or she has actually noticed the adaptation of the system. Thus, the system can be almost sure that the user is aware of the adaptation and is supported in the presumably most adequate way. A negative side-effect of this strategy is that the confirmation dialog may distract the user during his/her regular workflow.

Compensation

Each of the adaptation strategies differs as to how much it contributes to the compensation of possible usage errors.

Ad-hoc adaptation ensures consistency between the controllable environment and the corresponding user interface in order to prevent usage errors with respect to outdated user interfaces. But this approach provides no active support to the user at all.

Besides consistency with the current configuration of the environment, notification provides limited feedback to the user (e.g., by visually emphasizing new devices). Due to the fact that this communication with the user is unidirectional, such notifications can be easily overlooked by the user.

Confirmation provides all the functionality of the first two approaches on the one hand and demands confirmation by the user on the other hand. Thus, the user interface is aware of the fact that the user has recognized the new configuration and can therefore proceed with the regular functionality.

Hypothesis

According to the diverse properties of these strategies, we wanted to investigate which is the most adequate one in the case of our model-driven approach and therefore we formulated hypotheses that we are going to verify or falsify initially in a preliminary study described below. The hypotheses are tailored to the specific set-up of the controlled experiment.

H1. Effectiveness – completion rate

We assume that, on average, at least 85,6% of the test tasks can be completed without help.

Explanation: This hypothesis assumes that test persons can deal with the user interfaces and have understood their tasks and therefore can complete at least 6 out of the 7 tasks.

H2. Effectiveness – assistance

100% of the given tasks can be completed (with help – if needed).

Explanation: The user interface ensures consistency between environment and visualization independent of the adaptation strategy. Therefore, it should be possible to complete each task.

H3. Efficiency – strategy performance

Confirmation outperforms notification, which outperforms ad-hoc adaptation.

Explanation: On the basis of the different attributes elaborated earlier, we assume this ranking according to the performance of the different strategies.

EVALUATION – PILOT STUDY

To evaluate these hypotheses, we decided to conduct a controlled experiment. We implemented three instances of our model-driven process [2][3][4], including the corresponding adaptation strategies. The test persons had to complete seven tasks on the user interface, while the simulated production environment always reconfigured between task 2 and task 3. The idea was that the adaptation should only affect task 4, which had to be executed in a different way (as a result of the adaptation) than indicated at first. The other tasks served as an indication that the test persons perform in a similar way. We conducted a *pilot study* with several computer science students. All six test persons were between 21 and 34 years old.

After being asked for personal statistical information, the test persons got a thorough introduction to the production industry domain. They were provided with a detailed explanation of the simulated production environment as well as of our Universal Control Device. After execution of task 2, we initiated the reconfiguration of the simulation and, depending on the strategy used (of which the test persons were not aware) the user interface adapted itself, notified the user, or demanded confirmation. The strategies were distributed equally between the tests. Due to the adaptation of the environment, the user should not be able to complete task 4 in the way the task description called for. One of three redundant pumps was removed from the system. Here, the user should conclude that (as displayed in the documentation of the simulated environment) there are various ways to complete this task and ask the moderator if this is possible. The test was completed with a survey about the subjective properties of the user interface.

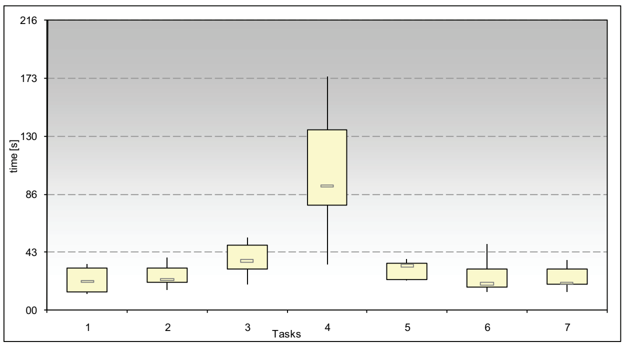


Figure 2. The performance of the 7 tasks (median, max/min and quantile).

Results

Figure 2 shows the results of the performance of the single tasks. As we assumed before conducting the experiment, all tasks (the control tasks) except task 4 were performed similarly. Referring to the standard deviation, which is an average of 16 seconds in case of the control tasks and 38 seconds in case of task 4, the difference in performance can be easily identified. Thus, we conclude that the results of this preliminary experiment are representative.

When executing task 4, the test persons recognized the redundancy of the three pumps and asked if they could use another pump, which was intended.

Discussion of the Results

All test persons were able to complete all tasks (with help in case of task 4), which confirms hypothesis H1. Since all of them needed help when executing task 4 and only during task 4, the estimated 85,6% of hypothesis H2 was almost exactly confirmed.

For ah-hoc adaptation, the execution of task 4 took an average of 117 seconds, for notification 125 seconds, and for confirmation 89 seconds. Hence, hypothesis H3 cannot be entirely confirmed, as ad-hoc adaptation outperformed notification, but still confirmation outperformed both of the other strategies.

Threats to validity

Because all the test persons were only computer science students, this may have had an effect on the result. But on the other hand, those students were already familiar with our previous work (without knowing the content of the experiment), which could also be a good simulation of domain knowledge. Nevertheless, the experiment needs to be conducted (and will be) using production industry domain experts. The most important threat to validity to mention is the fact that this was only the pilot study for the described experiment. This means that the sample was too small and therefore has no real statistical evidence, but it serves as a first indication as to whether an investigation according our idea would make sense.

CONCLUSION

When dealing with user interfaces in highly dynamic environments, such as intelligent production environments, there are special requirements. According to a shown dissonance in user interface design, when being applied in these environments, we have shown there is a special need for compensating usage errors. This can be achieved by systematically integrating adaptation strategies into the model-driven development process. Since multiple strategies exist, which provide different user experience, the performance with respect to our demonstrator was evaluated in a pilot study of a controlled experiment. First results show that there are differences in the performance and therefore some of our hypotheses could be verified.

ACKNOWLEDGMENTS

Our work as well as the GaBi project is funded in part by the German Research Foundation (DFG).

REFERENCES

1. Weiser, M. The computer for the 21st century. *Scientific American*, 265, 3 (1991), 94-104.
2. Breiner, K., Görlich, D., Maschino, O., and Meixner, G. Towards automatically interfacing application services integrated in an automated model based user interface generation process, *Proc. of the 4th International Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI)*, CEUR Workshop Proceedings Vol-439 (2009).
3. Breiner, K., Görlich, D., Maschino, O., Meixner, G., and Zühlke, D. Run-Time Adaptation of a Universal User Interface for Ambient Intelligent Production Environments. *Proc. of the 13th International Conference on Human-Computer Interaction (HCII-09), LNCS Vol. 5613* (2009), 663-672.
4. Breiner, K., Görlich, D., Maschino, O., and Meixner, G. Automatische Generierung voll funktionsfähiger mobiler Bediensoftware aus Benutzungs- und Funktionsmodellen. *Proc. of Informatik, LNI Vol. P-154* (2009), 2210-2215.
5. Gould, J. D. and Lewis, C. 1985. Designing for usability: key principles and what designers think. *Communications of the ACM,* 28, 3 (1985), 300-311.
6. Nielsen, J. Ten Usability Heuristics. www.useit.com/papers/heuristic/heuristic_list.html.
7. Norcio, A. F., and Stanley, J. Adaptive Human - Computer Interfaces. *NRL Report 9148*, Naval Research Laboratory (1988).
8. Mitchell, J. and Shneiderman, B. Dynamic versus static menus: an exploratory comparison. *SIGCHI Bulletin,* 20, 4 (1989), 33-37.
9. Lee, L. and Johnson T. URCousin: Universal Remote Control User Interface. *Proc. of the Human Interface Technologies Conference* (2006).
10. Zuehlke, D.: SmartFactory – From Vision to Reality in Factory Technologies. Proc. of the 17th International Federation of Automatic Control (IFAC) World Congress (2008), 82-89.

Model-Based Usability Evaluation and Analysis of Interactive Techniques

Jeff Ladry, Philippe Palanque, Eric Barboni & David Navarre
IRIT - University Paul Sabatier
118, route de Narbonne, 31062 Toulouse Cedex 9, France
{ladry, palanque, barboni, navarre}@irit.fr

ABSTRACT

This position paper presents a model based approach supporting development of advanced user interfaces for the design, simulation, tuning and the assessment of interaction techniques. It is based on a double concept: the introduction of additional information in models to allow designer to tune easily the interaction technique and the use of simulation and logging facilities to assess perform performance evaluation of the models. It proposes an alternative to user testing which is very difficult to setup and interpret when advanced interaction techniques are concerned.

Author Keywords

Model-Based approaches, formal description techniques, performance evaluation, multimodal interfaces, interactive software engineering, tuning.

INTRODUCTION

Using models for the design of computer systems provide a description of the system abstracted away from its implementation. Nowadays, such approaches are prominent in the area of software engineering via the Model Driven Engineering [9] field that emerged from the UML standard [11]. Indeed, as they provide a more abstract description of the system than the implementation code they also provide a unique opportunity for various stakeholders (users, developers…) to comment and propose modifications.

In The HCI community many researchers have described user interface elements by means of models. The interested reader can find a complete state of art of model-based user interface in [10] where the different modeling techniques are categorized by criteria such as: *Language*, *Interaction Coverage*, *Scalability, Tool support and Expressiveness.*

Beyond this descriptive aspect, models can also be used to support the evaluation of the user interface for properties (such as liveness and safety) or even for usability including: Model Based Usability remote evaluation as in RemUsine [13], EvaHelper [5], or in ReModEl [5]. Similar work can also be found for the Web domain as in AWUSA [14]. Usability evaluation can also be found for more generic purpose as in MeMo&MASP [6] or in [1] with MDA (Model-driven architecture)-compliant methods to improve software usability through model transformations.

Among these contributions, many have shown that HCI concerns must be integrated within the development process in order to design and develop usable systems. This is known as the "too little too late" problem detailed in [8] claiming that usability must be considered in the early stages of the development process or it will be only partially addressed.

Next section presents a model-based approach proposing an emphasis on *evolvability* and *modifiability* of models to be able to take into account usability concerns. The basic idea is to prepare models for modification at design time in order to be able to adjust them according to usability evaluation results.

THE APPROACH

This position paper proposes a design process for the design, simulation, tuning and assessment of interaction techniques.

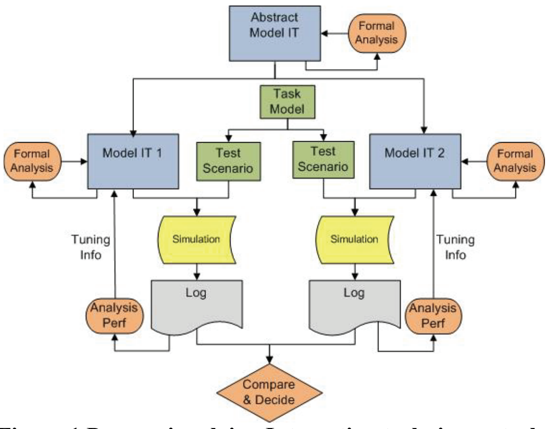


Figure 1 Process involving Interaction techniques, tasks models and Analysis +Log

Figure 1 presents the process of this approach exemplified for the comparison of two interaction techniques.

Modeling Interactions techniques

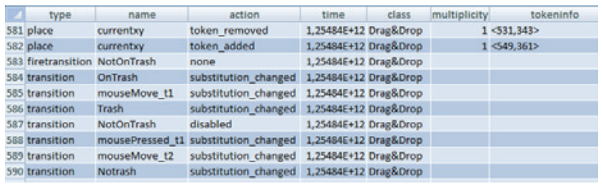
In the beginning of the process an abstract model (Abstract Model IT) is constructed using the ICO formalism [10] to accomplish the task represented in the Task model in CTT. From abstract model, multiple detailed models can be produced. These models refine the abstract model according

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

for instance to properties we want the technique to fulfill or according to the different modalities that have to be used. From the task model in CTT, a test scenario is extracted for each ICO model that the interaction technique must be able to perform. The detailed model is then simulated in Petshop according to the test scenario.

Simulation and Logging

During this simulation a log file is produced containing all the information about the evolution of the model.



	type	name	action	time	class	multiplicity	tokeninfo
581	place	currentxy	token_removed	1,25484E+12	Drag&Drop	1	<531,343>
582	place	currentxy	token_added	1,25484E+12	Drag&Drop	1	<549,361>
583	firetransition	NotOnTrash	none	1,25484E+12	Drag&Drop		
584	transition	OnTrash	substitution_changed	1,25484E+12	Drag&Drop		
585	transition	mouseMove_t1	substitution_changed	1,25484E+12	Drag&Drop		
586	transition	Trash	substitution_changed	1,25484E+12	Drag&Drop		
587	transition	NotOnTrash	disabled	1,25484E+12	Drag&Drop		
588	transition	mousePressed_t1	substitution_changed	1,25484E+12	Drag&Drop		
589	transition	mouseMove_t2	substitution_changed	1,25484E+12	Drag&Drop		
590	transition	Notrash	substitution_changed	1,25484E+12	Drag&Drop		

Figure 2 Excerpt of a Model-based log

The model-based log (presented in Figure 2) records all the change which occurs in the model during the simulation including firing of each transition, the removal and addition of a token in every place. This data is then exploited to assess the performance of each interaction technique in absolute value as well as their relative performance. If the performance does not fit the expectations, the log data can be used to modify or tune the model that will be simulated again. Such modification or tuning is made much easier as the information in the log is already related to the structure of the model.

Formal Analysis

Due to the Petri nets-based roots of the ICO formalism, we are able to use Petri nets properties analysis techniques such as place and transition invariant. These invariant allow us to prove properties such as liveness of a transition in a model for example. In the case of an interaction technique this would make it possible to assess that the transition handling mouse move events is always available (it is always possible to produce such events by moving the device.) According to the result of the analysis process, it can be decided to modify the model.

Tuning of models (Evolvability and Modifiability)

According to the performance evaluation obtained with the log analysis, some fine tuning can be applied in the model to increase the performance of the interaction technique. This fine tuning can either be done during or before the simulation as in PetShop models can be modified while executed.

Figure 3 presents the drag and drop interaction technique modeled using the ICO formalism. In the initial state, the interaction technique is Idle (there is a token in place *Idle*), the position x,y of the mouse cursor is stored as a token in place *Currentxy*, the reference to the graphical object frame (where the cursor is moving) is stored as a token in place *Frame* and the reference to the object trash is stored as a token in place *Trash*. From that initial state two transitions are available (represented in darker grey in the model: mouseMove_t1 and mousePressed_t1. Transition mouseMove-t1 is fired when the corresponding event is triggered by a user action on the input device.

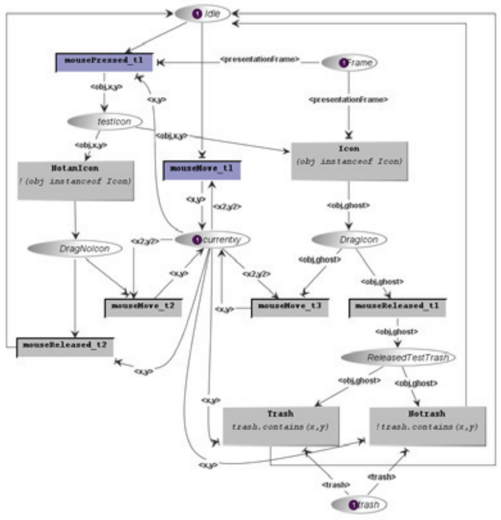


Figure 3 ICO model of basic Drag & Drop interaction technique

When this occurs, the value of the token stored in place *Currentxy* is changed to contain the new position of the cursor. Transition mousePressed_t1 is triggered by a user action on the button of the mouse. When this occurs, the model tests (represented by transitions NotonIcon and Icon) if the cursor is currently on an icon or not. If the cursor is on an icon then the model will process mouse move events (transition mouseMove_t3 which updates the cursor position) and mouse released events. When a mouse released event is received the model tests if the position of the cursor is on the icon of the trash (transition Trash) or not (transition Notrash). If yes, the file is deleted (this is modeled in the code of transition Trash and not represented here due to space constraints).

This model is not easily modified to integrate tunings that are currently made on drag and drop techniques such as acceleration and deceleration of the icon (according to the proximity of the target icon or according to the rapidity of the movement). However, it represents precisely and without any ambiguity the desired behavior of the initial interaction technique. According to our experience with interaction technique modeling, we know that fine tuning of the interaction technique is required.

Figure 4 presents and extended version of the model of Figure 3 adding possibility for tuning the interaction technique. Two new transitions have been added (in blue in Figure 4) allowing the possibility to check if the pointer is on the reactive object (here the trash) or not. With this information we can easily change the speed of the pointer when it is on the reactive object to "stick" it on the object for example. Such modification corresponds to changes in the motor space as introduced by [4].

To make it possible to tune this interaction, we have also added *Acceleration* and *Deceleration* places (circled in red in Figure 4) and linked them to MouseMove_t2 transition.

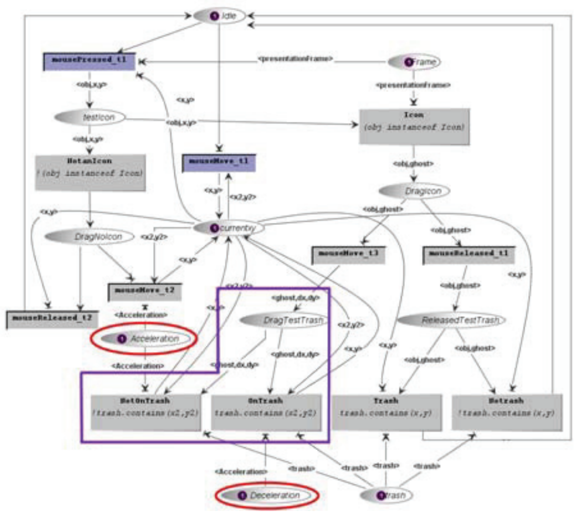


Figure 4 ICO model of tunable Drag & Drop interaction technique

The *Acceleration* place contained a value used for the acceleration of the mouse cursor when an object is dragged;

The principle of the approach is to run simulations of the models to identify possible limitations and propose modifications to be made in the models to improve the efficiency of the interaction technique. After tuning a new simulation is performed and the results are compared to the desired properties.

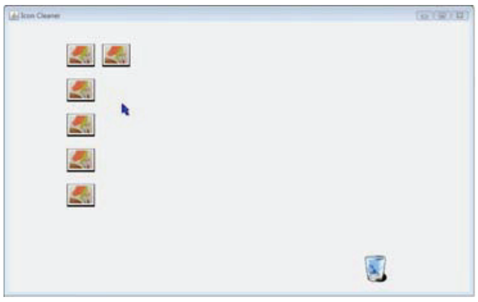


Figure 5 User Interface of the case study

Next section sketches how this approach can be instantiated on a case study, presenting the models, the transformations and the results of the performance evaluation.

CASE STUDY

The objective of the case study is to present the various phases of the approach on a simple but realistic application (see Figure 5). In the application a set of icons are presented in a window on a grid. The user's goal is to remove all the icons on the user interface by doing, iteratively in any order, the selection of an icon and the triggering of a deletion command the selection and deletion of icons. To support this goal two different interaction techniques have been modeled. Following the terminology of Figure 1, Model IT1 (called Drag & Drop) features an interaction technique of type Drag and Drop and behaved as described in the previous section. Model IT2 (called Speak & Click) features a multimodal interaction technique involving speech recognition (for the deletion command) and gesture (for icon selection). Systems and tasks models related to these two interaction techniques are presented in the next sections.

Modeling Interaction Technique 1

The behavior of IT 1 is presented in Figure 4. According to the more detailed description of the interaction technique, the abstract tasks to be refined as presented in Figure 6 in order to produced test scenarios (as presented in the design process of Figure 1). Selection is performed first by deciding the icon to be deleted then by moving the mouse cursor on the icon and by pressing the mouse button. Deletion is performed by moving the selected icon over the trash icon, verifying that trash icon is highlighted and releasing mouse button.

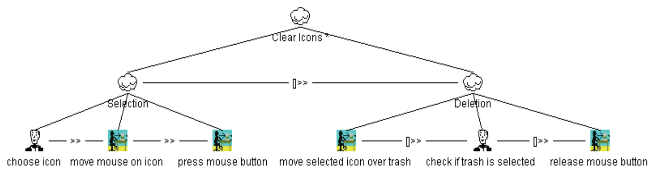


Figure 6 Task model refined to be conformant with Drag & Drop behavior

It is interesting to note that the temporal operator between tasks Deletion and Selection is *order independence*. The same imposed sequencing can be found in the ICO model where the model TI 1 (in Figure 4) imposes to start interaction with the selection (deletion is only available later on).

Modeling Interaction technique 2

Similarly to what has been done for Interaction technique 1, Interaction Technique 2 has been fully described using the ICO formalism and is presented on Figure 7.

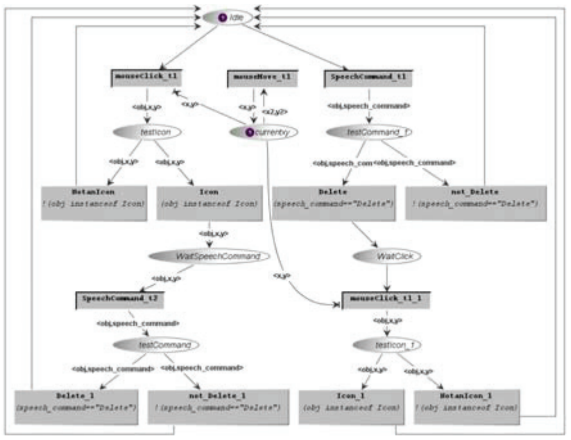


Figure 7 ICO model of system B (Speak & Click interaction technique)

This model allows users to either start by a speech command "*delete*" and then selecting an icon or selecting first an icon to be deleted and then uttering the word "*delete*". The abstract tasks model has to be refined similarly to the model in Figure 6.

Simulation

Simulation of the interaction technique models is done in the case tool Petshop. Further information about the case tool can be found in [4] and about the simulation in. We don't provide here more information about the simulation as it has been introduced in [2] and is beyond the scope of this position paper.

Logging

From log extracted from the simulation of IT1 we can produce information such as the total time for a Drag&Drop. We can also compute the number of time the move change from *OnTrash* to *notOntrash* before the releasing to represents the number of time the user has missed the trash and exited the target without releasing on the icon. All such information comes only from the places and transitions that can thus easily be seen on the model represented in Figure 4 and the relationship with the log as presented in Figure 2 is immediate.

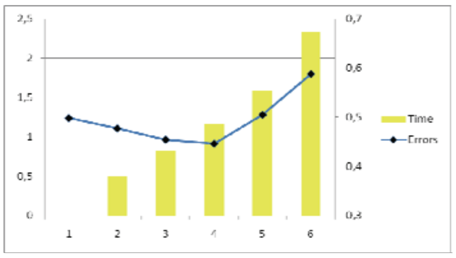


Figure 8 Result of Analysis of Log

After identifying where that information will be extracted from the model extract, we can simulate several time the model with different values and see if the total speed of the interaction technique and the number of errors to execute this task evolve. Such results can be gathered in a graph as presented in Figure 8. That graph shows that for an increase of acceleration of the mouse (Acceleration place in Figure 4), first the Drag&Drop is faster. But when the acceleration is 5, the errors are too important and the time to make the Drag&Drop increases and becomes worst than the standard interaction technique without acceleration

CONCLUSION

In this paper we have presented an approach to test and evaluate different interaction technique. This testing and evaluation is driven by models. With these models we can also tune finely different aspects of the interaction technique. This approach has exemplified on a small example where we show the interest of a model based usability evaluation. The results show well known results in HCI such as that acceleration improves efficiency of target selection to a certain extend. The objective of the approach is to apply it to novel and more sophisticated interaction techniques (possibly multimodal ones) which are much harder to assess especially through user testing.

REFERENCES

1. Abrahão S., Iborra E., and Vanderdonckt J. 2007. Usability Evaluation of User Interfaces Generated with a Model-Driven Architecture Tool. In Maturing Usability: Quality in Software, Interaction and Value, Series: Springer Human-Computer Interaction Series, Vol. 10, E. Law, E. Hvannberg, and G. Cockton (Eds.), 610p., 2007, ISSN: 978-1-84628-940-8, Springer.
2. Bastide, R., Navarre, D., and Palanque, P. 2002. A model-based tool for interactive prototyping of highly interactive applications. In CHI '02 Extended Abstracts on Human Factors in Computing Systems (Minneapolis, Minnesota, USA, April 20 - 25, 2002). CHI '02. ACM, New York, NY, pp. 516-517.
3. Balagtas-Fernandez F., Hußmann H., A Methodology and Framework to Simplify Usability Analysis of Mobile Applications In Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering (ASE 2009). Auckland New Zealand, November 2009, ISBN 1527-1366/09, pp. 520-524.
4. Blanch R., Guiard Y. and Beaudouin-Lafon M. Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation. In Proceedings of CHI 2004, pages 519-526, Vienna - Austria, April 2004.
5. Buchholz G, Engel J, Märtin C, Propp S. Model-based usability evaluation - evaluation of tool support. HCII 2007, Beijing, China, 2007. p. 1043-52. Springer-Verlag, Berlin, Germany, 0302-9743
6. Feuerstack, S., Blumendorf, M., Kern, M., Kruppa, M., Quade, M., Runge, M., and Albayrak, S. 2008. Automated Usability Evaluation during Model-Based Interactive System Development. In Proceedings of the 2nd Conference on Human-Centered Software Engineering and 7th international Workshop on Task Models and Diagrams (Pisa, Italy, September 25 - 26, 2008). P. Forbrig and F. Paternò, Eds. Lecture Notes In Computer Science, vol. 5247. Springer-Verlag, Berlin, Heidelberg, 134-141.
7. Kristoffersen, S. 2009. A Preliminary Experiment of Checking Usability Principles with Formal Methods. In Proceedings of the 2009 Second international Conferences on Advances in Computer-Human interactions (February 01 - 07, 2009). ACHI. IEEE Computer Society, Washington, DC, 261-270
8. Lim, K. Y. and Long, J. (1994). The Muse Method for Usability Engineering. Cambridge University Press.
9. Mellor S. and Balcer M. (2002). Executable UML: A Foundation for Model-Driven Architectures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
10. Navarre, D., Palanque, P., Ladry, J., and Barboni, E. 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. 16, 4 (Nov. 2009), 1-56.
11. Object Management Group (2003). Unified Modelling Language (UML) 2.0 Superstructure Specification, August 2003. Ptc/03-08-02, pp. 455-510
12. Paternò F., Mancini C., Meniconi S. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models, Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction, p.362-369, July 14-18, 1997
13. Paternò, F., Russino, A., Santoro, C. (2007) Remote evaluation of Mobile Applications, Task Models and Diagrams for User Interface Design 6th International Workshop, TAMODIA 2007, Toulouse, France, November 7-9, 2007, Lecture Notes in Computer Science , Vol. 4849, Winckler, Marco; Johnson, Hilary; Palanque, Philippe (Eds.) ISBN: 978-3-540-77221-7
14. T. Tiedtke, C. Märtin, N. Gerth. Awusa, A tool for automated website usability analysis. 9th Int. Workshop DSVIS, 2002.

A Saliency Model Predicts Fixations in Web Interfaces

Jeremiah D. Still
Missouri Western State University
Department of Psychology
jstill2@missouriwestern.edu

Christopher M. Masciocchi
Iowa State University
Department of Psychology
cmascioc@iastate.edu

ABSTRACT
User interfaces are visually rich and complex. Consequently, it is difficult for designers to predict which locations will be attended to first within a display. Designers currently depend on eye tracking data to determine fixated locations, which are naturally associated with the allocation of attention. A computational saliency model can make predictions about where individuals are likely to fixate. Thus, we propose that the saliency model may facilitate successful interface development during the iterative design process by providing information about an interface's stimulus-driven properties. To test its predictive power, the saliency model was used to render 50 web page screenshots; eye tracking data were gathered from participants on the same images. We found that the saliency model predicted fixated locations within web page interfaces. Thus, using computational models to determine regions high in visual saliency during web page development may be a cost effective alternative to eye tracking.

Author Keywords
Saliency, Interface Development, Design, Model

ACM Classification Keywords
H.1.2 User/Machine Systems; I.2.10 Vision and Scene Understanding

INTRODUCTION

Saliency, Search and Design

Some visual designs guide users to the locations of important information, while others mislead users. Visual saliency, inherent in a complex interface, cues users to certain spatial regions over others. If employed correctly by designers, salient cues may reduce information search times and facilitate task completion [cf. 18] by implicitly communicating to users where they ought to start their visual search [16]. In order to be considered salient, a feature must be visually unique relative to its surroundings. For example, text that is <u>underlined</u> amongst non-underlined text "pulls" the reader's attention to it. However, many interfaces, like web pages, are rich with visual media, such as text, pictures, logos and bullets, making the determination of salient features a complicated task. Given this complexity, designers are often left making best guesses about which spatial regions are salient within an interface. Previous research on visual search in web pages defines entry points as regions within a page where users typically begin their visual search. In this article, we will argue that these entry points are heavily influenced by visual saliency, that is, users will often begin searching web pages at the location of highest saliency. In related research examining cognitive processing these implicit and low level cues that guide a viewer's visual search are referred to as stimulus-driven properties – certain characteristics of the stimulus quickly "drive", or direct attention to certain locations over others. Currently, no consensus has been reached as to which visual characteristics, or stimulus-driven properties, make for effective entry points.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

Measuring Overt Attention through Eye Tracking

Given the over abundance of visual information in our environments and our working memory limitations, attention must be selective, only allowing a limited amount of information into consciousness, for our cognitive system to function properly [8]. It has been suggested that the programming of eye movements has a direct and natural relationship with visual attention in that attention is often directed to whichever item is fixated [10]. Only information that falls directly on the fovea during a fixation is encoded with high resolution and only a limited amount of this high resolution information is processed, while the rest falls into rapid decay [see 4]. Thus, it is critical that users fixate on relevant visual information or that content will not reach users' awareness.

It is no surprise then, that designers often monitor eye movements to evaluate a web page's saliency, or entry points. Eye tracking systems allow designers to test whether their web pages actually guide users' fixations to important locations. However, eye tracking has a number of recognized costs. Eye tracking systems are often expensive, not easily accessible, time consuming to employ and they gradually lose calibration [1, 2, 7, 15].

Stimulus and Goal Driven Searches

In this article we investigate the influence of stimulus-driven saliency on attention within the context of a web page. Stimulus-driven saliency guides attention quickly and without explicit intention, thus some might question its role during a purposeful search on a web page. There is ample evidence to suggest that goals do influence the guidance of attention. For example, web page eye tracking research has shown that changing the task (or goal) during a search, or seeking navigational or informational indicators, changes observers' fixation patterns [3]. Additional research has shown that, given enough time, expectations can cause a consistent pattern of fixations – F-shaped pattern or reading patterns (e.g., left-right/top-bottom) [14]. However, these goal-driven effects interact with stimulus-driven effects, making the stimulus-driven influences more difficult to examine [cf. 11]. Also, it is often the case that only a few seconds are spent on a web page (even with a goal in mind) making the understanding of stimulus-driven processing, which is believed to influence attention very rapidly, critical. For instance, when searching for information observers often only skim through approximately 18 words, and spend 4 to 9 seconds, per web page [2, 12]. One way to investigate the pure influence of stimulus-driven guidance is to use a computational saliency model designed to make predictions about what properties or features of a web page attention ought to select within complex media, or scenes.

Predicting Fixations through a Saliency Model

Visually salient items often draw observers' attention. To better understand the influences of saliency, or stimulus-driven selection, on attention, Koch and Ullman (1985) developed a model to compute an image's visual saliency without any semantic input (i.e., meaning of objects). Their model is based on the assumption that eye movement programming is driven by local image contrast leading to logical serial searches through complex spatial environments. These serial searches are guided by low level primitives extracted from a scene. The saliency model was developed under the pretense that low level visual features (i.e., color, light intensity, orientation) are processed pre-attentively in humans and, in turn, rapidly influence overt attention. Thus, the underlying assumption is that visual saliency is used to guide the fovea to unique areas within a scene that might provide the most efficient processing [5].

The computational model is implemented on a computer using digital pictures as stimuli to produce a pre-attentional or "saliency" map [9]. To create a saliency map, the model receives input from pixels within a digital picture. Then, it extracts three feature channels – color, intensity, orientation – at eight different spatial scales. These three channels are normalized and differences of center-surround are calculated for each separate channel. The separate channels are additively combined to form a single saliency map. An image's saliency map provides predictions of where spatial attention should be deployed [for detailed explanations refer to 6, 13]. In essence, the model makes predictions about which regions in an image have the most and least likely chance to be attended based purely on stimulus-driven properties. The saliency model is available for download from <SaliencyToolbox.net> as a collection of Matlab functions and scripts [17].

Testing a Saliency Model within Web Pages

Designers recognize the need to predict and identify where users' attention will be guided on a web page. For example, it is well known that one should avoid using poor designs that increase the likelihood of users missing important interface features such as branding, navigational or informational symbols. But, using an eye tracking system to monitor guidance of attention – as is traditional – can be expensive, difficult to employ and time consuming within the context of a practical iterative design process. Thus, we investigated the utility of a computational saliency model in predicting the guidance of attention in web page screenshots. This new method is benchmarked and compared to another set of data in which participants' eye movements were tracked while they viewed the same web page screenshots.

METHOD

Participants

The data from eight undergraduate participants are examined. All participants reported extensive web site experience.

Stimuli and Equipment

The images were 50 screenshots of various web pages. Each participant saw each screenshot only once.

Participants' eye movements were recorded by an ASL eye tracker with a sampling rate of 120 Hz. Screenshots were shown on a Samsung LCD monitor, which had a viewing area of approximately 38.0 cm × 30.0 cm. A chin rest maintained a viewing distance of approximately 80 cm. Images subtended approximately 26.7^0 x 21.2^0 visual angle.

Procedure

Participants first read and signed an informed consent document, and were then seated in front of the monitor with their chin in the chin rest. The experiment began and concluded with a 9-point calibration sequence to calibrate the eye tracker and estimate the amount of tracking error.

Participants were told that they would view a series of web page screenshots, and that they should, "look around the image like you normally would if you were surfing the internet." A fixation cross was presented at the center of the screen to signal the beginning of a trial. After a delay of approximately 1 second, a randomly selected web page screenshot was presented for 5 seconds. The fixation cross then reappeared to signal the beginning of the next trial.

The experiment took approximately 15 minutes to complete.

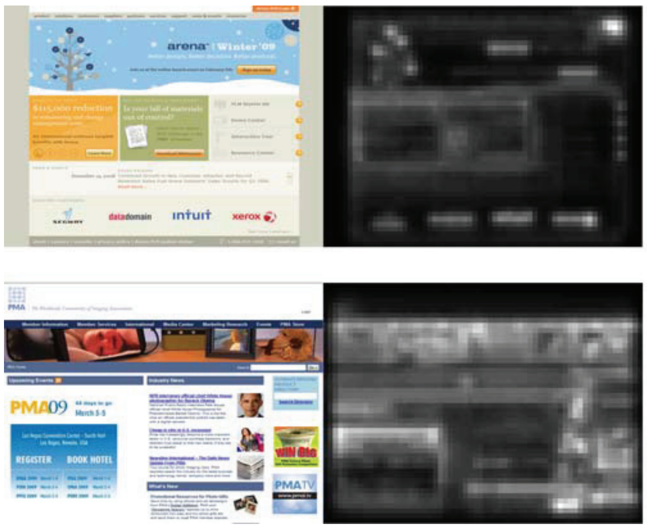


Figure 1. Two examples of web page screenshots and their corresponding saliency maps.

Creation of saliency maps

Saliency maps were created using the algorithms developed by Itti, Koch, and Niebur (1998). The model was run on each image individually and the output was normalized by dividing all values by the maximum value for that map, and multiplying all values by 100. To simplify data analysis, the size of the saliency maps was increased to be identical to the size of the screenshots (1024 x 768 pixels). As described in the Introduction, these saliency maps are 2-D representations of areas in the screenshot that show the relative saliency of locations in the image. Figure 1 shows an example of two web page screenshots and their corresponding saliency maps. Low values (dark areas in the image) indicate regions of the image that are low in saliency, while high values (light areas in the image) indicate regions high in saliency.

RESULTS

We used a similar technique to Parkhurst, Law, and Niebur (2002) to determine whether salient regions in web pages were fixated more often than would be expected by chance. Specifically, the values of the saliency map at the location of each participant's first ten fixations were extracted. For example, the x, y coordinates of the first fixation for each participant was determined for every screenshot and the value at the same location in the corresponding saliency map was extracted. This process was repeated for fixations two through ten. These values formed the *Observed Distribution* of participant responses (Figure 2).

To determine the likelihood that salient regions would be fixated by chance, we repeated the process used to find the Observed Distribution after rearranging the fixations and saliency maps for all screenshots. For example, the values from the saliency map for screenshots 2 to 50 were extracted at the fixated locations from screenshot 1. The saliency values of all other screenshots were extracted at the location of the first ten fixations for all subjects for each screenshot. These values formed the *Shuffled Distribution*. The method used to create this distribution controls for spatial biases that may inflate correlations between fixations and salient regions. If the values of the Shuffled Distribution are larger than those of the Observed Distribution, it would indicate that participants fixated on regions that are lower in saliency than what is expected by chance. If, however, the values of the Observed Distribution are larger than those in the Shuffled Distribution, it would indicate that participants fixated regions that are higher in saliency than what is expected by chance.

Figure 2 shows the means for the Observed and Shuffled Distributions of the first ten fixations for each screenshot. An analysis of variance was conducted with fixation number (1-10) as a within-subjects variable and distribution (observed, shuffled) as a between-subjects variable, to determine whether any differences between the distributions varied as a function of fixation number. The main effect of fixation number was reliable, $F(9, 882) = 6.39$, $MSE = 19.03$, $p < .001$. Pairwise comparisons revealed that the values for the first fixation were higher than all other values, and that the values of the tenth fixation were lower than all other values. This indicates that early fixations tend to occur at regions of higher salience than those of later fixations. More importantly, the main effect of distribution was also reliable, $F(1, 98) = 4.86$, $MSE = 397.95$, $p < .05$, indicating that the values of Observed Distribution were larger than those of the Shuffled Distribution. This difference confirms that participants fixated regions higher in saliency than would be expected by chance, showing that the saliency model is effective at predicting fixations. Distribution x Fixation number was not significant, $F < 1$.

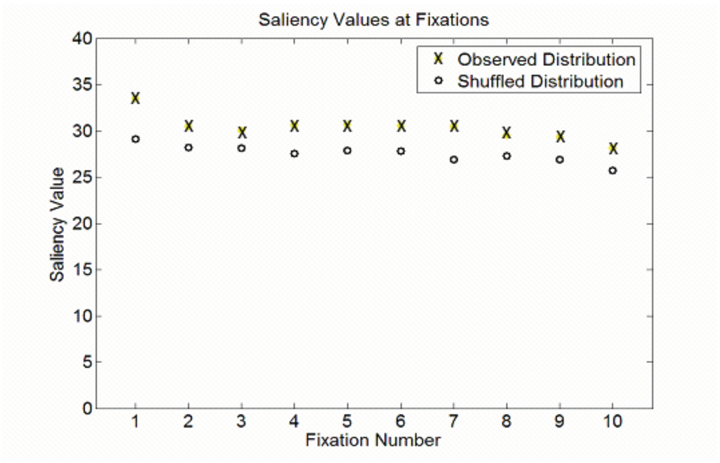


Figure 2. Mean saliency values for the observed ('X') and shuffled ('o') distributions for the first ten fixations.

DISCUSSION

Eye tracking is a commonly employed method for examining the guidance of overt attention within interfaces (e.g., web pages). However, it has several drawbacks. We propose that a web page's saliency, stimulus-driven properties, may be revealed through the use of a computational saliency model. Therefore, we compared the performance of the model to eye tracking data collected from human observers. We were able to demonstrate that, indeed, the saliency model predicts the deployment of overt attention within a web page interface.

Previous research has shown a modest correlation between saliency and eye fixations in natural and artificial scenes [13]. We have extended this research by showing that even in web pages, which may contain more semantic information (e.g., meaningful: text or images) than nature scenes, fixations are correlated with saliency. Specifically, participants were more likely to fixate on regions in the web pages with a higher saliency value than predicted by chance.

Our data suggest that saliency maps alone can provide reasonable predictions of overt attention. In addition, saliency maps can be generated quickly, and require no additional equipment or participants. Even with these positive attributes, one may be hesitant to abandon eye tracking altogether. Our recommendation to designers is to choose the method most appropriate for your project given your constraints and needs. It is often the case that developing effective interfaces requires many levels of analysis. For example, during the early formative testing process it would be appropriate to begin by using the saliency model to ensure that regions identified as being important are also visually salient. Then, during the 'final' prototype development stage, employ the eye tracking method to verify that your participants are actually looking at the critical elements in the design.

REFERENCES

1. Arroyo, E., Selker, T. & Wei, W. Usability tool for analysis of web designs using mouse tracks. *Computer-Human Interaction extended abstracts on human factors in computing systems* (2006), 484-489.
2. Chen, M., Anderson, J. R. & Sohn, M. What can a mouse cursor tell us more?: Correlation of eye/mouse movements on web browsing. *Computer-Human Interactions extended abstracts on human factors in computing systems* (2001), 281-282.
3. Cutrell, E. & Guan, Z. What are you looking for? An eye-tracking study of information usage in web search. *Proceedings of the SIGCHI conference on human factors in computing systems* (2007)*,* 407-416.
4. Egeth, H. E. & Yantis, S. Visual attention: Control representation and time course. *Annual Review of Psychology* (1997)*, 48*, 269-297.
5. Itti, L. & Koch, C. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research, 40*(10-12) (2000), 1489-1506.
6. Itti, L., Koch, C. & Niebur, E. A model of saliency-based fast visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11): 1254-1259, November 1998.
7. Johansen, S. A. & Hansen, J. P. Do we need eye trackers to tell where people look? *Proceedings of Computer-Human Interaction extended abstracts on human factors in computing systems* (2006), 923-928.
8. Johnston, W. A. & Dark, V. J. Selective attention. *Annual Review of Psychology*(1986), *37*, 43-75.
9. Koch, C. & Ullman, S. Shifts in selective visual attention: Towards the underlying neural circuitry. *Human Neurobiology* (1985)*, 4*, 219-227.
10. Kowler, E., Anderson, E., Dosher, B. & Blaser, E. The role of attention in the programming of saccades. *Vision Research* (1995), *35*, 1897-1916.
11. McCarthy, J. D., Sasse, M. A. & Riegelsberger, J. (2003). Can I have the menu please? An eyetracking study of design conventions. *Proceedings of Human-Computer Interaction*, 401-414.
12. Nielsen, J. (2008, May). *How little do users read?* Retrieved May 12, 2009 from http://www.useit.com/alertbox/percent-text-read.html.
13. Parkhurst, D., Law, K. & Niebur, E. Modeling the role of salience in the allocation of overt visual attention. *Vision Research* (2002)*, 42*, 107-123.
14. Rayner, K. Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin* (1998), *124*(3), 372-422.
15. Tarasewich, P., Pomplun, M., Fillion, S. & Broberg, D. The enhanced restricted focus viewer. *International Journal of Human-Computer Interaction* (2005), *19*(1), 35-54.
16. Treisman, A. M. Perceptual grouping and attention in visual search for features and for objects. *Journal of Experimental Psychology: Human Perception and Performance* (1982)*, 8,* 194-214.
17. Walther, D. & Koch, C. Modeling attention to salient proto-objects. *Neural Networks* (2006)*, 19*, 1395-1407.
18. Wolfe, J. M. Guided Search 4.0: Current Progress with a model of visual search. In W. Gray (Ed.), *Integrated Models of Cognitive Systems* (pp. 99-119). New York: Oxford, 2007.

Embedding Requirements in Design Rationale to Deal Explicitly with User eXperience and Usability in an “intensive” Model-Based Development Approach

Célia Martinie, Jeff Ladry, David Navarre, Philippe Palanque & Marco Winckler
IRIT - University Paul Sabatier
118, route de Narbonne, 31062 Toulouse Cedex 9, France
{martinie, ladry, navarre, palanque, winckler}@irit.fr

ABSTRACT

Requirements engineering for interactive systems remains a cumbersome task still under-supported by development processes. Indeed, in the field of HCI, currently the most common practice is to perform user testing to assess the compatibility between the designed system and its intended user. Other approaches such as scenario-based design [14,16], promote a design process based on the analysis of the actual use of a technology in order to design new technologies better supporting users’ tasks. However, these approaches do not provide any support for a) the definition of a set of requirements that have to be fulfilled by the system under design and b) as a consequence for assessing which of these requirements are embedded in the system and which ones have been discarded. This position paper proposes a notation and a tool for addressing precisely these two challenges. These elements are integrated within a more global approach aiming at providing notations and tools for supporting a rationalized design of interactive systems following a model-based approach.

Keywords

Model-based UI design, requirements, interaction technique, design rationale, user experience, usability.

INTRODUCTION

Traceability of choices and systematic exploration of options is a critical aspect of the development processes in the field of safety critical systems. Some software standards such as DO 178 B [15] (which is widely used in the aeronautical domain) require the use of methods and techniques for systematically exploring design options and for increasing traceability of design decisions. However, such standards only define what *should* be done and provide no information on *how* such goals can be reached by designers. Recent work in the field of software engineering has been trying to provide solutions to that problem and a collection of papers on that topic can be found in [4]. One of the remaining problems pointed out by many contributions, such as chapters 1, 19 and 20, is that requirements are poorly or even not addressed. However, Requirements Engineering (which is the first phase of the development process) provides input to all the subsequent phases and must be dealt with adequately. Indeed, ESARR (Eurocontrol Safety Regulatory Requirement) on Software in Air Traffic Management Systems [5] explicitly requires traceability to be addressed in respect of all software requirements (p. 11 edition 0.2).

This position paper addresses the problem of traceability of requirements for model-based approaches. It tackles the problem by providing an extension to a notation TEAM and its associated tool DREAM which have previously been presented in [9]. The current contribution makes it possible to relate design options with functional and non functional requirements. While other approaches such as SCRAM [16] focus on requirements identification, our approach is intended for supporting the traceability of such identified requirements within interactive systems models. As an example we show how two different interaction techniques modeled using ICOs [11] satisfy different functional and non-functional requirements. While the approach could address any kind of requirements, we put the emphasis on *Usability* and *User eXperience*.

Next section quickly presents the basic principles of the TEAM notation and the extensions that have been made to include information related to requirements. They will then be used in the third section on a case study for comparing the two interaction techniques with respect to requirements providing ways of answering two fundamental questions: 1) Which current design (among the many ones available following a prototyping phase for instance) satisfies a given requirement and 2) What is the exhaustive list of requirements fulfilled by a particular design.

EXTENSION OF THE TEAM MODEL AND NOTATION

The TEAM notation (Traceability, Exploration and Analysis Method) and its CASE tool DREAM (Design Rationale Environment for Argumentation and Modeling) has been proposed in [9] to support the exploration of options and traceability of choices during the development process of interactive safety critical systems.

TEAM notation

The TEAM notation is based on Question Option Criteria Design Rationale notation introduced by MacLean and al. [10]. QOC notation allows to list the available options for a design interrogation and to trace the selection of an option with regards to a list of relevant criteria. The TEAM notation is an extension of QOC that enables the recording

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

(in an exhaustive manner) of much information produced during design meetings. Such information can be:

- The questions that have been raised,
- The design options that have been investigated and the ones that have been selected,
- The evaluation performed for the different options,
- The collection of factors that have been taken into account and how they relate to evaluation criteria,
- The task models corresponding to options
- The resulting scenarios

Besides this recording of information, an important feature is to record design decisions and relate them to selected factors.

This notation and associated tool can then leverage the design rationale process for several interactive applications and help engineers in deciding to reuse or not conception choices when facing an already experienced issue.

Adding requirements to the TEAM notation

In the earlier version, the notation did not allow to express the needs and requirement. But this is required from the designers' extensive work to trace back for the selected options what were the requirements met. In addition, this lack of relationship prevents designers from exploiting requirements for the generation of options and/or to take into account requirements aspects when designing an option. We have thus added requirements as an explicit entity in the TEAM notation and in the DREAM tool. Several requirements are represented in Figure 1 (grey rectangles with a folded corner on the top left hand side) but the content of the figure will only be described in the next section. As far as HCI aspects are concerned this addition is very important as it makes it possible to explicitly represent contributing factors to usability and User eXperience (UX) and thus assess the relevance of design options to them. The requested usability requirements relate to the efficiency and effectiveness factors of the system from an ISO 9241-11 perspective [8]. The requested user experience requirements relate to the pragmatic quality and stimulation hedonic quality factors of the system from a Hassenzahl perspective [6]. This specific aspect will be detailed in the next section through a case study.

CASE STUDY

The application we chose allows a user to remove a set of icons on a computer screen. It has first been used and presented by Maurice H. ter Beek and al. to evaluate the "*Resilience of interaction techniques to Interrupts*" [17]. In order to represent design choice and design rationale we consider two different interaction techniques for performing that task. The first interaction technique only uses a mouse-based one while the second is multimodal as it uses speech additionally. The first interaction technique is an enriched drag and drop interaction, with which the user is able to move an icon onto the trash icon. When the system detects that the icon file has entered a 2 centimeters circle area around the trash icon, the user has 2 seconds to drop the file icon on the trash, or the trash icon will move to another location on the display. More sophisticated techniques could be easily modeled using our approach but this is beyond the scope of the position paper. The second interaction technique is a speech and click interaction where the user utters "delete" and clicks on the icon to be deleted.

The *Speech & Click* and *enriched Drag & Drop* options, among other design choices, are going to be evaluated according to the initial requirements for the application. One having an important impact is the one requiring the interaction technique to be tolerant to interruptions (for more details see [17]).

Presentation

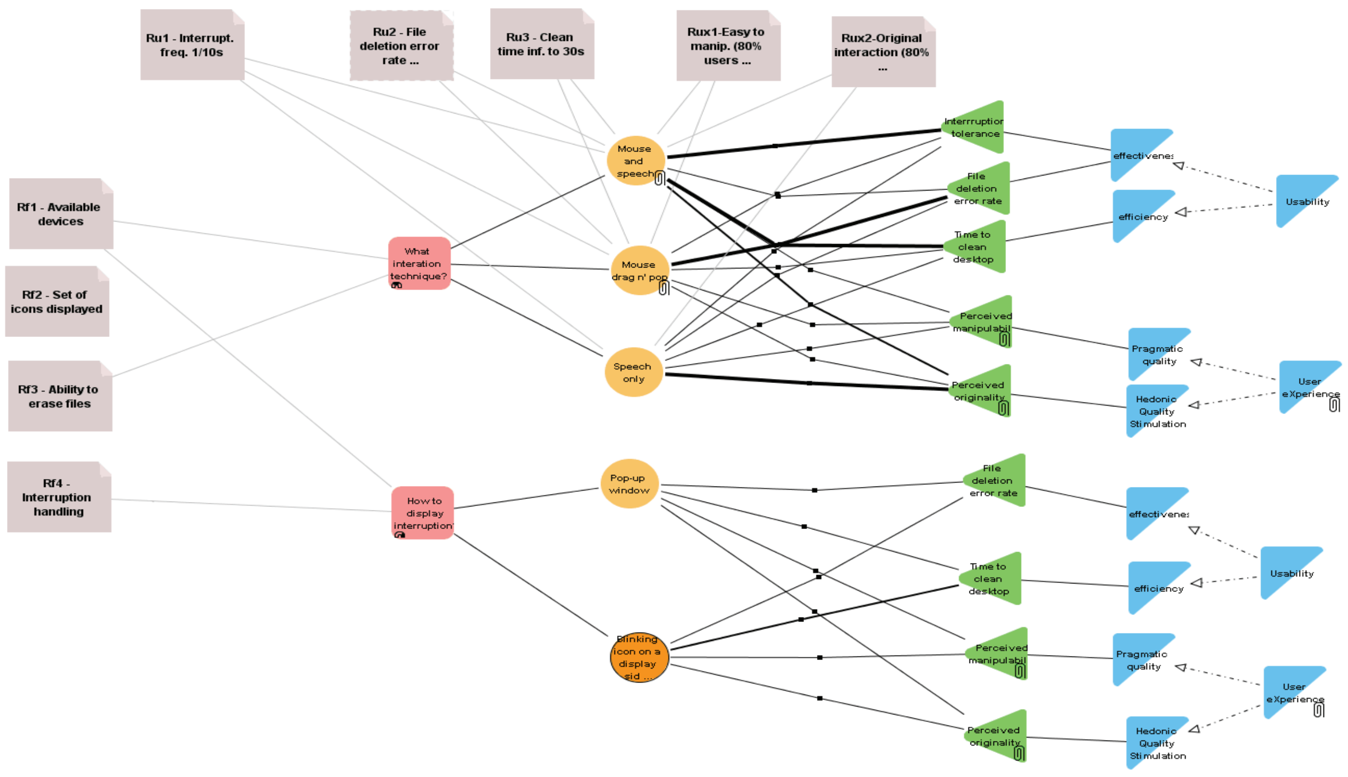
An initial list of these requirements gathers functional needs and non-functional ones (mainly Usability and User eXperience). An extract of the set of usability requirements for this user interface are ("u" stands for usability requirement):

- Ru1-The application shall support one interruption every 10 seconds
- Ru3-The application shall allow the user to clean the desktop in less than 30 seconds

The set of user experience requirements for this user interface are ("ux" stands for user experience):

- Rux1-80% of the users shall find the application easy to use
- Rux2-80% of the users shall find that interacting with this application is surprisingly different

Given this list of requirements, evaluation criteria are defined and linked to appropriate factors. The different questions, options and design paths with regards to elected criteria are consigned within the DREAM design rationale tool. Figure 1 gives an overview of the design options linked to requirements and evaluation criteria. Due to space constraints, the schema has been shrunk, but several display techniques for the tool are currently being studied for big and complex diagrams. Rectangles show the requirements, rounded-shape squares contain the design questions that have been asked, right-angle triangles on the right side describe factors and the other triangles describe criteria. Each question has several possible options to address the issue. For instance, as far as the interruption is concerned ("How to display interruptions?"), the interruption can be displayed as a "pop-up window" or as a "small icon blinking" on one side of the display. In the TEAM mode, the connection between these options and the four criteria "File deletion error rate", "Time to clean desktop", "Perceived manipulability" and "Perceived originality" represents the relative impact of the option. The strong link between the option "small icon blinking" and the "time to clean desktop" criterion shows that it favors that criterion.



Ru1 - Interrupt. freq. 1/10s
Ru2 - File deletion error rate ...
Ru3 - Clean time inf. to 30s
Rux1-Easy to manip. (80% users ...
Rux2-Original interaction (80% ...
Rf1 - Available devices
Rf2 - Set of icons displayed
Rf3 - Ability to erase files
Rf4 - Interruption handling
What interaction technique?
How to display interruption
Mouse and speech
Mouse drag n' pop
Speech only
Pop-up window
Blinking icon on a display sid ...
Interrruption tolerance
File deletion error rate
Time to clean desktop
Perceived manipulabil
Perceived originality
effectivenes
efficiency
Usability
Pragmatic quality
Hedonic Quality Stimulation
User eXperience

Figure 1. Design Rationale overview from a design session output with DREAM

The right-hand side of the diagram makes explicit the relationship between criteria and factors. Factors correspond to desired characteristics of the system namely "Usability" and "User eXperience". They are in turn decomposed into sub-factors; two out of the three classical ones for usability i.e. "effectiveness" and "efficiency" and two for "User eXperience" i.e. "Pragmatic quality" and "Hedonic Quality Stimulation".

Design options modeling

One of the issues that remain to be solved is how to assess the options with respect to the connected criteria. For instance, how to identify if the option "Mouse and Speech" for the interaction technique is better than the option "Enriched Drag and Drop" with respect to the criterion "Time to clean desktop". In order to address that problem we propose to apply a model-based approach and to define for each option a detailed behavioral description. For that purpose we use the ICO notation [11], but other ones such as [2 or 6] would also be applicable. ICO notation stands for Interactive Cooperative Objects and is a formal notation to describe interactive systems. It is based on object-oriented design pattern and high-level Petri nets. The PetShop associated tool [1 and 13] allows editing the ICO model and prototyping the associated interface at the same time (this aspect is detailed in [12]).

The DREAM diagram shows that a third design option, speech only, had initially been suggested and that this option does not fulfill all the non-functional requirements, even before having prototyped or modeled it. The modeling of an option has a cost indeed and the gain of this approach is effective if the number of modeled and prototyped options is balanced with the modeling and prototyping cost. To help in comparing the two remaining interaction techniques, models of these interactions have been built and embedded in the DREAM diagram as indicated by the paperclip symbol at the bottom-right of each option. ICO models of the two interaction techniques are built and assessed (Enriched Drag and Drop technique ICO model detailed in Figure 2), with respect to both "user experience" and "usability" factors. These models are then verified and prototyped by means of PetShop tool. Effectiveness can be verified and performance evaluation technique can be used to assess time performance for instance. Due to space constraints this is not detailed here. Usability tests can be performed on the prototypes with the building of the application while performance evaluation can be done on the model only. For the user experience aspects, using the prototyped options, an evaluation has been performed with AttrakDiff tool [7] to rate the selected criteria. Paperclip symbols on "perceived manipulability" and "perceived originality" criteria represent the fact that such results are stored in the design rationale model. Furthermore, to help in checking this coverage and to ensure that one or more evaluation criteria are not missing, the tool allows linking the requirements to criteria.

Interpretation and benefits

Once the design options have been assessed, DREAM model makes it easy to perceive which one (if any) has received the best evaluation.

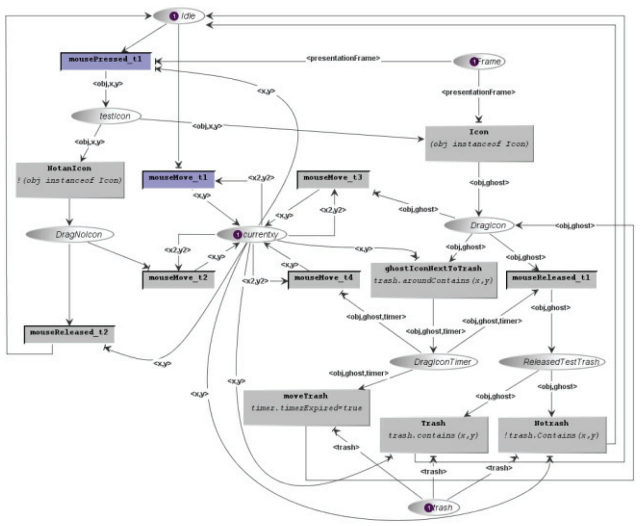


Figure 2. ICO model of Enriched Drag & Drop

In addition it makes it explicit for this option which requirements it fulfils. Coming back to the importance of requirement engineering, DREAM diagram is a critical help to argue about, to select an option as well as to trace back the rationale underlying this selection. In our example, we see at a glance that the “Mouse and Speech” option is the best rated and that it fulfills the entire set of non-functional requirements. The final choice then belongs to the various stakeholders who have additionally direct access to the related requirements which, in turn, cannot be ignored. Furthermore, all the necessary information about the designed interactive system can be gathered and synthesized in one diagram. From a designer perspective, it can help to share conception ideas and materials. From another participant involved in the implementation and/or deployment process, it can be seen as an entry point to the designed system.

CONCLUSION

This position paper argues that various models are useful for the design of interactive systems as they can complement each other and correspond to the needs of different stakeholders. We have presented a model-based approach for description within and single diagram requirements, design questions, design options, criteria and factors. This structured set of information supports different activities such as requirements traceability, design choices decision support and the traceability of design choices decisions. We have also presented how detailed behavioral description of advanced interaction techniques such as “Enriched drag and drop” or “speech and click” can be integrated within that framework to provide additional benefits. Of course such “intensive” model-based approaches require tools to support model construction, analysis, simulation, interpretation and reuse. The CASE tools supporting TEAM and ICO notation are publicly available [3 and 13] and are a corner stone of the applicability of the approach.

REFERENCES

1. Bastide, R., Navarre, D., and Palanque, P. 2002. A model-based tool for interactive prototyping of highly interactive applications. In CHI '02 Extended Abstracts on Human Factors in Computing Systems (Minneapolis, Minnesota, USA, April 20 - 25, 2002). CHI '02. ACM, New York, NY, pp. 516-517.
2. Coninx, K., Cuppens, E., De Boeck, J. & Raymaekers, C., 2007, Integrating Support for Usability Evaluation into High Level Interaction Descriptions with NiMMiT . In: Interactive Systems: Design, Specification, and Verification. 2007, Springr Verlag LNCS, pp. 95-108.
3. DREAM web site: Internet Resource http://ihcs.irit.fr/dream/index.html , accessed Jan 2010.
4. Dutoit, A.H., McCall, R., Mistrík, I., Paech, B., Rationale Management in Software Engineering: Concepts and Techniques, Rationale Management in Software Engineering, p. 1.
5. ESARR 6. EUROCONTROL Safety Regulatory Requirement. Software in ATM Systems. Edition 1.0. http://www.eurocontrol.int/src/public/standard_page/esarr6.html (2003)
6. Hassenzahl, M. The thing and I: understanding the relationship between user and product. In M.Blythe, C. Overbeeke, AF Monk, & PC Wright (Eds.), Funology: From Usability to Enjoy-ment. Dordrecht: Kluwer, 2003, pp. 31-42.
7. Hassenzahl, M. - Internet Resource http://www. attrakdiff.de, accessed Jan 2010.
8. ISO DIS 9241-11 (1996) Ergonomic requirements for office work with visual display terminals (VDT) - Part 11 Guidance on usability.
9. Lacaze, X., Palanque, P., Barboni, E., Bastide, R., Navarre, D., From DREAM to Reality : Specificities of Interactive Systems Development With Respect to Rationale Management, Rationale Management in Software Engineering, pp.155-170.
10. MacLean, Allan; Young, Richard M.; Bellotti, Victoria M. E., and Moran, Thomas P. Questions, Options, and Criteria: Elements of Design Space Analysis. Lawrence Erlbaum Associates; 1991; 6, pp. 201-250.
11. Navarre, D., Palanque, P., Ladry, J., and Barboni, E. 2009. ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. ACM Trans. Comput.-Hum. Interact. 16, 4 (Nov. 2009), pp. 1-56.
12. Palanque P., Ladry J., Navarre D. and Barboni E. High-Fidelity Prototyping of Interactive Systems can be Formal too 13th International Conference on Human-Computer Interaction (HCI International 2009) San Diego, CA, USA.
13. PetShop web site: Internet Resource http://ihcs.irit.fr/petshop, accessed Jan 2010.
14. Rosson, M.B. & Carroll, J.M. 2002. Usability Engineering: Scenario-Based Development of Human-Computer Interaction. San Francisco: Morgan Kaufmann.
15. RTCA. Software Considerations in Airborne Systems and Equipment Certification, DO-178B RTCA, Washington D.C. 1992
16. Sutcliffe A. & Ryan M. Experience with SCRAM, a SCenario Requirements Analysis Method, in Proceedings of the 3rd International Conference on Requirements Engineering: Putting Requirements Engineering to Practice, April 1998, pp. 164-173.
17. Ter Beek, M. H., Faconti G. P., Massink, M., Palanque P., Winckler M. Resilience of interaction techniques to Interrupts: A Formal Model-Based Approach. In proc. of Human-Computer Interaction (INTERACT 2009), Uppsala, Sweeden, Vol. 1, Springer-Verlag, LNCS 5726, p. 494-509.

A global process for using model-driven approaches in user interface design

Sybille Caffiau, Patrick Girard
LISI / ENSMA
Site du Futurosope – Téléport 2
86961 Futuroscop Chasseneuil Cédex – France
{caffiaus, girard}@ensma.fr

ABSTRACT
In user interface design, model-driven approaches usually use a generative solution, which has obvious limitations, especially for advanced user interfaces. Based on strong associations between task models and dialogue models, we propose a global process, which facilitates the design of interactive applications conform to their models, with the including of a rule-checking step. This process permits either to start from a task model or a user-defined prototype. In any case, it allows an iterative development, in line with user-centered design standards.

Author Keywords
Task Model, Dialogue Model, Metamodel, Design Method, Model-Driven Approach.

ACM Classification Keywords
H5.2 [Information Interfaces and Presentation]: User Interfaces, User-centered design; H.1.2 [Models and Principles]: User/Machine Systems.

INTRODUCTION
Model-driven approaches have been promoted for years. Despite their great interest, they remain hard to use in the context of user-centered design, especially when novel interaction techniques are expected. Thus, several research works [1-3] used a generative approach to build user interfaces – mainly skeletons to be completed – from task models. Following the analysis we made in [4], we can argue that this approach has several limitations:

- Generating requires the addition of information in order to reach an operative stage of interfaces. This information can be added to high-level models, which then loose their original goal; so doing they become hard to understand and to use, because of their multiple semantics (for example, adding presentation information to task models results in adding new semantics to this model). The other way is to insert this information during the generating process. This second approach is for example used in TERESA [5] by the way of heuristics, which are applied during the process. This however results in a lack of understanding of such transformations by users.
- All considered research issues are concerned with classical WIMP[1] applications. The hierarchical structure of task models is used to build the interface navigation scheme. We demonstrated in [4] that introducing non menu-based interactions implies a non-automatic transformation of the dialogue.
- Generating is not easy to include in iterative design cycles such as HCI-adapted cycles. When changes are required, it is necessary to modify the high-level models, and to generate again a new skeleton, to be improved again by hand-made add-ons. Some results have been obtained around the definition of "round-trip engineering" [6, 7], but were not applied to HCI. More, this approach prevents the designers from starting from the prototype, which method is often used in post-WIMP design.

Our aim is to introduce a new way to use models in user interface design. Leaning on meta-models of one task model and one dialogue model, we wrote equivalence rules between such models. Then, we defined a new development cycle that can be used in a user-centered iterative approach.

In this paper, the context of the used models is first described. Then, an example of the meta-models is given, and equivalence rules are presented. In the third part, the proposed way to use these models in a development cycle is outlined.

CONTEXT OF THE STUDY: THE MODELS
The starting point of our work is the analysis from [4]. Whilst the generation appears to be too limitative, links between task models and user interfaces seem obvious. So, we decided to explore the possibility to establish strong links between task model and other models, and to consider exploiting said links in software design methods. For some reasons, which are external to our subject here, we chose the K-MAD model [8] as our task model.

In our laboratory, we have been working for some time on dialogue models and formal approaches in HCI. We introduced a software architecture model, H^4, which was

[1] Windows, Icons, Menus and Pointers

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

first dedicated to computer-aided systems. Coupled with that architectural model, we proposed a dialogue model, the hierarchical interactors [9], and developed tools to apply it [10]. Because of its hierarchical structure, the Hierarchical Interactor (HI) model appeared as the most suitable for our purpose. A previous study demonstrated the capacity to exploit these two models (K-MAD and HI) in HCI design [11].

Then, we defined the meta-model of these two models, which is published in [12]. We chose to use the EXPRESS language, an alternative to the OMG approach for meta-modeling. EXPRESS is a standard data modeling language for product data. It is formalized in the ISO Standard for the Exchange of Product model STEP (ISO 10303), and standardized as ISO 10303-11[2]. It is supported by complete verification tools, and allows a full expression of constraints [13].

PRINCIPLES

In this section, we give a short description of the K-MAD and HI models, and provide some examples of the meta-models.

The K-MAD model

The K-MAD model is a hierarchical model where tasks are decomposed in sub-tasks, with temporal operators describing the dynamics of the model. The description can be enhanced by the definition of objects and expressions (preconditions, post-conditions, and actions) to control the model dynamics more precisely. The semantics of these different elements is defined in details.

Figure 1 illustrates a sample of the EXPRESS definition of the central element of the model, the task.

The Hierarchical Interactor (HI) model

The HI model consists in a state machine model where the dialogue of the application is split into independent automata. Transitions are activated by tokens that represent user inputs or automaton productions.

The hierarchical organization of the model allows the automata to produce and consume tokens. The main advantages of this system are two-fold:

- Automata are independent from each other. They can be removed or added independently, without any change to others.
- Tokens are the key elements of the model. As they can refer to both user entries and automaton productions, they break the binding between user inputs and transitions. This allows to consider the dialogue at the level of abstract level one wants. This is particularly important when post-WIMP interaction techniques are used.

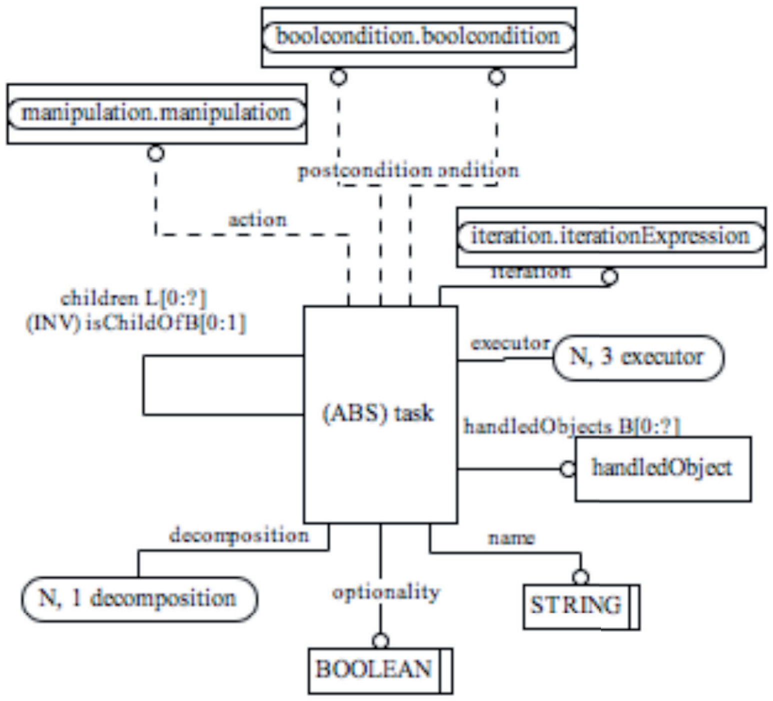


Figure 1: EXPRESS definition of the Task entity (partial)

Such as in advanced state machines, transitions may be guarded by expressions, which involve variables. They also can trigger actions. Figure 2 (next page) illustrates a transition meta-model.

Associations between models

The general philosophy of our approach is to take advantage of the hierarchical nature of the two models to establish strong associations between them.

The task/transition association

The first obvious association can be made between tasks from the task model and transitions from the dialogue model. This link has been largely used in the previous research works, but for us, the link is not a bijective link: because of the need for interaction facilities in applications, there can be more transitions than user tasks.

The compound-task/automaton association

The structure of the dialogue model encourages considering each task decomposition as equivalent to a specific automaton. The structure of the automaton must then be compatible with the dynamics described through the temporal operator of the compound task. Again, the dialogue may be richer than the simple translation of the temporal operator. Another consequence of this association is the equivalence between tokens and

[2] http://www.tc184-sc4.org/

compound tasks: each compound task may be achieved by the way of an automaton that produces a token that stands for the task achievement.

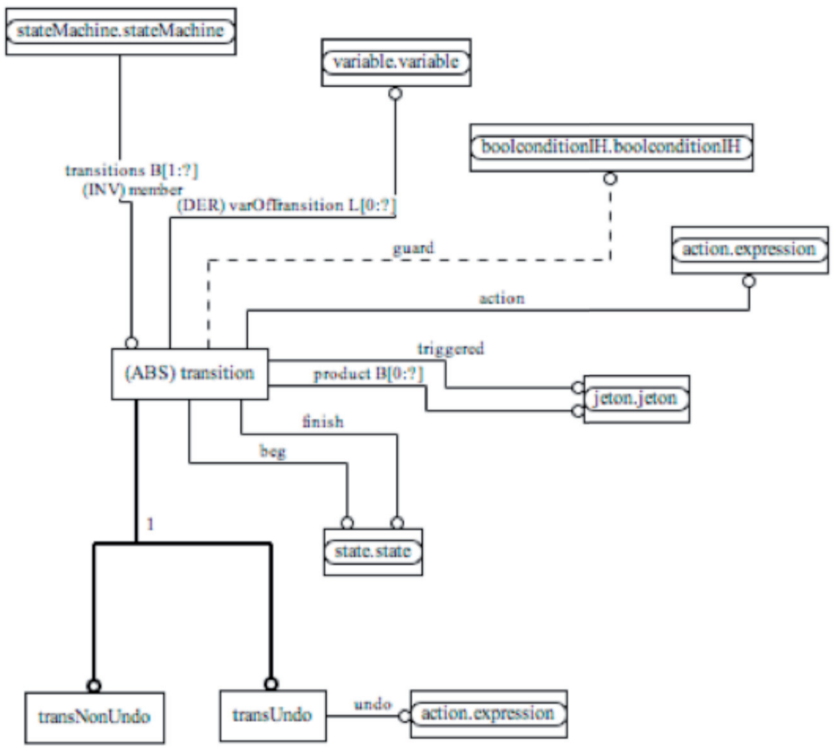


Figure 2: EXPRESS definition of the Transition entity

The object/variable and expression associations
Both task model and dialogue model use expressions, which manipulate objects/variables. This link is patent, but was not described in the previous works because the used task model did not formally consider objects and expressions.

Rules between models
Two kinds of rules can be established between the models [12], based on the previously defined associations.

The first kind of rules concerns the existence of logically associated entities in both models. For example, are there one token and one automaton for each compound task? Or is there one transition in the dialogue model for each task in the task model?

The second kind of rules relates to the semantics of the models. Are the semantics of the expressions we can find in each model equivalent? Is the navigation, which is allowed by the automata, consistent with the temporal decomposition of the tasks?

These rules can be exploited in two ways. They can be used in initial design to generate a skeleton of the dialogue, or they can be used in verification to state that two models are compatible. In that way, our work might be compared to [14]. We describe in the next section the different usages of this duality.

THE GLOBAL PROCESS
In this section, we describe the global process we propose to utilize the model-driven approach we describe above.

As previously claimed in the introduction, generating dialogue model from task models suffers from drawbacks; the most important of them is related to the iterative nature of user-centered approaches. When changes must be made in response to new or enhanced user needs, the generating process must be run again, and all hand-made changes in the interface are lost. We argue that, if a generation phase occurs, it must be restricted to a starting point; then, the process must be able to achieve without any further generation. The scenario schema we propose is as follows.

Assuming we are able to design, edit and verify each of both task and dialogue model. Each of these phases will be called "X-editing phase" thereafter. These phases may be realized independently from each other. Our model-driven approach consists in including these phases in a dynamic design process.

Optionally, one can start by a "task-editing phase", from which a starting skeleton for the dialogue model can be derived (e.g. generated, but only once). Either kinds of rules, existence rules and semantic rules, can be used to produce this skeleton. Then, the next phase consists in filling in the skeleton, in a "dialogue-editing phase". Adding specific dialogue elements, the dialogue model can be completed.

During this step, the two models can be confronted for detecting inconsistencies. By adding specific interaction elements to the skeleton, the designer might have changed the semantics of the model.

To reach this objective, the designer must associate the two models: some added dialogue entities might be related to task entities.

After analysis, depending on the result, different solutions can be applied:

- *Fail. The two models do not match. Some tasks are missing in the dialogue model.* The dialogue model must be improved to take into account the whole task model.

- *Fail. The two models do not match.* The dynamics of the two models differ. The task model and/or the dialogue model must be modified.
- *Success. The two models match.* The system is now ready to being tested by users.

A user evaluation phase may result in new requirements, which may lead us to coming back to either dialogue or task modeling, and resuming the loop.

Figure 3 is a Petri net diagram that represents the global process. The process can start either from the Dialog-Editing Phase or the Task-Editing Phase. After rule checking, a failure results in redoing both Task and Dialogue Editing Phases. If problems are detected with usage or interaction during user evaluation, the process must also be repeated.

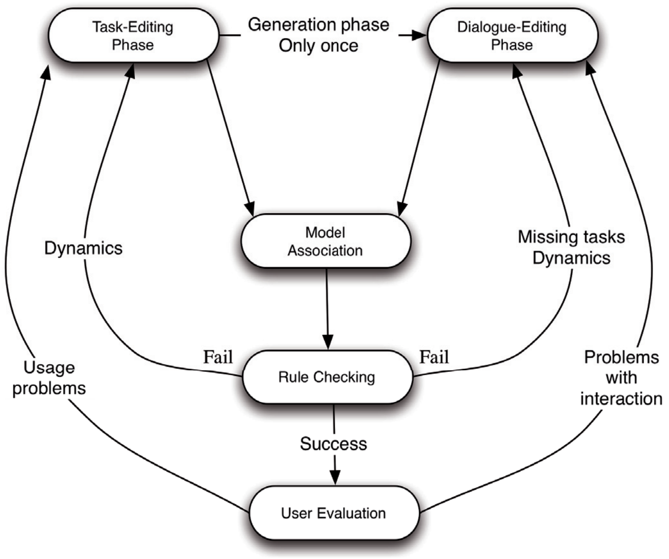


Figure 3: The global process.

CONCLUSION

In this paper, we present a global process to use a model-driven approach in user interface design. This process uses rules that allow to check the validity of task models and dialogue models. Moreover, this process is compliant to user-centered approaches that promote iterative design.

REFERENCES

1. Mori, G., F. Paternò, and C. Santoro, Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. IEEE Transactions on Software Engineering, 2004: p. 507-520.
2. Luyten, K., et al. Derivation of a Dialog Model from a Task Model by Activity Chain Extraction. in DSV-IS'2003. 2003. Funchal, Madeira Island, Portugal: Springer-Verlag. p. 203-217.
3. Wolff, A. and P. Forbrig. Deriving User Interfaces from Task Models. in MDDAUI'09. 2009. Sanibel Island, USA: CEUR-WS. p. 4.
4. Caffiau, S., et al. Generating Interactive Applications from Task Models: a Hard Challenge. in TAsk MOdels and DIAgrams (TAMODIA). 2007. Toulouse, France: Springer Berlin/Heidelberg. p. 267-272.
5. Berti, S., et al. TERESA: a transformation-based environment for designing and development multi-device interface. in Conference on Human Factors in Computing Systems - CHI'04. 2004. Vienna, Austria: ACM NY. p. 793-794.
6. Hettel, T., M. Lawley, and K. Raymond. Model Synchronisation: Definitions for Round-Trip Engineering. in Theory and Practice of Model Transformations, ICMT 2008. 2008. Zürich, Switzerland: Springer. p. 31-45.
7. Sendall, S. and J. Küster, Taming Model Round-Trip Engineering, in OOPSLA Workshop on Best Practices for Model Driven Software Development. 2004: Vancouver, Canada.
8. Lucquiaud, V. Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs. in 17th French-speacking conference on Human-computer interaction. 2005. Toulouse. p. 83-90.
9. Depaulis, F., et al., Le modèle d'architecture logicielle H4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. Revue d'Interaction Homme-Machine, 2006. 7(1): p. 93-129.
10. Depaulis, F., S. Maiano, and G. Texier. DTS-Edit : an Interactive Development Environment for Structured Dialog Applications. in CADUI'02. 2002. Valenciennes (France): Kluwer Academics. p. 75-82.
11. Caffiau, S., et al., Hierarchical Structure: A Step for Jointly Designing Interactive Software Dialog and Task Model, in Human-Computer Interaction. Novel Interaction Methods and Techniques, Springer, Editor. 2009, Springer: Berlin. p. 664-673.
12. Caffiau, S., Approche dirigée par les modèles pour la conception et la validation des applications interactives : une démarche basée sur la modélisation des tâches, in LIIS/ENSMA. 2009, Poitiers: Poitiers. p. 240.
13. Dehainsala, H., et al. Ingénierie dirigée par lesmodèles en EXPRESS : un exemple d'application. in IDM. 2005.
14. Kavaldjian, S., et al. Transformations between Specifications of Requirements and User Interfaces. in MDDAUI'09. 2009. Sanibel Island, USA: CEUR-WS. p. 4.

E-Composer: Enabling the Composition of Mobile Assistants

Ilhan Aslan*, Dyuti Menon*, Robert Brauer*, Kristin Albert* and Christian Maugg*
Fraunhofer ESK, Germany*
name.lastname@esk.fraunhofer.de*

ABSTRACT

ELEPHANT (ELEments for Pervasive and Handheld AssistaNTs) is a system that aims to integrate a broad range of users (e.g. designers, domain experts and end users) with different backgrounds in the process of developing personal mobile assistants. In this paper we present a user study that we have conducted for two reasons: First, to screen characteristics of modeling mobile assistants by non-experts of mobile software development; and second, to test a first prototype of the ELEPHANT system's graphical modeling tool (E-Composer).

1. INTRODUCTION

Today, the use of mobile phones is very wide spread. In addition, the capabilities of mobile technology as also the underlying infrastructure are increasing on a regular basis. This development qualifies mobiles phones as digital companions in everyday life. However, when it comes to modeling the interaction for a broad spectrum of target users, target domains and context of use, the modeling process becomes very cumbersome. On the one hand, designing interaction and user interfaces is a profession in itself and most software engineers do not have the required skills to build user centered, attractive and usable interactions without being guided or having a framework set for them. On the other hand, general modeling languages (e.g. UML based) that are being used by software engineers are either too low level or foreign to most designers and domain experts. The ELEPHANT (ELEments for Pervasive and Handheld AssistaNTs) system aims to integrate non-software engineers (e.g. designers, domain experts and end users) in the process of developing personal mobile assistants. The ELEPHANT system's modeling tool that we refer to as the E-Composer allows a high level of modeling based on components [1]. One of the reasons why users access services while mobile is basically because they need assistance to complete an activity (e.g. shopping, dining, driving or route finding) or to proceed with an activity in the real world. Although today's mobile phones have advanced interfaces and can handle most websites that have been originally designed for the desktop environment, single services that focus on content and functionality are not sufficient in assisting mobile users during their specific activities. Especially, if users are involved in real world activities in which they are pressed for time, the assistance provided through the capabilities of the mobile phones has to be highly personalized and centered to the user's activity. The requirements on personalization and adaptation to user activities are very high. To fulfill these requirements, domain experts and end users have to participate in the design process. Therefore, the ELEPHANT system provides a browser based tool support for the participative design of mobile assistants. The E-Composer is the front-end of the ELEPHANT system that allows users to graphically compose mobile assistants based on components. The graphical presentation of a mobile assistant modeled with the E-Composer has a tree-like structure (see figure 1). The backend of the ELEPHANT system manages these components. Components can be accessed and tagged with information by all users. Users can search for components and they can set up a components library. In [1] we described the component based development of mobile assistants in more detail.

In order to derive essential feedback regarding the ELEPHANT's composer tool, its reception by users and its functionalities, we describe in this paper usability tests that we conducted to measure user satisfaction from working with the tool and the overall performance of the tool. A small test scenario was setup, where users were given the task of modeling a mobile assistant using the ELEPHANT composer. Based on the user reactions and suggestions during and after the tests, conclusions were drawn regarding the performance and efficacy of the composer and how it may be improved. In this paper we present a description about the usability tests, the set-up and the data, what we intend to deduce from these usability tests and what methods we used to evaluate the data.

2. User Study

The usability tests were conducted with 11 participants in the age group of 22 – 28 years. They came with different backgrounds in the areas of computer expertise, authoring systems and system modeling skills. The tests were conducted individually and in an undisturbed setting with the test subject being initially instructed as to the nature and goal of the test. The test subjects were advised to complete the test within 1 hour and to keep in mind that this test was composed of 2 separate tasks. Once the test subjects were given all the instructions and provided with all the material to proceed with the test, the members of our team left the premises The goal of the tests was for the participants to create a mobile assistant, which would assist a friend who would shortly be travelling to the city of Barcelona. This mobile assistant would aid the visitor with the Spanish language by helping them with the translations of common phrases (to buy tickets, order food etc.), be a guide for sightseeing in the city of Barcelona (by providing background information on the interesting places to see) and provide additional information such as suggestions about interesting places to eat or things to do in Barcelona. Keeping the generation of a Barcelona mobile assistant as the common goal, two tasks were designed to differentiate between a known and an unknown framework. The first task was to design a paper based Barcelona mobile assistant (see figure 2). The second task was to do the same, i.e. design a Barcelona mobile assistant, with the

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

help of the ELEPHANT composer (see figure 1). For both the tasks, the test subjects were provided with a list of content they had at their disposal to create this assistant. The content included text data, images, video clips and audio files, all connected to Barcelona and the Spanish language.

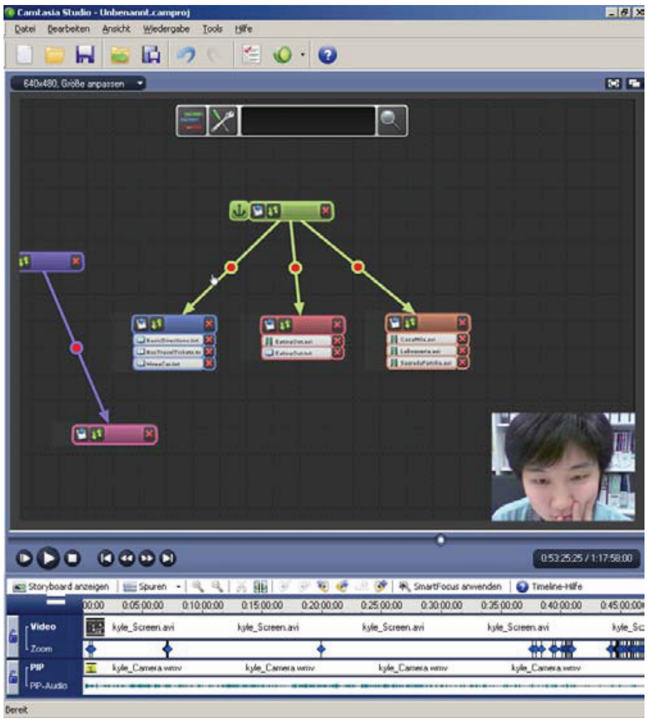


Figure 1: Screenshot of one of the subject's audio and video data

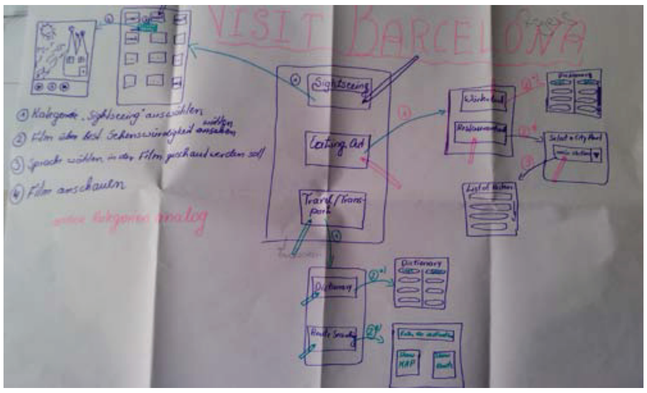


Figure 2: Photo of a result of one of the subject's paper based model of a mobile assistant

Our aim in conducting these tests was to measure the system performance, user satisfaction and the emotional response (in terms of stress and cognitive load on the participant) due to using the tool. System performance: Evaluating the operation and efficiency of the tool is a key step in its development. Identifying areas that require more attention or areas that we can build up on help enrich the authoring tool and provide a solid basis to create an advanced product. User satisfaction: Based on actual user experience, this metric is a powerful indicator of how the product might be received and how quickly it might be adopted by users. The test subjects rate and rank different features and functionalities of the tool and we as developers are able to interpret this and change and improve the authoring tool accordingly. Indication of stress and Cognitive Load: The term cognitive load (CL) may be described as the amount of effort that accompanies learning, thinking and reasoning [9] and hence has a bearing on the overall evaluation of the tool.

System performance and user satisfaction: In our usability tests, both these metrics were evaluated from user feedback in the form of questionnaires, user comments and user reactions. Real-time user reactions were also recorded by capturing the screen activity, recording any comments made by the test subjects while doing the tests and by using a webcam to record the activity of the test subjects (see figure 1). Stress and Cognitive Load: As discussed earlier, both stress and cognitive load introduce physiological changes in body, they can be identified using biosensors that monitor and record certain bio-signals. In our usability tests, we monitored the heart rate, skin conductivity and skin temperature of our test subjects.

3. Data Collection

Two questionnaires were administered to the users. The first was used to understand the background of the user and his experience with any of the authoring tools available in the market. This was answered by the test subject before beginning the usability test. The second questionnaire addressing issues related to the ELEPHANT Composer was answered by the test participants after the completion of both the tasks. This one was largely based on the USE Questionnaire for User Interface satisfaction, designed by Arnold Lund [6]. This particular questionnaire evaluates four key factors, Usefulness, Ease of Use, Ease of Learning and Satisfaction, through a series of questions, which are answered by rating (from 1 to 7) between a strongly positive reaction (scored as 7) to a strongly negative one (scored as 1). Test subjects were also given the freedom to express their suggestions and ideas. The test subjects were asked to think aloud and a continuous audio and video recording was made, whereby we could register their thoughts and reactions during the course of the task. In order to correlate these audio comments with the task being performed, the activity on the screen was also captured with the help of Camtasia Studio 5, Screen Recording Software. Using Camtasia we were also able to record the video feed from a webcam that was monitoring the test subject (see figure 1). All these 3 inputs were recorded to be part of the usability test analysis.

In our study, we intended to measure changes in 3 physiological variables, namely heart rate (indicator of stress), skin conductivity (or electrodermal activity [3] - an indicator of CL) and skin temperature (indicator of stress). To carry out these measurements we used two biosensors, the Alive Technologies Heart Monitor and the SenseWear BMS from Body Media. We monitored the bio-signals of the test subjects over both the tasks, allowing us to compare levels of parameters such as CL or stress between the paper-based and tool-based task.

4. Data Interpretation

An initial questionnaire was answered by the test subjects at the start of the test to ascertain the level of computer knowledge and experience with authoring tools and system modeling. Since the test subjects' profession ranged from computer scientists to economists and electrical engineers, we have encountered different levels of both computer knowledge and designing and modeling experience. However, all participants estimated themselves as being capable of operating personal computers, while the self-assessment regarding the experience with software modeling and authoring tools varied quite a lot between the test subjects. We were expecting to see reduced cognitive load for participants with a high level of knowledge regarding software modeling and authoring tools. The second questionnaire (based on the USE Questionnaire for User Interface) was administered after the completion of both the tasks. The second questionnaire was evaluated based on the guidelines as set by the author, and gave us an insight into the levels of user satisfaction and ease of use of the composer. The audio and video recording was evaluated in conjunction with the task that was being performed at that time. The comments made were interpreted along with the activity occurring on the screen and the webcam feed recorded within that time frame, to see what it was about our tool that caused them to have a problem and to see if they had any suggestions to change and improve the tool. As our aim was to analyze the cognitive load (the evaluation of stress is a part of our future work) on the test subjects and depending on the findings, find ways to improve the tool, making it easier to use. To this effect, we analyzed the Galvanic Skin Response (GSR) values tracked by the SenseWear BMS biosensor. We performed a simple statistical analysis, calculating the mean over the entire test duration and over each of the tasks separately. Any task which requires learning, thinking and/or reasoning, puts a certain amount of load on the working memory, known as Cognitive Load (CL) [8]. There are 3 types of CLs associated with learning a task. The intrinsic CL is the inherent difficulty and complexity associated with a task. The extraneous CL is produced based on the manner in which the instruction or information is presented to the student and must be minimized for optimum learning. Finally, the germane CL also originates from the manner of instruction, but contributes towards the learning process [8]. As the number of issues that can be simultaneously handled by the working memory is limited, the Cognitive Load Theory (CLT) provides a basis for designing optimum instructional interfaces which reduces the extraneous CL thereby ensuring more effective learning [7]. A lot of work has been done on using CL to reduce the difficulties associated with learning computer programming which is a highly interactive task. More interaction increases the CL on the working memory as multiple activates and skills are being called upon simultaneously [10]. For tasks rich in interactivity, it is particularly important to reduce the extraneous CL [8]. As in [9] we use the GSR data obtained from our biosensors in order to analyze the effect of CL on our participants, as there is a directly proportional correlation between the GSR values and CL (an increase in CL results in an increase in the GSR [9] and vice versa). Out of the 11 participants, 9 were chosen for the analysis of biosensor data (the data for the other 2 participants was not collected as planned due to problems with improper skin contact).

For the analysis, the entire duration of the test was split up into 3 parts (see figure 3), namely:

- Listening to instructions: where the participants received the initial instructions, including a brief description of the test and the goals
- Paper Based task: where the participant carried out the paper-based task (not time limited) to design a mobile travel assistant on paper
- Computer Based task: where the participants used the ELEPHANT composer to create the same travel assistant

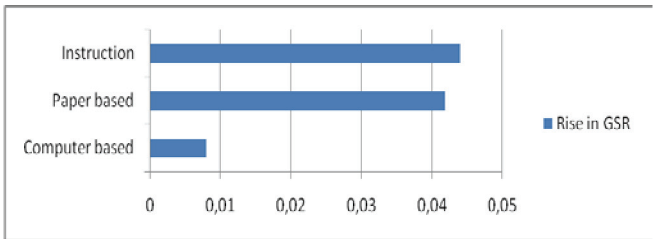


Figure 3: Rise of GSR in µS for participant Banner, opposed for each of the three individual parts (instruction, paper based and computer based)

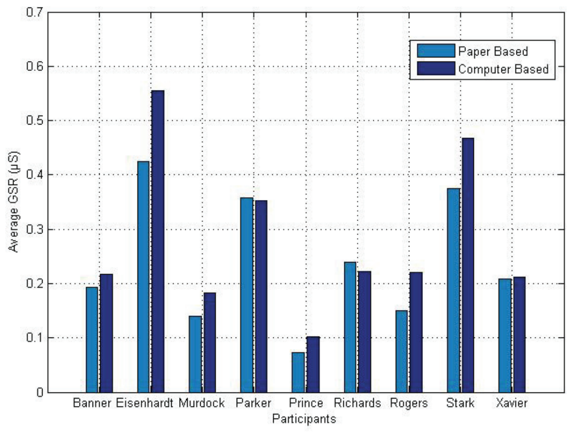


Figure 4: Average GSR for paper based and computer based tasks for each participant

The SenseWear BMS from Body Media provided us with a moving average of GSR for every minute over the entire duration of the test. As each participant spent variable amounts of time on each of the tasks, we calculated the mean GSR for each of the above time intervals for each participant, which allowed us to compare these values.

$$avgGSR_{task}(i) = \frac{\sum GSR_{task}(i)}{t_{task}(i)} \qquad (1)$$

where t_{task} is the duration of each task, i represents the participant and GSR_{task} represents the recorded moving average GSR values for the task being undertaken (listening to the instructions, working on the paper-based, or using the composer). The mean GSR values of the paper-based and computer-based tasks for each of the participants were then compared. Based on these metrics, we present our results in the next section.

5. Conclusion and Future Work

Using the composer people felt comfortable with the system and recommended the quiet simple use of its interface. User-friendliness and the ease of learning were also appreciated by most of the participants. All participants succeeded in searching for resources and arranging them to an expected final structure with marginal variations based on the respective level of creativity and effort put into the application. A limited scale of ELEPHANT elements (E-elements) provided from the system within the testing scenario delimitated freedom of choice. Participants felt restricted of the predetermined set of E-elements. They desired a drilldown of basic E-elements with the possibility to vary these items according to their goals.

Once the mean GSR for each participant for each of the tasks was calculated, we performed the following comparisons to deduce the CL generated in our test subjects, due to using our tool. The average GSR for the 3 tasks of the usability tests were as follows: listening to instructions 0.18 µS, paperbased 0.24 µS and computer based 0.28 µS. As expected, there was an increase in the average GSR for the computer based task, indicating an increase in the CL. This clearly supports the theory that moving from a known environment (paper based) to an unknown environment (the ELEPHANT Composer) which involves the usage of a new computer tool causes a rise in the cognitive load on the memory. The next step was to examine the average GSR for each of the participants individually. As we are specifically interested in the paper based and computer based tasks, figure 4 plots the average GSR calculated for each participant in these 2 tasks. In order to see the significance of the change (increase or decrease), we also calculated the change in the average GSR in the computer based task with respect to that of the paper based task and expressed it as a percentage.

$$Change\ \% = \frac{\underline{avgGSR_{computer}(i) - avgGSR_{paper}(i)}}{avgGSR_{paper}(i)} \times 100 \qquad (2)$$

where *i* is represents each participant. While the general trend is to have an increase in the GSR (and hence an increase in CL), we observed that for 2 participants (Richards and Parker) there was a decrease in the GSR recorded during the computer based test. Comparing the GSR results with those of the questionnaires, we saw that Richards and Parker, both hailing from background of IT and with extensive computer expertise and experience in using authoring systems found our tool easy to use and were able to learn the use of it quickly. This was expected, as we have already noticed the test subjects' varying knowledge level in software modeling and authoring, as pointed out above. The CL that was exerted on their working memories reduced during the computer based task.

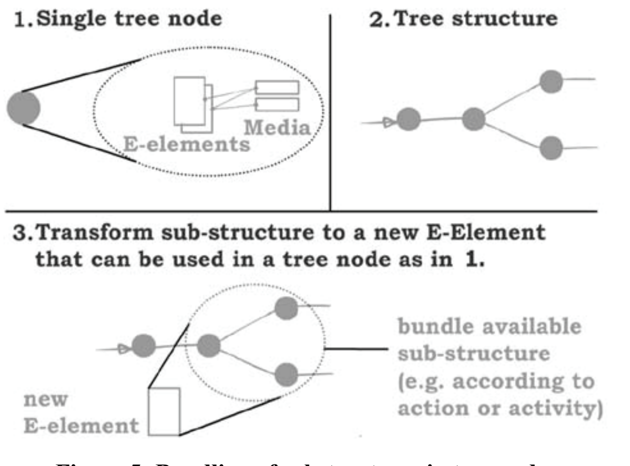

1. Single tree node
E-elements
Media
2. Tree structure
3. Transform sub-structure to a new E-Element that can be used in a tree node as in 1.
new E-element
bundle available sub-structure (e.g. according to action or activity)

Figure 5: Bundling of substructures in tree nodes

In [1] we defined an ELEPHANT element (E-element) as a component with application logic. E-elements could only be developed by software engineers or designers with scripting abilities. We are planning to allow that new E-elements can also be composed with the E-Composer (see figure 5). With this improvement, the modeling based on components becomes more flexible but still keeps the high level. Because of the flexibility we gain, we also approach our long term goal of supporting activity-based design. Activities are dynamic and hierarchical structures. In activity theory, the objective of an activity can be realized through different sets of actions [5], different people might need different actions for the same activity and hence different ways to model the assistance for the same activity. Same actions can contribute to different activities, and may also have different meanings for the people undertaking them [4].

6. ACKNOWLEDGMENTS

This work was funded in part by the Bavarian Ministry of Economic Affairs, Infrastructure, Transport and Technology within the project „Dynamische Plattformen für Verteilte Systeme“.

7. REFERENCES

[1] I. Aslan and D. Menon. Component-based development of mobile assistants with the elephant system. In Proceedings of Mobility 2009, Nice, France, September 2-4, 2009.

[2] Elliot, S. N. et al., Cognitive load theory and universal design principles: Applications to test item development, Vanderbilt University, NASP Session, 2009

[3] Haag, A., Goronzy, S., Schaich, P., and Williams, J. Emotion recognition using bio-sensors: First steps towards an automatic system.2004, pp. 36-48.

[4] K. Kuutti. Activity theory as a potential framework for human-computer interaction research. In Context and Consciousness: Activity Theory and Human-computer Interaction, pages 17–44, 1996.

[5] A. Leont'ev. Activity, Consciousness, and Personality. Prentice Hall, New Jersey, 1978.

[6] Lund, A. Measuring usability with the use questionnaire, stc usability sig newsletter, 8:2.

[7] Oviatt, S., Human-Centred Design meetns Cognitive Load Theory: Designing interfaces that help people think, Proceedings of the 14th annual ACM international conference on Multimedia, 2006, pp. 871-880

[8] Richard E. M., The Cambridge handbook of multimedia learning, Cambridge University Press, 2005

[9] Shi, Y., Ruiz, N., Taib, R., Choi, E., and Chen, F. Galvanic skinresponse (gsr) as an index of cognitive load. In CHI '07: CHI '07 extended abstracts on Human factors in computing systems (New York, NY, USA, 2007), ACM, pp. 2651-2656.

[10] Yousoof, M., Sapiyan, M., and Kamaluddin, K., Reducing cognitive load in learning computer programming, World Academy of Science and Technology, Volume 12, 2006, ISSN 1307-6

End-User Customization of Multi-Device Ubiquitous User Interfaces

Fabio Paternò, Giuseppe Zichitella
HIIS Laboratory – CNR-ISTI
Via Moruzzi 1, 56124 Pisa, Italy
{fabio.paterno, giuseppe.zichitella}@isti.cnr.it

ABSTRACT
In this paper we discuss an approach to supporting end-users in customizing multi-device ubiquitous user interfaces. In particular, we show a tool allowing end-users to customize desktop-to-mobile adaptation by exploiting model-based descriptions in the MARIA language. Some results are presented along with indications for future work.

Author Keywords
End-user Development, Ubiquitous Applications, Multi-Device Environments, Adaptation.

ACM Classification Keywords
H.5.2 User Interfaces.

INTRODUCTION

One of the main issues in current technological settings is how to design and develop interactive applications that can be accessed through a wide variety of devices (ranging from small watches to very large screens, including various types of smartphones, PDAs and Digital TVs). This is particularly important in Web application, which are the most common applications.

One important research area in this context is the model-based approach, in which declarative descriptions of the user interface are used in order to avoid dealing with a plethora of low-level implementation details associated with the wide number of available devices and implementation languages. Despite such potential benefits, its adoption has mainly been limited to professional designers, but new solutions are recently emerging that are able to extend such approaches in order to achieve natural development by enabling end-users to develop or modify interactive applications still using conceptual models, but with continuous support that facilitates their development, analysis, and use [1].

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

End-User Development [3] (EUD) can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact. End-users have already difficulties with single device applications, thus it is easy to understand how such difficulties increase when considering applications for multi-device environments. This is one further reason for providing better support for EUD in ubiquitous applications.

The vision of ubiquitous computing [9] is that the users operate in intelligent environments, which are aware of users' needs and able to assist, even proactively, the users in performing their activities and reaching their goals. To this end, one important aspect is the possibility for a user surrounded by multiple devices to freely move about and continue the interaction with the available applications through a variety of interactive devices. Indeed, in such environments one big potential source of frustration is that people have to start their session over again from the beginning at each interaction device change. Continuous task performance implies that interactive applications be able to follow users and adapt to the changing context of use while preserving their state. Thus, migratory user interfaces require integrated solutions able to address state persistence and user interface adaptation when the user changes the device.

Model-based languages are utilized at design time to help the user interface designer cope with the increasing complexity of today's applications and contexts. The underlying user interface models are mostly used to generate a final user interface code, which is then executed at run time. Nevertheless, approaches utilizing the models at run time are receiving increasing attention. We agree with Sottet et al. [8], who call for keeping the models alive at run time to make the design rationale available and show a solution for this purpose.

In the following we present some research work that exploits model-based approaches for multi-device ubiquitous applications. We show how we have enriched a software model-based platform for migratory user interfaces with a new tool for desktop-to-mobile adaptation, called parametric bidimensional semantic redesign. One of its features is that it allows the end-users to customize the adaptation process. We present some initial results and then discuss how we plan to extend them.

MIGRATORY USER INTERFACES

Migration is the result of two main features: state persistence across multiple devices and adaptation to the device interaction resources. They have to be supported while users interact with the applications made available by the intelligent environment. For this purpose, we have designed and developed a migration architecture [5], which supports a number of reverse and forward transformations that are able to transform existing desktop Web applications for various interaction platforms and support task continuity. The basic assumption is that there exists a huge amount of easily accessible content for desktop Web applications, which can be processed and transformed to support migratory interfaces, even across non-Web implementation languages. The advantage of this solution with respect to others (e.g. [4]) is that it does not require that the applications be implemented using a particular toolkit in order to make them able to migrate.

In this environment the client devices subscribe to the migration service by running a migration client that provides the environment with information regarding the device characteristics. The devices access Web applications through the migration server, which includes proxy functionalities. Migration can be triggered either by the user or it can be automatically triggered by the smart environment when some specific event (such as very low battery or connectivity) is detected, or in a mixed solution in which the environment suggests possible migrations and the user decides whether or not to accept them.

When the user accesses the application through an interaction platform other than the desktop, the server transforms its user interface by building the corresponding logical description and using it as a starting point for creating the implementation adapted to the accessing device. Figure 1 shows how adaptation is obtained. There are three main phases: reverse engineering, semantics-based adaptation, and generation. In the first phase, the tool automatically builds the logical description of the accessed page. It has rules able to handle HTML and CSS tags and associate them with the corresponding logical elements. For example, if DIV, or FIELDSET or IFRAME tags are used then it recognises that there is a group of logically connected elements in the page. We call the adaptation module semantic redesign since its purpose is to change the design still considering the interaction semantics of the implementation elements that are specified in the corresponding logical description. In addition to interface adaptation, the environment supports task continuity. To this aim, when a request for migration to another device is triggered, the environment also takes the state of the user interface, which depends on the user input (elements selected, data entered, …) and identifies the last element accessed in the source device. Thus, when a logical version of the interface for the target device is generated, it also contains the state detected in the source device version so that the user inputs (selections performed, data entered, …) are not lost. In the last phase, the user interface implementation for the target device is generated and activated remotely at the point corresponding to the last basic task performed in the initial device.

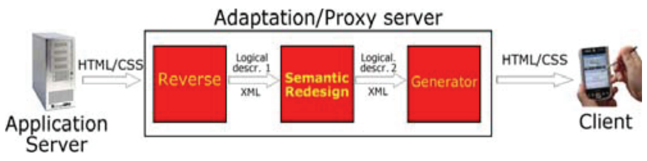


Figure 1. The main phases of the adaptation process.

In the process of creating an interface version suitable for a platform different from the desktop, we use a semantic redesign module. This part of the migration environment automatically transforms the logical description of the desktop version into the logical description for the new platform. Therefore, the goal of this transformation is to provide a description of the user interface suitable for the new platform. This means that intelligent rules are used for adapting the description of the user interface to the new platform taking into account its capabilities (e.g.: using interface elements that are more suitable for the new platform) but ensuring at the same time that the support for the original set of tasks is maintained. This solution allows the environment to exploit the semantic information contained in the logical description. In this case the semantic information is related to the basic tasks that the user interface elements are expected to support.

This software architecture for migratory user interfaces currently uses MARIA [7], a recent model-based language, which allows designers to specify abstract and concrete user interface languages according to the CAMELEON Reference framework [2]. This language represents a step forward in this area because it provides abstractions also for describing modern Web 2.0 dynamic user interfaces and Web service accesses. In its first version it provides an abstract language independent of the interaction modalities and concrete languages for graphical desktop and mobile platforms. In general, concrete languages are dependent on the typical interaction resources of the target platform but independent of the implementation languages.

In MARIA an abstract user interface is composed of one or multiple presentations, a data model, and a set of external functions. Each presentation contains a number of user interface elements (interactors) and interactor compositions (indicating how to group or relate a set of interactors), a dialogue model describing the dynamic behaviour of such elements, and connections indicating when a change of presentation should occur. The interactors are classified in abstract terms: edit, selection, only_output, control, interactive description, etc.. Each interactor can be associated with a number of event handlers, which can change properties of other interactors or activate external functions.

END-USER ADAPTATION CUSTOMIZATION

In the research on migratory user interfaces, one issue that we are considering is how to provide users with more control on the migration process in order to improve its usability. In this context more control can mean various things. One important aspect is control on the rules that drive adaptation to the various platforms (the most common case is desktop-to-mobile adaptation). For example, the adaptation engine is able to split the desktop pages when they require considerable amount of interaction resources but some users may like to have more control on the splitting algorithm.

In particular, we have designed a new tool for adaptation: *Parametric Bidimensional Semantic Redesign*. It supports adaptation from desktop-to-mobile devices and overcomes limitations of previous approaches in the area [6] because it allows users to configure the adaptation process and provides more control on costs calculation and the adaptation results. For example, while previous solutions calculated the screen space requested by the user interface elements mainly in terms of its vertical use, the new algorithm calculates both the horizontal and the vertical consumption of screen space.

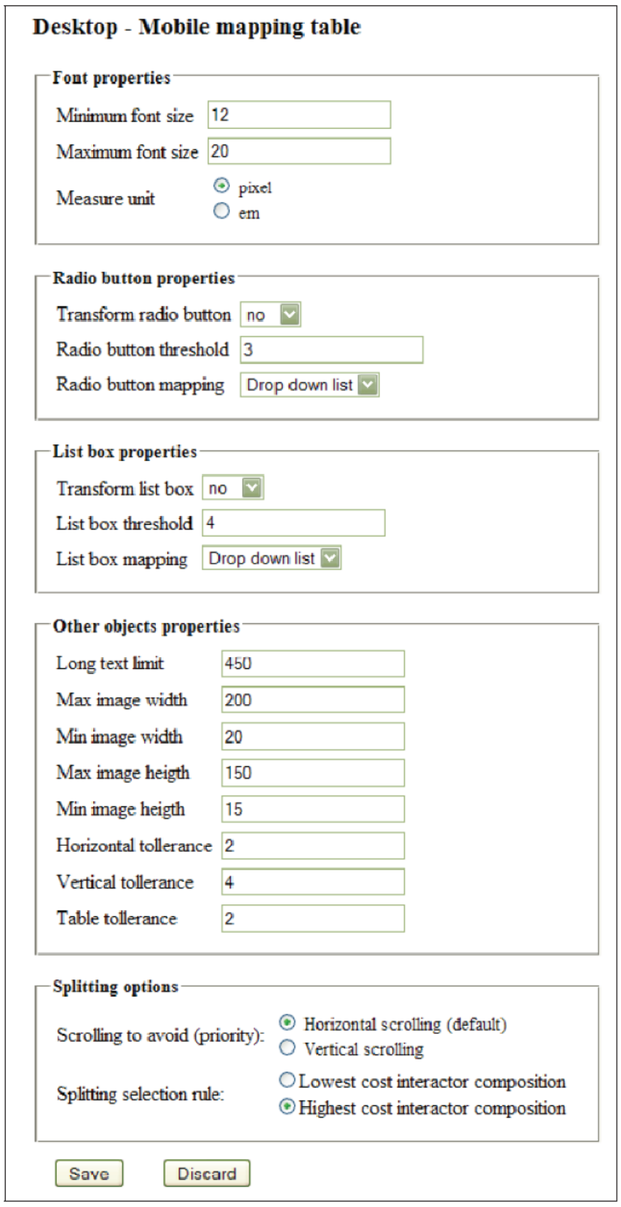
The adaptation tool takes as input the concrete description of a desktop user interface in the MARIA language and goes through a number of steps. For each step a number of specific rules are applied. First, it performs some basic transformations: if the user provides preferences regarding the minimum and maximum fonts for the target device then it transforms all the textual content in order to fit in the given range. Next, it calculates the cost of all the interactors and composition operators in the provided specification. If the resulting total cost is sustainable for the target device then the corresponding logical description is generated otherwise it starts the process to reduce the cost in order to make it sustainable. First, basic elements are adapted for the target device: the images are reduced in space while preserving their aspect ratio, some interactors are replaced with others that are semantically equivalent but needs less screen space, long texts are reduced in such a way that the part exceeding a limit is shown only on request, image and text in tables are reduced in size. After these basic transformations the overall cost is calculated again and if it is not yet sustainable by the target device then the part related to page splitting is activated. The purpose of this phase is to split the original desktop presentation into two or more presentations, which are sustainable for the target mobile device. For this purpose the algorithm considers the interactor compositions, and associates some of them to newly generated mobile presentations, removing them from the current presentation in order to decrease its overall cost.

The elements that determine the cost of the interactors are: the font attributes (size, style, type), the vertical and horizontal space required by a text, image dimensions, interline value, interactor type, and so on

Figure 2 shows the user interface that allows end users to configure the adaptation process. The various parameters are grouped according to the related user interface aspect considered. For the fonts, it is possible to specify the minimum and maximum font in the target device, and the associated measure unit. For the radio buttons it is possible to indicate whether they should be transformed into an interactor that supports the same semantics but with using less space screen. In this case, it is possible to specify the threshold, in terms of number of choice options, which should trigger the transformation and the type of interactor to use for its replacement. Similar parameters are available for the list boxes. Other parameters concern the maximum number of characters for a text, maximum and minimum dimensions for images. These parameters determine the cost of rendering a presentation. This cost is compared with the overall sustainable cost in the target device, which is given by the screen resolution multiplied by horizontal and vertical tolerance. The higher the tolerance coefficient values are, the more scrollable the generated user interface will be. This means that end users have the possibility to specify to what extent the adapted content will be scrollable in the target device. The table tolerance provides an additional factor to consider when calculating the sustainable cost. In practise, this means that when there are tables, more scrolling will be acceptable before deciding to split the presentation.

The customization interface also allows the user to indicate two additional parameters: what type of scrolling (horizontal or vertical) to avoid has the priority, and the splitting algorithm version to apply. Indeed, the tool supports two ways to determine how splitting should be performed. In both cases it analyses the cost of the composition operators (grouping or relation), which includes those of the composed interactors, and the cost of the tables (both data and layout tables). Then, the decision of the set of elements to allocate to the newly generated mobile presentation is given in one case by the most expensive element. In the other case the algorithm first calculates what elements are able to make the current presentation sustainable by the target device if removed, and then selects among them the one that has the lowest cost. The rationale for this second option is that it allows users to obtain a sustainable presentation by removing the least amount of information possible, thus preserving as much as possible the original design.

In terms of results of the adaptation process we have conducted a study comparing our tool with two publicly available tools for desktop-to-mobile adaptation: Mowser (http://mowser.com) and Skweezer (http://www.skweezer.com). The results were encouraging because our tool has shown to be more flexible since it allows end users to customize the adaptation parameters and is able to adapt a higher number of types of interface elements than the other two tools (e.g. tables and long texts do not receive specific adaptation transformations with the other two tools).



Desktop - Mobile mapping table

Font properties

Minimum font size	12
Maximum font size	20
Measure unit	[✓ pixel] em

Radio button properties

Transform radio button	no
Radio button threshold	3
Radio button mapping	Drop down list

List box properties

Transform list box	no
List box threshold	4
List box mapping	Drop down list

Other objects properties

Long text limit	450
Max image width	200
Min image width	20
Max image heigth	150
Min image heigth	15
Horizontal tollerance	2
Vertical tollerance	4
Table tollerance	2

Splitting options

Scrolling to avoid (priority):	[✓ Horizontal scrolling (default)] Vertical scrolling
Splitting selection rule:	Lowest cost interactor composition [✓ Highest cost interactor composition]

Figure 2. The customization user interface.

CONCLUSIONS

Ubiquitous environments call for adaptive systems in order to adapt to the varying interaction resources. Model-based approaches can provide useful support in this context. However, there is a need for providing users with more control on ubiquitous interfaces, according to the end-user development paradigm.

In this paper we have presented first results that allow end-users to customize desktop-to-mobile adaptation in order to change the results that can be obtained by automatic user interface generation.

We plan to further extend this work in various directions. The customization user interface can be improved in order to make the effects of the various customization parameters more understandable. In addition, in this work we have considered only desktop-to-mobile adaptation but other types of transformations can benefit from the approach proposed, e.g. graphical-to-vocal adaptation.

ACKNOWLEDGMENTS

This work has been supported by the EU project OPEN (http://www.ict-open.eu/)

REFERENCES

1. Berti, S., Paternò, F., Santoro C., "Natural Development of Ubiquitous Interfaces", Communications of the ACM, September 2004, pp.63-64, ACM Press.
2. Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., and Vanderdonckt, J. 2002. The CAMELEON reference framework. CAMELEON Project. Deliverable 1.1
3. Lieberman, H., Paternò, F., Wulf W. (eds), End-User Development, Springer Verlag, ISBN-10 1-4020-4220-5, 2006.
4. Melchior, J., Grolaux, D.,Vanderdonckt, J.,Van Roy, P., A Toolkit for Peer-to-Peer Distributed User Interfaces: Concepts, Implementation, and Applications, pp. 69.78, EICS'09, July 15–17, 2009, Pittsburgh, Pennsylvania, USA.
5. Paternò, F., Santoro, C., Scorcia, A., Ambient Intelligence for Supporting Task Continuity across Multiple Devices and Implementation Languages, the Computer Journal, the British Computer Society, 2009.
6. Paternò, F., Santoro, C., Scorcia A Automatically Adapting Web Sites for Mobile Access through Logical Descriptions and Dynamic Analysis of Interaction Resources. AVI 2008, Naples, May 2008, ACM Press, pp. 260-267.
7. Paternò F., Santoro C., Spano L.D., "MARIA: A Universal Language for Service-Oriented Applications in Ubiquitous Environment", ACM Transactions on Computer-Human Interaction, Vol.16, N.4, November 2009, pp.19:1-19:30.
8. Sottet J., Ganneau V., Calvary G., Coutaz J., Demeure A., Favre J., Demumieux R.: Model-Driven Adaptation for Plastic User Interfaces. INTERACT (1) 2007: 397-410.
9. Weiser M., "The Computer for the 21st Century" - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991.

Aspect-Oriented UI Modeling with State Machines

Gefei Zhang[*]
Ludwig-Maximilians-Universität München
gefei.zhang@pst.ifi.lmu.de

ABSTRACT
Separated modeling of User Interface (UI) widgets is a very natural way to tackle the complexity of UI models. Due to interactions between widgets, however, this is not always an easy task. We propose an aspect-oriented approach to widget-oriented UI modeling: each widget's behavior is modeled separately in a UML state machine; synchronization of the state machines is modeled in aspects and is woven into the widget models automatically. The weaving process is transparent to the modeler. This way, we can strongly increase the degree of separation of concerns in UI modeling and reduce the complexity of UI models.

Keywords
UI modeling, UML, State machine, Aspect-oriented Modeling

1. INTRODUCTION
Modern User Interfaces (UI) are mostly interactive: the widgets are no longer supposed to be sheerly receiving input from the user or presenting the data of the system, but may also have inner lives themselves. They may have their own states, trigger events, and call other widgets to execute certain behaviors. For instance, besides acting as an input terminal for strings, a text field is more often than not supposed to be able to automatically fill in its value, or trigger database queries and fill in other widgets.

Separation of concerns in modeling such rich UI is challenging. On the one hand, it is appealing to model the widgets separately from each other, each in its own model. On the other, the synchronization of the widgets' behaviors obviously cross-cuts the widgets. An example is the requirement that only one of the widgets in a window be focused at a given time. Modeling such requirements often torpedos the natural, widget-based separation of concerns. As a result, the complexity of UI models may increase rapidly as soon as the UI gets non-trivial.

We propose an aspect-oriented approach to rich-UI modeling. Our approach enables separation of widget modeling. Each widget is modeled in its own UML state machine [11], separated from other widgets. We call the state machines *widget machines*. Necessary synchronization between widgets, if any, is left out of the widget machines, which keeps them rather simple. The synchronization is then modeled using aspect-oriented techniques: we define *aspects*, separately from the widget machines, to define constraints on or additional behaviors of the widgets. The overall behavior of the UI is then obtained by the composition of the widget machines and the aspects. We refer to the composition process as *weaving*.

In our approach, the complexity of widget synchronization is hidden behind weaving, which is transparent to the modeler. This fact and the increased separation of concerns make our approach easy to use and reduce the complexity of rich-UI models considerably.

The remainder of this paper is organized as follows: in the following Sect. 2 we present our modeling approach, including separate modeling of the widgets and aspect-oriented modeling of their synchronization; in Sect. 3 a brief discussion is given on how the aspects are woven to the state machines. Related work is discussed in Sect. 4, some concluding remarks, as well as an outline of our future research, are given in Sect. 5.

2. MODELING APPROACH
Our modeling approach is very simple. It contains three steps:

1. Construct a top-level state machine to model the basic control flow of the application, and use submachine states as place holders for widgets.
2. Model the behaviors of the widgets and complete the submachines, without considering inter-widget synchronization.
3. Define necessary aspects to synchronize the widgets.

We demonstrate this approach by means of a simple address book application. The application should provide two windows: the first containing a list of the names of all contacts and a button to show the second view, the second containing text fields for inputting data of a new contact. The second window is supposed to have a rich UI.

2.1 Top-level state machine
The first step is to model the top-level widgets, i.e., the two windows, see Fig. 1. When the application is started, the list of all contacts (`ContactList`, details ignored in this paper) is presented. The user can select to add a new contact (`newContact`), and then enter the contact details in `NewContact`. This window is supposed to have nine widgets: four input fields, four labels of the input fields, and an `OK` button to finish the input.

[*]Partially supported by the DFG Project MAEWA (WI 841/7–2)

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

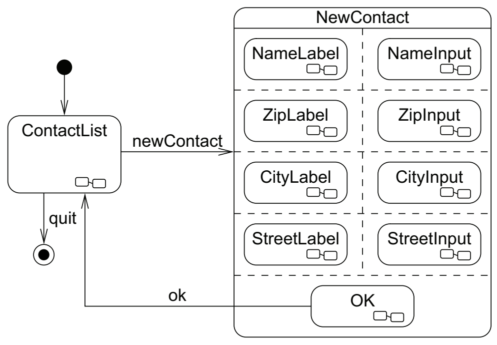


Figure 1: Top-level state machine

2.2 Widgets in Separation

The details of the widgets' behaviors are modeled separately in the sub-machines. In this step, synchronization of widgets is not considered, which simplifies the widget modeling considerably.

In our address book example, the labels have a very simple behavior: they display some predefined text, and do not react to any event. This behavior is modeled with a state machine given in Fig. 2. It contains only one state `Show`, presenting the label showing its caption.

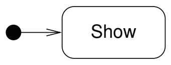


Figure 2: State machine for the label widgets

We ignore in this paper the state machine of `OK`, which is also very simple. More interesting are the state machines of the text input widgets. Figure 3 models the behavior of `NameInput`: it may be either unfocused (`NoFocus`) or focused (`WaitForInput`). If it is focused it is ready for user input (`input(t)`), and updates its text (`text = text + t`) upon each input; otherwise it does not react to any event.

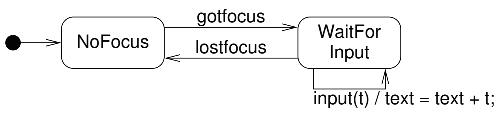


Figure 3: State machine for `NameInput`

The state machines for the other three input widgets are slightly more complex, see Fig. 4. Additionally to the behavior of `Name-Input`, the other three input widgets also send the current text (`push(text)`) to whomever it concerns, and are, focused or not, ready to receive a call back (`pull(newText)` from whomever and to update the text (`text = newText`). This additional feature models the capability of automatic completion of one widget (e.g. `CityInput`) by another (e.g. `ZipInput`). Usually, getting a Zip code is only possible from a combination of a city and a street. We ignore such details here and assume they are implemented correctly in `pull` and `push`.

2.3 Synchronization by aspects

The state machines so far are simple because they do not include synchronization with each other, which is usually necessary in a rich UI. For example, the input widgets obviously are not supposed to be in state `WaitForInput` simultaneously. Modeling such synchronization in the widget machines would break the separation of concerns, therefore we source them out and model them in aspects.

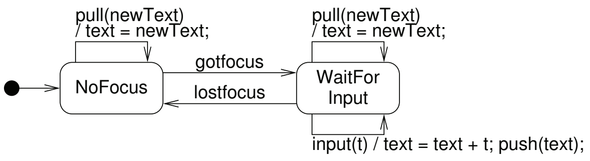


Figure 4: State machines for `ZipInput`, `CityInput` and `StreetInput`

An aspect is in our approach a first-class model element. It contains a restriction to or an extension of the behavior defined in some widget machine. For instance, we model the requirement that only one widget is supposed to be in `NoFocus` by two aspects:

```
(aspect non-simultaneous
   (mutual-exclusion
      (transition NoFocus WaitForInput)))
(aspect send-others-away
   (before WaitForInput)
   (scope except-me (goto NoFocus)))
```

where the aspect `non-simultaneous` defines a restriction: only one submachine (keyword `mutual-exclusion`) is allowed to fire the transition from `NoFocus` to `WaitForInput` (keyword `transition`) at a given time. This aspect prevents the widget machines from transitioning from `NoFocus` to `WaitForInput` at the same time. The aspect `send-others-away` defines an additional behavior of the input widgets, to be executed just before (keyword `before`) state `WaitForInput` gets active: tell the others (`scope except-me`) to go to state `NoFocus` (`goto NoFocus`). These two aspects thus models the above synchronization rule concisely and separately from the widget machines.

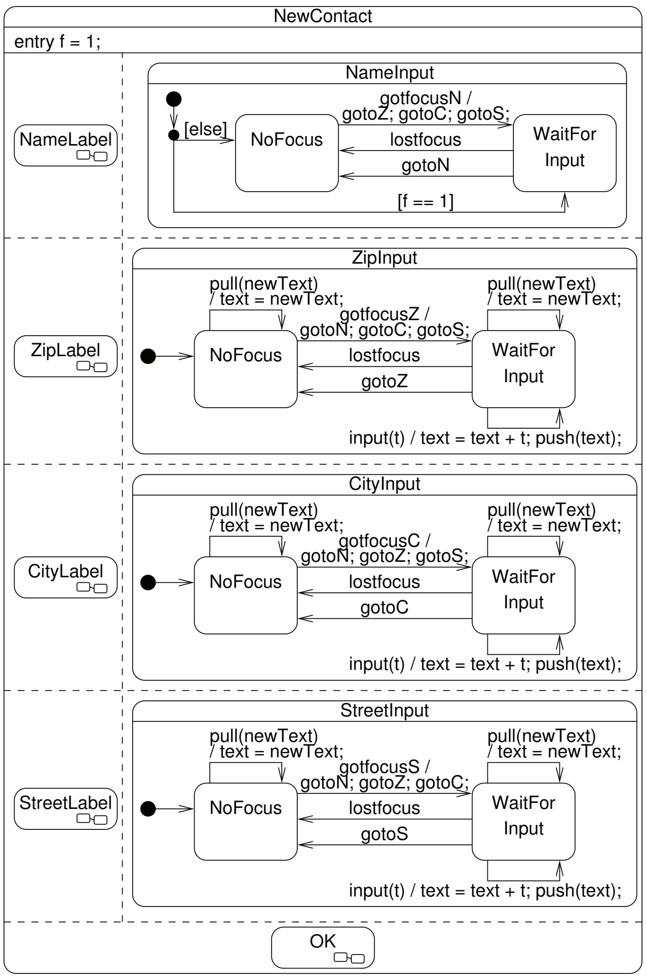
Note that using these two aspects is not the only way of preventing the input widgets from being in `WaitForInput` simultaneously. A direct definition of mutual exclusion of states is also possible. Actually, such an aspect would be implemented as a combination of the two above aspects. We decided to use the more detailed aspects, since they are closer to the weaving (see below).

Another synchronization requirement in our sample application is that when the window `NewContact` is shown, `NameInput` should be the focused widget, i.e. in the state `WaitForInput`. We model this with the following aspect `has-focus`, which tells the sub-machine `NameInput` (by `scope (NameInput)`) to goto state `WaitForInput` just after `NewContact` gets active (`after NewContext.enter`).

```
(aspect has-focus
  (after NewContact.enter)
  (scope NameInput (goto WaitForInput)))
```

3. WEAVING

As simple as the aspects are, the implementation of the synchronization requires rather complex modification to the widget models.



NewContact
entry f = 1;
NameLabel
NameInput
gotfocusN / gotoZ; gotoC; gotoS;
[else]
NoFocus
lostfocus
gotoN
WaitFor Input
[f == 1]
ZipLabel
ZipInput
pull(newText) / text = newText;
gotfocusZ / gotoN; gotoC; gotoS;
gotoZ
input(t) / text = text + t; push(text);
CityLabel
CityInput
gotfocusC / gotoN; gotoZ; gotoS;
gotoC
StreetLabel
StreetInput
gotfocusS / gotoN; gotoZ; gotoC;
gotoS
OK

Figure 5: Partial weaving result of the sample application

This modification is taken care by an automatic weaving process, which is still ongoing work. With the automatic weaving, the aspects will be composed with the base machine "off stage", i.e., the modeler is refrained from the cumbersome details. We explain our weaving by means of the weaving result of the above aspects, see Fig. 5.

Mutual exclusion of transitions is implemented by a static renaming the events of the transitions, so that the transitions are no longer enabled at the same time. In Fig. 5, aspect `non-simultaneous` is therefore implemented by renaming the event `gotfocus` in the widget models to `gotfocusN`, `gotfocusZ`, `gotfocusI`, and `getfocusS`.

Generally, `before X` and `after X` are woven by intercepting all transitions leading to and leaving state `X`, respectively; `goto X` is woven by introducing a new transition to `X` and sending a signal to the respective state machine to fire that transition. In Fig. 5, aspect `send-others-away` is implemented as an additional transition in the widget machines from `WaitForInput` to `NoFocus`, triggered by a uniquely named event (`gotoN`, `gotoZ`, `gotoC`, `gotoS`), and an effect of the transition from `NoFocus` to `WaitForInput`, firing the "right" events.

One of the (many) exceptions to the above general rule is `after X.enter`. Obviously an interception to all transitions leaving `X` does not help in this case. Therefore, we implement `after X.enter` by introducing an entry action to `X`. In Fig. 5, aspect `has-focus` is implemented by `NewContact`'s entry action, setting `f` to `1`, and splitting the transition leaving `NameInput`'s initial vertex to active `WaitForInput` immediately if `f == 1`.

A brief glance at Fig. 5 suggests how cumbersome modeling widget synchronization may get. In comparison, modeling with our aspects is simple and straight-forward. All the complexity of actually implementing the required synchronization is hidden behind the weaving (once implemented) and invisible to the modeler.

4. RELATED WORK

Model driven development is a promising paradigm for UI development. There are several proposals of UI modeling, see [1, 5, 6, 10, 12, 17]. In particular, state machines are also used in [14, 16], where the former work describes a translation of Concurrent Task Tree (CTT) models into UML state machines, and the latter defines an extension of UML state machines to model navigation of web applications. Compared with these approaches, the distinguishing feature of our approach is its use of aspect-oriented modeling (AOM) to model synchronization of state machines. This makes the UI models of our approach easy to construct and easy to use, since the cumbersome details of interaction between widgets are hidden behind a (yet-to-implement) automatic weaving process.

AOM was also applied to reduce the complexity of design models in other application areas, such as adaptive systems [2, 13, 18] or crisis management systems [7], see in [4] for a more general overview of aspect-oriented techniques. Compared with other proposals of aspect-oriented state machines, such as [15, 21], the aspect language used in this paper is *high-level* in the sense that it is used to define modifications of behaviors on a more abstract level than (syntactical) modifications of modeling elements, see [19, 20] for a more thorough dicussion on the advantages of high-level aspect-oriented modeling.

5. CONCLUSIONS AND FUTURE WORK

We presented a widget-oriented modeling approach for interactive user interfaces. Our approach uses UML state machines, a very popular language for modeling software behaviors. By supporting aspect-oriented modeling our approach achieves a high degree of separation of concerns, and thus increases the feasibility of widget-oriented UI modeling considerably.

We plan to integrate the aspect language into HiLA[1], our general approach to aspect-oriented state machines. Using state machines as the modeling language, and in particular the definition the weaving result in the form of a state machine, makes it possible to verify the weaving result by formal methods like model checking or theorem proving. In particular, we plan to apply the UML model checker Hugo/RT [8] to verify temporal logical properties of our UI models.

6. REFERENCES

[1] Goetz Botterweck. A Model-Driven Approach to the Engineering of Multiple User Interfaces. In Thomas Kühne, editor, *Reps. and Rev. Sel. Papers Wshs and Symp. at*

[1]`http://hila.pst.ifi.lmu.de`

MoDELS'06, volume 4364 of *Lect. Notes in Comp. Sci.*, pages 106–115. Springer, 2007.

[2] Sven Casteleyn, William Van Woensel, and Geert-Jan Houben. A Semantics-based Aspect-oriented Approach to Adaptation in Web Engineering. In Simon Harper, Helen Ashman, Mark Bernstein, Alexandra I. Cristea, Hugh C. Davis, Paul De Bra, Vicki L. Hanson, and David E. Millard, editors, *Proc. 18th ACM Conf. Hypertext and Hypermedia (HYPERTEXT'07)*, pages 189–198. ACM, 2007.

[3] Gregor Engels, Bill Opdyke, Douglas C. Schmidt, and Frank Weil, editors. *Proc. 10th Int. Conf. Model Driven Engineering Languages and Systems (MoDELS'07)*, volume 4735 of *Lect. Notes Comp. Sci.* Springer, 2007.

[4] Robert E. Filman, Tzilla Elrad, Siobhán Clarke, and Mehmet Aksit, editors. *Aspect-Oriented Software Development*. Addison-Wesley, 2004.

[5] Guillaume Gauffre, Emmanuel Dubois, and Rémi Bastide. Domain-Specific Methods and Tools for the Design of Advanced Interactive Techniques. In Holger Giese, editor, *Reps. and Rev. Sel. Papers Wshs and Symp. at MoDELS'07*, volume 5002 of *Lect. Notes in Comp. Sci.*, pages 65–76. Springer, 2008.

[6] Daniel Görlich and Kai Breiner. Useware Modeling for Ambient Intelligent Production Environments. In Andreas Pleuß, Jan Van den Bergh, Heinrich Hußmann, Stefan Sauer, and Daniel Görlich, editors, *Proc. Workshop Model Driven Development of Advanced User Interfaces (MDDAUI'07)*, volume 297 of *CEUR Workshop Proceedings*. CEUR, 2007.

[7] Matthias Hölzl, Alexander Knapp, and Gefei Zhang. Modeling the Car Crash Crisis Management System with HiLA. *Trans. Aspect-Oriented Software Development (TAOSD)*, 7, 2010. Accepted.

[8] Alexander Knapp, Stephan Merz, and Christopher Rauh. Model Checking Timed UML State Machines and Collaborations. In Werner Damm and Ernst Rüdiger Olderog, editors, *Proc. 7th Int. Symp. Formal Techniques in Real-Time and Fault Tolerant Systems*, volume 2469 of *Lect. Notes Comp. Sci.*, pages 395–416. Springer, 2002.

[9] Gerrit Meixner, Daniel Görlich, Kai Brainer, Heinrich Hußmann, Andreas Pleuß, Stefan Sauer, and Jan Van den Bergh, editors. *Proc. 4th Wsh. Model Driven Development of Advanced User Interfaces (MDDAUI'09)*, volume 439 of *CEUR Workshop Proceedings*. CEUR, 2009.

[10] Gerrit Meixner, Marc Seissler, and Marcel Nahler. Udit—A Graphical Editor for Task Models. In Meixner et al. [9].

[11] Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.2. OMG Available Specification, OMG, 2009. `http://www.omg.org/spec/UML/2.2/Superstructure`.

[12] Andreas Pleuß, Arnd Vitzthum, and Heinrich Hußmann. Integrating Heterogeneous Tools into Model-Centric Development of Interactive Applications. In Engels et al. [3], pages 241–255.

[13] Andrea Schauerhuber. *aspectUWA: Applying Aspect-Orientation to the Model-Driven Development of Ubiquitous Web Applications*. PhD thesis, Technische Universität Wien, 2007.

[14] Jan Van den Bergh and Karin Coninx. From Task to Dialog Model in the UML. In Marco Winckler, Hilary Johnson, and Philippe A. Palanque, editors, *Proc. 6th Int. Wsh. Task Models and Diagrams for User Interface Design (TAMODIA'07)*, volume 4849 of *Lect. Notes Comp. Sci.*, pages 98–111. Springer, 2007.

[15] Jon Whittle, Ana Moreira, João Araújo, Praveen K. Jayaraman, Ahmed M. Elkhodary, and Rasheed Rabbi. An Expressive Aspect Composition Language for UML State Diagrams. In Engels et al. [3], pages 514–528.

[16] Marco Winckler and Philippe A. Palanque. StateWebCharts: A Formal Description Technique Dedicated to Navigation Modelling of Web Applications. In Joaquim A. Jorge, Nuno Jardim Nunes, and João Falcão e Cunha, editors, *Proc. 10th Int. Wsh. Design Specification and Verification of Interactive Systems (DSV-IS'03)*, volume 2844 of *Lect. Notes Comp. Sci.*, pages 61–76. Springer, 2003.

[17] Andreas Wolff and Peter Forbrig. Deriving User Interfaces from Task Models. In Meixner et al. [9].

[18] Gefei Zhang. Aspect-Oriented Modeling of Adaptive Web Applications with HiLA. In Gabriele Kotsis, David Taniar, Eric Pardede, and Ismail Khalil, editors, *Proc. 7th Int. Conf. Advances in Mobile Computing & Multimedia (MoMM'09)*, pages 331–335. ACM, 2009.

[19] Gefei Zhang and Matthias Hölzl. HiLA: High-Level Aspects for UML State Machines. In *14th Int. Wsh. Aspect-Oriented Modeling (AOM@MoDELS'09)*, Denver, 2009.

[20] Gefei Zhang, Matthias Hölzl, and Alexander Knapp. Enhancing UML State Machines with Aspects. In Engels et al. [3], pages 529–543.

[21] Jing Zhang, Thomas Cottenier, Aswin van den Berg, and Jeff Gray. Aspect Composition in the Motorola Aspect-Oriented Modeling Weaver. *Journal of Object Technology*, 6(7):89–108, 2007.

Model-driven User Interface Development with the Eclipse Modeling Project

Andreas Wolff
Institut für Informatik
Universität Rostock
Andreas.Wolff@uni-rostock.de

Peter Forbrig
Institut für Informatik
Universität Rostock
Peter.Forbrig@uni-rostock.de

ABSTRACT

Model-driven development nowadays often is done using the tools from the Eclipse Modeling Project (EMP). We developed a number of meta-models and transformations to support a model-driven user interface development within EMP. This paper presents our Swing and XUL meta-models and demonstrates what they can be used for.

Author Keywords
UI Model Transformation, CUI Models, Reverse Engineering

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: Miscellaneous

INTRODUCTION

Modeling user interfaces (UI) is a well investigated research topic. For more than twenty years user interfaces are specified in terms of instances of meta-models. Available approaches differ in level-of-detail, some are tailored to a specific domain or interface modalities or target devices, others try to cover general interfaces. A general consensus in model-driven user interface design (MD-UID) exists to distinguish several levels of abstraction. As well as in general model-driven software development, a distinction is made between platform independent (PIM) and platform specific models (PSM). A platform in MD-UID is often considered to be a certain device, or rather the interface source code for that device, itself. Platform independent UI models are called abstract user interfaces (AUI) and platform specific UI models concrete user interfaces (CUI). UI source code is not necessarily source code of a certain programming language, but may as well be an interface description written in a markup language.

There exist numerous markup languages and often they are derived from XML, common examples include XAML, XUL or UsiXML. The main advantage of those languages is the provision of a well-defined grammar and the inherent hierarchical structure, i.e., they provide a suitable meta-model. These characteristics makes them ideal candidates for a model-driven UI development, but not necessarily ideal languages for UI design.

Pre-proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

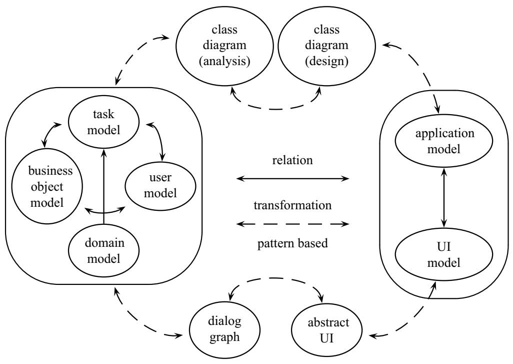


Figure 1. MDD process overview

Non-XML based user interface description languages are even more numerous. They range from comparatively simple template languages to large frameworks as for example Swing or SWT. Creating meta-models, for use within a MD-UID process, of those languages is not as straightforward as with XML-based languages, but we think it might be worth the effort.

In general, we consider model-driven software development as a sequence of transformations of models. We do not think that those transformations can be performed in a fully automated way, but should be executed by humans using interactive tools. We also think these persons, i.e., software engineers and user interface designers; have to base their work on the same models. Our work is especially focused on methods and tools supporting transformations by patterns. Figure 1 sketches all models we consider important within this process and how they are interrelated.

In this paper we focus on the user interface part of the overall process[6]. It is located in the lower right part of Figure 1. UI creation starts off from the task model. A task model itself is not sufficient to express the dynamic relations within an user interface. To capture these, an auxiliary model, the dialog graph, was introduced. The dialog graph model eventually is transformed into an abstract user interface, our PIM.

The idea presented in Figure 1 as such is a process model. In the last years we created different implementations for it. A common problem with these implementations was, that they would only cover quite small portions of the overall problem. We think this was mainly due to a scaling problem. With increasing level of detail, the implementations became too large and too complex to handle.

In our recent implementation we make use of the Eclipse Modeling Project (EMP). EMP is ”a unified set of modeling frameworks, tooling and standards implementations”[1]. ”Standards implementations” refers to OMG MDA standards whose reference implementations are developed by EMP.

Figure 2 is a details view on the transformation from ”abstract UI” to ”UI model” in Figure 1. The meaning of the line styles is as follows: a solid line is a model-to-model transformation. Dotted lines denote a model-to-text transformation. Reverse Engineering (RE) is marked using dashed lines. Abstract and concrete user interface (AUI, CUI$_x$) are EMF models each.

Next to an arrow the applied technique for this transformation was annotated. QVTo[5] stands for OMG’s ”Query View Transformation operational” standard. Acceleo[1] is the EMP implementation of OMG’s ”Model to Text Language” (MTL[4]) standard. PLML[2] itself is a pattern language; we developed an EMF model for it. The PLML model contains transformation specifications in either of EMP’s model-to-model transformation engines. For details on PLML and our usage of it see [9].

Forward engineering from task models typically only yields quite simple user interfaces. To test our ideas against more complex UIs, and for other reasons, we developed some mechanism to reverse engineer the GUI of existing legacy software into instances of our own CUI meta-models. This reverse engineering is not a standard x-to-y transformation.

The rest of the paper is organized as follows: First, we present the XUL EMF meta-model and some of its applications. Afterwards some details about our Swing meta-model and its uses in reverse engineering are discussed.

CONCRETE USER INTERFACE MODELS

We consider meta-models for the platform-specific user interface model to be the core models of a model-driven user interface development. In continuation of our previous experience with MD-UID, we decided to develop CUI meta-models for Mozilla’s XUL and Java Swing.

XUL Meta-Model

For a long time our work is focused around XUL. However, most of the time we did not care about creating a complete and correct meta-model of XUL. We adhered to a minimal

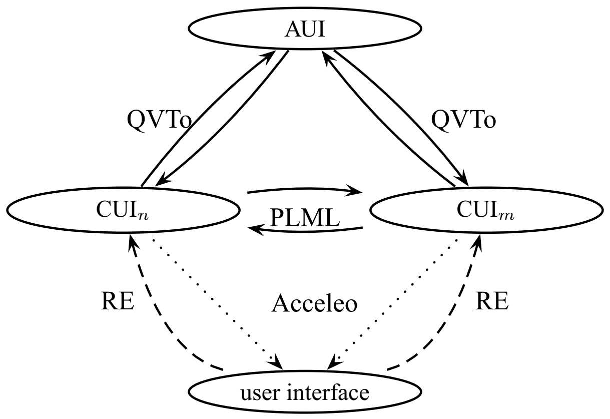


Figure 2. Model relations and transformation techniques

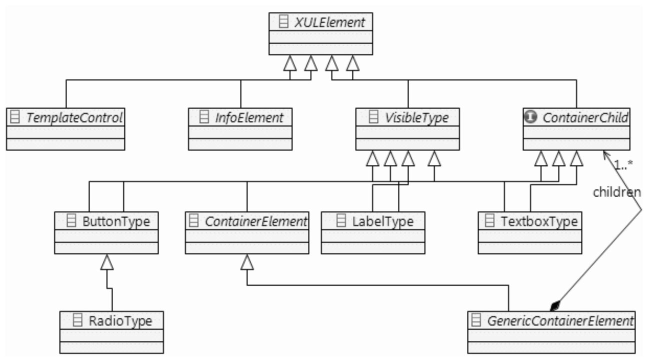


Figure 3. Excerpt from the XUL meta-model

implementation that was only extended if absolutely necessary. So, the XUL standard evolved over time, but our minimal implementation did not. This started to create a number of problems and errors.

A good starting point to create an EMF meta-model for XUL is its grammar. Using freely available sources, it is possible to build this grammar as XML schema definition (XSD) for XUL.

Having a schema definition, a first Ecore model for XUL was easily generated. This automatically generated model had some disadvantages, e.g. anonymous types, multiple types for the same purpose etc. Also, containment relations were not transformed properly and had to be corrected and inserted by hand.

Figure 3 visualizes the basic hierarchy within the XUL Ecore model, many types were omitted. To give an idea about the size: the model consists of 31 data types, 151 classes and interfaces with 443 attributes in total.

XULElement is the root element of the type hierarchy; it contains attributes which are valid for every XUL tag. Each attribute is initialized with a sensible default value or it remains unset if no value is required explicitly.

TemplateControl is the root type for any tag that controls XUL's template engine. InfoElement is the super type of every tag that is not displayed but defines actions, key bindings and such. Any type that implements ContainerChild can be placed into a visual container. Most implementors are concrete VisibleType objects, some are visual containers themselves.

ContainerElement is a marker class for all elements which are able or require to contain sub-elements. For example: a menu-element requires that menu-items are defined. Subtypes of GenericContainer object are a visual containers that include children of type ContainerChild.

RadioType is a radio-button, ButtonType a plain button, LabelType a label and TextboxType a textbox. Those four types are the only concrete classes of Figure 3.

EMF provides a facility to generate editor plug-ins for Ecore models. Using this standard mechanism unfortunately does not yield a valuable XUL editor. Because the default serialization mechanism of Ecore is XMI, the generated editor would not be able to read or write valid XUL files. We had to write our own implementation of the model de-/serialization.

An attempt to use the XUL Ecore model as source to generate an GMF editor was aborted. The resulting generated editor was hardly useful. While it was possible to do some basic editing of labels or buttons, it became very complex to implement a correct layout mechanism. Also, things like the unlimited nesting of group- or tab-boxes proved to be a serious problem. Nevertheless, it would be interesting to build a GMF XUL editor using the XUL Ecore model. We resorted to continue to use our existing graphical XUL editor, but to generate its internal model by customized JET transformations from the XUL Ecore model.

The complete XUL meta-model can be used as a descriptive model for XUL user interfaces, it is freely available from [1].

Meta-Model for Swing

As mentioned earlier, we don't think that XML-derived UI-languages are the best choice for every problem. Also, we acknowledge the fact that a large part of existing user interfaces was not specified using a markup language, but by other means.

The GUI framework Swing of the Java programming language is one example of a user interface framework. It has been included as part of Java since version 1.2. In contrast to its competing GUI framework SWT, Swing is completely platform-independent. For this reason we decided to build a Swing CUI model instead of an SWT model, although the latter is closer related to Eclipse.

Creating the Swing CUI meta-model means to create an Ecore model of Swing. Such a model can be built using a number of methods. EMF has so called model importers which are able to construct Ecore models from XMI, XML Schema, class models of Rational Rose or from annotated Java source code.

So, to define a meta model for Swing there are several possibilities. For example, it is possible to define it manually; nevertheless we dropped this possibility due to the foreseeable large size of the resulting model.

Importing a rational rose model should work well, but it certainly requires to have such a model in the first place. Preparing an XML Schema definition (XSD) for transformation into an Ecore model has its own challenges, as mentioned for the XUL meta-model, but in the absence of such a schema definition this also does not apply.

This leaves two sources for model creation, either using another XMI model or annotated Java code. As most parts of Java Swing are available in source code, it should be possible to create either of these thereof.

EMF's model importer for Java source code requires that each identifier, method, class or interface that is to be transferred into an Ecore model is marked using the annotation @model. This is a highly flexible approach and very useful if one extracts only small parts of existing source code into a model. However, for Swing's about 620 files this method seemed to be almost as laborious, time-consuming and error-prone as defining the model manually.

Many UML tools serialize their models using XMI. Also examining source code to derive a model thereof is a common feature of these tools. In combination, EMF's importer for XMI and an external tool for source code to XMI transformation should provide the desired model of Swing. Unfortunately it was not that simple. The XMI-output produced by UML tools we tested could not be imported by EMF. Looking back on a history of known problems [3] with XMI incompatibilities we did not investigate why exactly the import failed and if it could possibly be repaired by some XSL transformation or the like.

Instead we were searching for a method to create Ecore from Java sources in one transformation, without any intermediate steps, tools or importers.

Developing yet another Java source code parser apparently would not have been a good idea, because there already is a number of parsers available as library or in source code. After some research we developed a small tool which makes use of Java's own JavaDoc parser and is able to convert any Java code into an Ecore model, for details and special considerations see [8].

Swing's source code, in Java version 1.6, has 620 files, and since Swing makes heavy use of AWT, its 368 source code files had to be included into the model as well. The comprehensive Ecore model now consists of more than 2000 types. Additionally it references about 200 external types which are not defined within the source code of AWT and Swing.

Working with such a huge model reveals some weaknesses

[1]http://wwwswt.informatik.uni-rostock.de/metamodels/

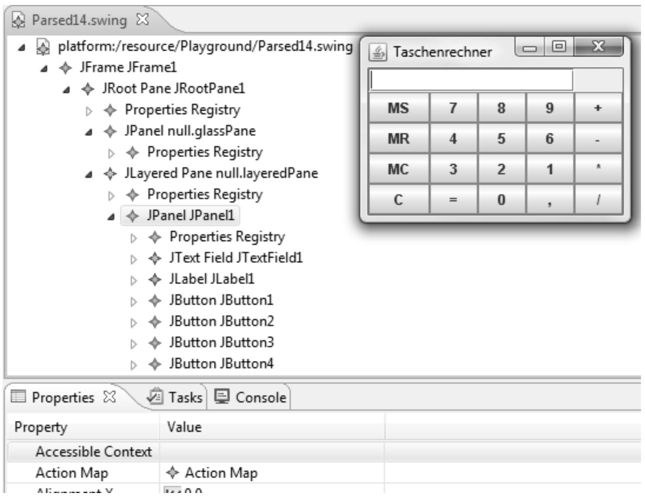


Figure 4. Calculator application and their model screenshot

of the tools of the Eclipse Modeling Project. For example with code generation: The current implementation becomes almost unusable with respect to generation time. Also, some generated methods became larger than allowed by Java, i.e. their compiled byte-code exceeded 64kByte.

To reduce size and complexity the meta-model was pruned using the approach of Sen et al.[7]. Roughly, their idea is as follows: they start with a minimal pruned model that only contains a selection of desired must-have features and classes from the source model. Then their start model is extended step-by-step with features from the source model until their pruned model is valid and is a subset of the source model.

Running this pruning process on the Swing CUI meta-model greatly reduces its size. Size reduction is of course dependent on the selected must-have features, but nevertheless we found that more than 90 percent of the types can be safely eliminated from the CUI model without loosing much expressiveness. This fairly high percentage can be explained by the generation process of the source model. Obviously a lot of implementation-specific types had been generated into the model that were not directly related to properties of UI elements. The complete Swing meta-model, as well as the pruned meta-model, can be obtained from [10].

Populating a Swing model

One use of the Swing meta-model is in reverse engineering. We developed an application which uses the meta-model to capture the static layout of a frame in a Swing application into a meta-model instance. Such a meta-model instance might be considered as a screenshot.

The principle is to traverse the hierarchy of Swing objects within a certain screen and to create the corresponding meta-model instance for each Swing object. Since the meta-model now is a true reflection of Swing itself, no special mappings or other tricks are required. However, some technical problems did arise, because all of the mapping is done at runtime and relies on reflection.

Figure 4 is a combined screenshot of a sample application and its user interface as an Ecore model instance. Beside layout properties, the Swing CUI model does also cover dynamic aspects, like action listener and action command. By a sequence of transformations, we can now edit the user interface in a GUI editor and afterwards re-generate a calculator which would feature another GUI, but uses the same business logic.

CONCLUSION

In this paper we presented our approach to model-driven user interface development using the tools from the Eclipse Modeling Project. We introduced Ecore models for XUL and Swing user interfaces and explained how we derived, build and refined those models. It was also shown that those models can be used for a number of different purposes. These usages include reverse engineering, generating internal models for other tools and also direct editing of user interfaces.

In the future we may resume our attempt to create a GMF-based XUL editor. Beside that, there is a lot of work to do to improve and extend the existing transformations, model-to-model and model-to-text.

REFERENCES

1. Eclipse modeling project (last visited on 22-02-2010). http://www.eclipse.org/modeling/.
2. S. Fincher. Perspectives on hci patterns: concepts and tools (introducing plml). In *Workshop at CHI 2003*, 2003.
3. B. Lundell, B. Lings, A. Persson, and A. Mattsson. Uml model interchange in heterogeneous tool environments: An analysis of adoptions of xmi 2. In *Proc. of MoDELS 2006*, pages 619–630, 2006.
4. Omg standard: Model to text (last visited on 22-02-2010). http://www.omg.org/spec/MOFM2T/1.0/.
5. Omg standard: Query view transformation (last visited on 22-02-2010). http://http://www.omg.org/spec/QVT/1.0/.
6. D. Reichart, P. Forbrig, and A. Dittmar. Task models as basis for requirements engineering and software execution. In *Proc. of Tamodia 2004*, pages 51–58, 2004.
7. S. Sen, N. Moha, B. Baudry, and J.-M. Jezequel. Meta-model pruning. In *Proc. of MoDELS 2009*, pages 32–46, 2009.
8. A. Wolff and P. Forbrig. Deriving emf models from java source code. In *Proc. of Reverse Engineering Models from Artifacts 2009*, 2009.
9. A. Wolff and P. Forbrig. Pattern catalogs using the pattern language meta language. In *Proc. of Visual Formalisms for Patterns 2009*, 2009.
10. Meta-model download (last visited on 22-02-2010). http://wwwswt.informatik.uni-rostock.de/metamodels/.

MDDAUI 2010 Workshop Report

Jan Van den Bergh
Hasselt University - tUL - IBBT
Expertise Centre for Digital Media
Jan.VandenBergh@uhasselt.be

Gerrit Meixner
DFKI
Gerrit.Meixner@dfki.de

Stefan Sauer
University of Paderborn
s-lab – Software Quality Lab
sauer@s-lab.upb.de

ABSTRACT
The workshop on Model-Driven Development of Advanced User Interfaces is a forum of multidisciplinary discussion on how to integrate model-driven development with the more informal methods of user-centered design and development of user interfaces. Starting point of the discussion were the tools, models, methods and experiences of the workshop participants. This report presents the overall aims of the workshop and presents the results of the discussion groups.

Author Keywords
Model-driven development, user-centered design, models, workshop report.

ACM Classification Keywords
D.2.2. Design Tools and Techniques (User Interfaces), H5.2. User Interfaces (User-centered design).

INTRODUCTION
The workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI) was the fifth edition of this workshop series, organized for the first time together with the CHI conference. Previous editions were organized at MODELS and IUI conferences. More information on the background and motivation for this workshop can be found in the CHI 2010 Extended Abstracts [5].

MDDAUI 2010 focused on challenges, opportunities, practical problems, and proposed solutions to increase the usability and user experience of user interfaces created with a model-driven development approach. A highly interactive format was used to foster discussion between participants. All participants, except the organizers, were selected based on papers, which were reviewed by the program committee.

After a short introduction, the thirteen accepted papers were presented in three blocks. Each block consisted of four or five seven-minute presentations followed by a short discussion to identify potential points for more elaborate discussions. The resulting set of potential discussion points were grouped in a collaborative effort by all participants into three discussion topics, which were later discussed in three separate discussion groups. Thus, the identified topics reflect important issues in this field from the viewpoints of the workshop participants.

Discussion Topic	Discussion Points
Integrate knowledge from other fields	• HCI patterns • Cognitive models • HCI and usability guidelines and standards on know-ledge representation • User experience and usability
Multi-device and multi-modal interaction generation	• New common reference framework? • Different abstractions? • Cost versus usability • Dynamic distribution
Development processes	• End-user development • Customization • Iteration and prototyping • ISO 13407 [3] • Standards about development processes

Table 1 Discussion topics and discussion points

Table 1 shows an overview of the discussion topics and the individual discussion points that were identified. Some of the points evolved as a result of merging several initial discussion points raised during the presentation blocks.

In the remainder of this workshop report, the results of the different discussion groups are summarized, followed by a discussion on how we created and presented the workshop

Proceedings of the 5th International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2010): Bridging between User Experience and UI Engineering, organized at the 28th ACM Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA, April 10, 2010.

poster, which was also created in a collaborative effort of several workshop participants.

GROUP DISCUSSIONS

The group discussions formed a major part of the workshop. They started before lunch and ended just before the end of the workshop, leaving enough room for the plenary presentation of the results of the group discussions and a few short closing remarks.

Integrate knowledge from other fields

Out of practical considerations the discussion in this discussion group focused on the integration of knowledge from cognitive sciences to complement the knowledge already available from the engineering disciplines. The central goal of integrating this knowledge is to improve usability of user interfaces that are generated by a model-driven development approach.

Starting point of the discussion were the models and abstraction levels of the (revised) reference framework for plastic user interfaces [2] (Cameleon reference framework). Different contributions to these models from the cognitive sciences community were proposed for specific abstraction levels. The saliency model could e.g. be used to test final user interfaces (as presented by Jeremiah Still in the workshop) and provide feedback to the concrete user interface model. It was however indicated that a significant amount of research is still necessary to establish this capability. Cognitive workload models were identified as a potential candidate to enable transition from concrete user interfaces to abstract user interfaces. The participants of the discussion group believed that the latter could also benefit from an inclusion of the (relative) importance of its components in the model. Finally, the concepts and tasks layer of the Cameleon reference framework could include knowledge from the GOMS model (Goals, Operators Methods Selection rules) [1] for task analysis. Support for indicating task frequency was also considered important.

Besides the models, HCI patterns, guidelines and standards were identified as important containers of knowledge to support transitions between the different abstraction levels. Patterns (also discussed in the presentation of Andreas Wolff) could assist in transitioning between all abstraction levels, while standards (including guidelines) were mostly considered beneficial in the transition between final and concrete user interfaces.

HCI patterns promise to solve the shortcomings of standards and guidelines by representing the design knowledge in a machine-readable and reusable form. By specifying “when, how and why” they enable automatic processing of the represented design information and are therefore suitable for the model-based generation of user interfaces. For using HCI patterns in a model-based development process several problems should be addressed in the future e.g. lack of formalization, lack of organization and the lack of effective tool support.

Multi-device and multi-modal interaction generation

This discussion group focused on how to create good user experience for user interfaces that can (semi-) automatically adapt to multiple devices and support multi-modal interaction at runtime. Consistency was identified to be an important factor for good user experience (or rather usability, as argued by some participants?). Consistency has different aspects that need to be balanced: consistency within the user interface of a single application on a single device, consistency with user interfaces of other applications on the same device and consistency between the user interfaces of the same application on different devices. Depending on the kind of applications and the desired user experience, different kinds of consistency may be more important.

Another discussion point was centered upon the question: "How to ensure consistency?". Different approaches were discussed which would be suitable for different kinds of consistency. Guidelines were considered important, especially to ensure consistency between user interfaces of different applications on the same device, but they could also support consistency within an application's user interface. The usage of one or more common models (e.g. at the concepts and task level in the Cameleon reference framework [2]) for the user interfaces of an application on multiple platforms was raised as another potential means to ensure consistency. Device-specific models were, however, considered necessary to generate usable user interfaces.

In order to benefit from common models for ensuring consistency, the existence of both mappings and automated transformations was considered important. Mappings are important for logging and explaining links between the different models, while transformations support automation in the generation of models (and mappings). A last thing that was prevalent during the discussion was the importance of explaining why certain guidelines and/or transformations are present. Guidelines and transformations should become transparent to other stakeholders and be accomplished, e.g., by descriptions of the rationale.

Development processes

A large part of the discussion in the third discussion group was centered on the role of models in software engineering and user-centered design processes. First, a set of models and sub-models was gathered (see Table 2). The discussion then moved to different properties of these models within the development process. Three central questions were examined: "How are models used?", "At which level of abstraction are they?" and “How are the models related?”

The former question led to the distinction between design time models and runtime models. For design time models, tool support (user interface and languages) was considered a major area for future work, while models at runtime could be used for simulation (interpreting and executing) and analysis, and could play a major role in reverse engineering of UI models from interactive systems. The discussion group members agreed: "Design-time models can ideally be used or instrumented as runtime models (or they should at least be related)."

Model	Sub-models
User model	• Preference, knowledge • Perception, cognition, (motor) action • Behavior, learning
User activity model	• Task • Scenario • Role • Workflow (procedures, cooperation)
Collaboration model	• Social behavior
User interface model	• Guidelines (e.g. ergonomic rules) • Dialogue model • Presentation model (input, output) • Platform model (CPU, description of device) • Interaction model (interaction techniques, widgets, gadgets, UI components, modalities)

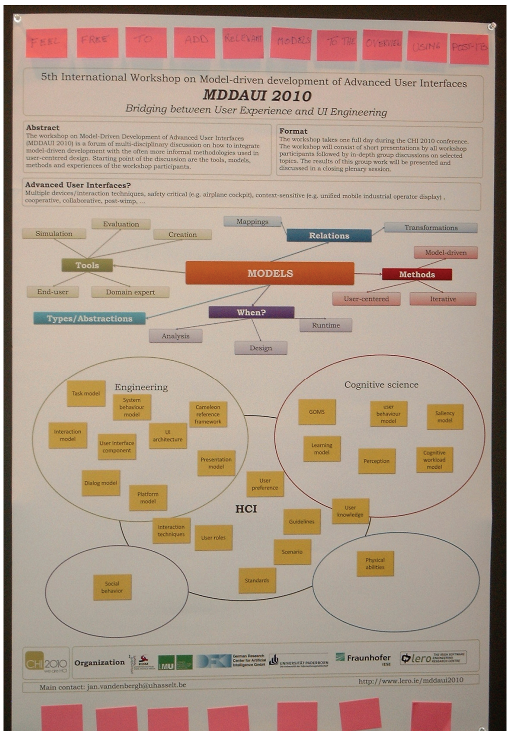
Table 2 Models and sub-models

The discussion about abstraction levels started off with the observation that there are different views of abstractions. Among them, the Cameleon reference framework [2] and the Model-Driven Architecture [4] (MDA) were considered the most relevant. A mapping between their respective levels of abstraction was documented, while remarking the different notions of platform: Platform for the Cameleon reference framework refers to "a class of devices with similar interaction resources"; for MDA it refers to a certain combination of software and hardware. When reviewing the models, it was observed that most of the models did not fit completely in neither the Cameleon reference framework nor MDA; some extensions are required to either of them. The user activity model only fitted the concepts and tasks level of the Cameleon reference framework.

Finally, end-user development was also discussed. First, domain experts were considered as end-users of models, further distinguishing models for them from (user) models of them. Models for end-users were considered to support the active participation of end-users in the development of the user interface. The activities for which models are suitable were listed as design, program and customize.

In order to establish these goals, several requirements were listed: The models should be understandable, have the "right" level of abstraction and show the correspondence of concepts. Furthermore, tool support on different levels as well as domain-specific languages are also required for end-user development.

The last discussion group also touched the topic of guidelines and standards in user interface development. To ensure consistency and a high degree of usability across several user interfaces, standards and guidelines should be used during the user interface development process. For automatically considering standards (e.g. ISO 9241, VDI/VDE 3850) during the user interface generation process, e.g. for automatic verification in the concrete user interface layer, standards and guidelines have to be available in a formal notation. One possible solution could be the integration of the knowledge of standards and guidelines in knowledge bases. To reach this goal, much work still has to be done.



The workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI 2010) is a forum of multi-disciplinary discussion on how to integrate model-driven development with the often more informal methodologies used in user-centered design. Starting point of the discussion are the tools, models, methods and experiences of the workshop participants.

The workshop takes one full day during the CHI 2010 conference. The workshop will consist of short presentations by all workshop participants followed by in-depth group discussions on selected topics. The results of this group work will be presented and discussed in a closing plenary session.

Multiple devices/interaction techniques, safety critical (e.g. airplane cockpit), context-sensitive (e.g. unified mobile industrial operator display), cooperative, collaborative, post-wimp, ...

Figure 1 The "interactive" workshop poster

WORKSHOP POSTER

After the workshop, a poster was prepared in cooperation with several workshop participants. It was designed as an "interactive poster" inviting the viewers of the poster at the CHI conference to add artifacts. The poster consists of three major parts, as can be seen in Figure 1. The top part documents the goals and the format of the workshop, the middle part consists of a mind map that illustrates the topics

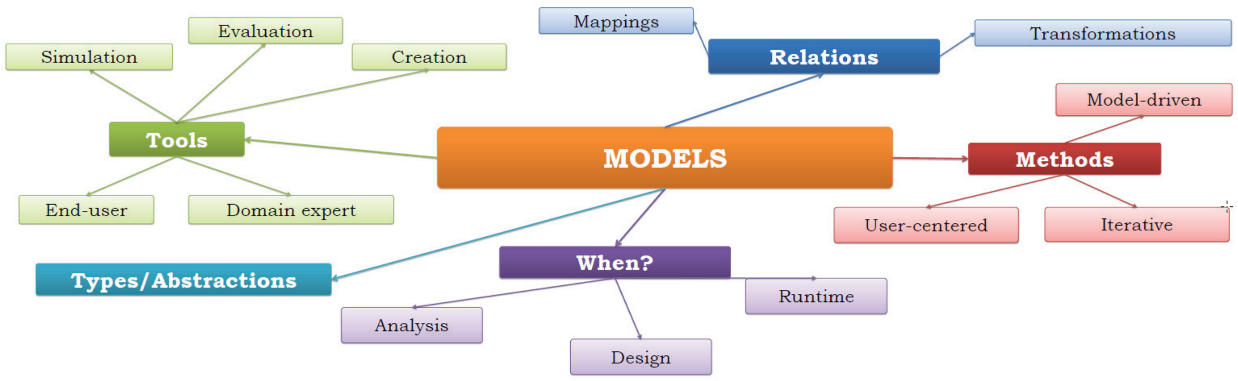


Figure 2 Mind map of the topics discussed during the workshop

that were discussed in the workshop, and the bottom part gives an overview of the different models and information artifacts that were considered from the different disciplines in the creation of user interfaces. The models were represented in a way similar to yellow post-it notes, while pink post-its invited viewers of the poster to contribute other models and information artifacts. In this way some more information artifacts were collected during CHI 2010.

Figure 2 gives a detailed view of the elaborated mind map that structures the area of model-driven user interface development. The main aspects identified around the central concept “models” are relationships (transformations and mappings), the main characteristics of the development methods (model-driven, iterative, or user-centered), the software development phase when models are applied (analysis, design, or runtime), the different scopes of tools (creation, evaluation, simulation, end-user, or domain expert), and the models’ types and abstraction levels. The latter were elaborated further in the bottom part of the poster, based on the models in Table 2, but also including a set of other models as discussed in the previous section.

DISCUSSION

The fifth edition of MDDAUI, organized the first time at the CHI conference, shifts the workshop further towards a better integration with user interface design and HCI. According to the conference guidelines, the number of participants was restricted and emphasis was put on the discussions. Thus, paper presentations were kept really short, giving us much time for comprehensive and fruitful discussions, including both highly respected and well established experts and novices to the field and/or the scientific community.

The workshop also highlighted a number of challenges for the community and the workshop organizers:

- Development of a reference framework that better captures the needs of model-driven development of user interfaces providing good user experience.
- Incorporation of often relatively informal knowledge (such as HCI patterns, guidelines and standards) from non-engineering fields into models, mappings and transformations to support better user experience.
- Convince domain experts ― both horizontal, (such as designers, information architects and usability experts) and vertical (such as health, finance and transportation) ― of the benefits of participating in the MDDAUI community.

ACKNOWLEDGMENTS

We thank the CHI organizers for providing the opportunity to organize MDDAUI 2010 together with CHI 2010 as well as the student volunteers and other people from the CHI organization and program committees. Special thanks also go to the institutes and organizations supporting the participants and organizers to attend and organize the workshop. Last but not least, many thanks to all MDDAUI 2010 participants who helped to make this workshop a success.

REFERENCES

1. S.K. Card, T. P. Moran, and A. Newell. The psychology of human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.
2. G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, N. Souchon, L. Bouillon, and J. Vanderdonckt. Plasticity of user interfaces: a revised reference framework. In: First International Workshop on Task Models and Diagrams for User Interface Design TAMODIA2002, pages 127–134, July 18–19, 2002.
3. International Standards Organization. ISO 13407 – Human-centred design processes for interactive systems. Geneva, Switzerland 1999.
4. J. Miller and J. Mukerji. MDA guide version 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf, 2003.
5. J. Van den Bergh, G. Meixner, K. Breiner, A. Pleuss, S. Sauer, and H. Hussmann. Model-driven development of advanced user interfaces. In: CHI 2010 Extended Abstracts, pages 4429–4432, 2010.

5th International Workshop on Model-driven development of Advanced User Interfaces

MDDAUI 2010

Bridging between User Experience and UI Engineering

Abstract

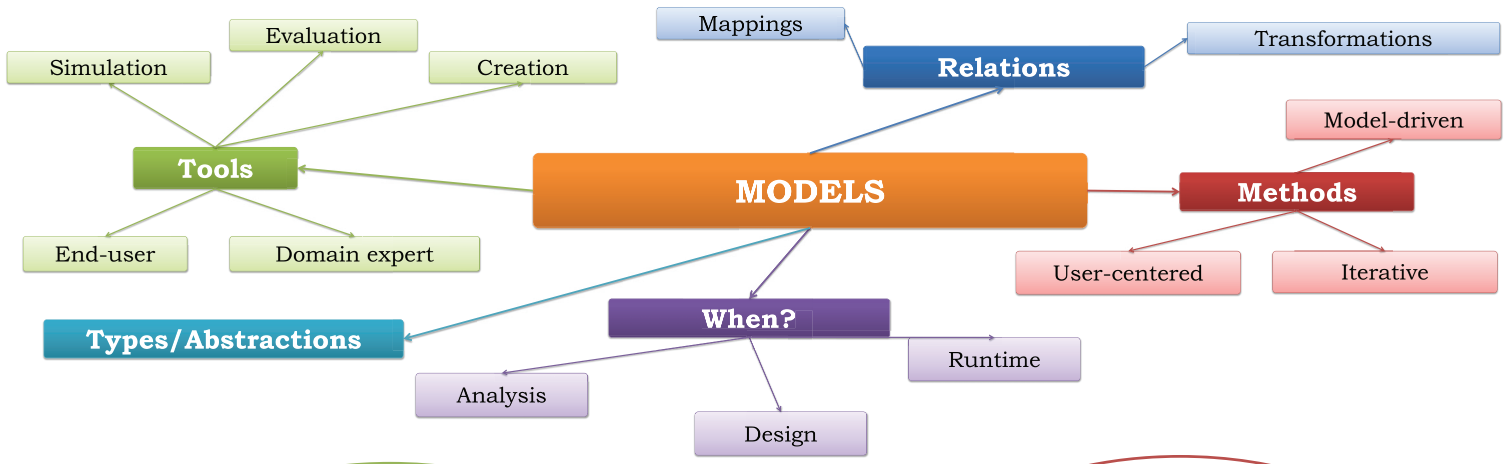
The workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI 2010) is a forum of multi-disciplinary discussion on how to integrate model-driven development with the often more informal methodologies used in user-centered design. Starting point of the discussion are the tools, models, methods and experiences of the workshop participants.

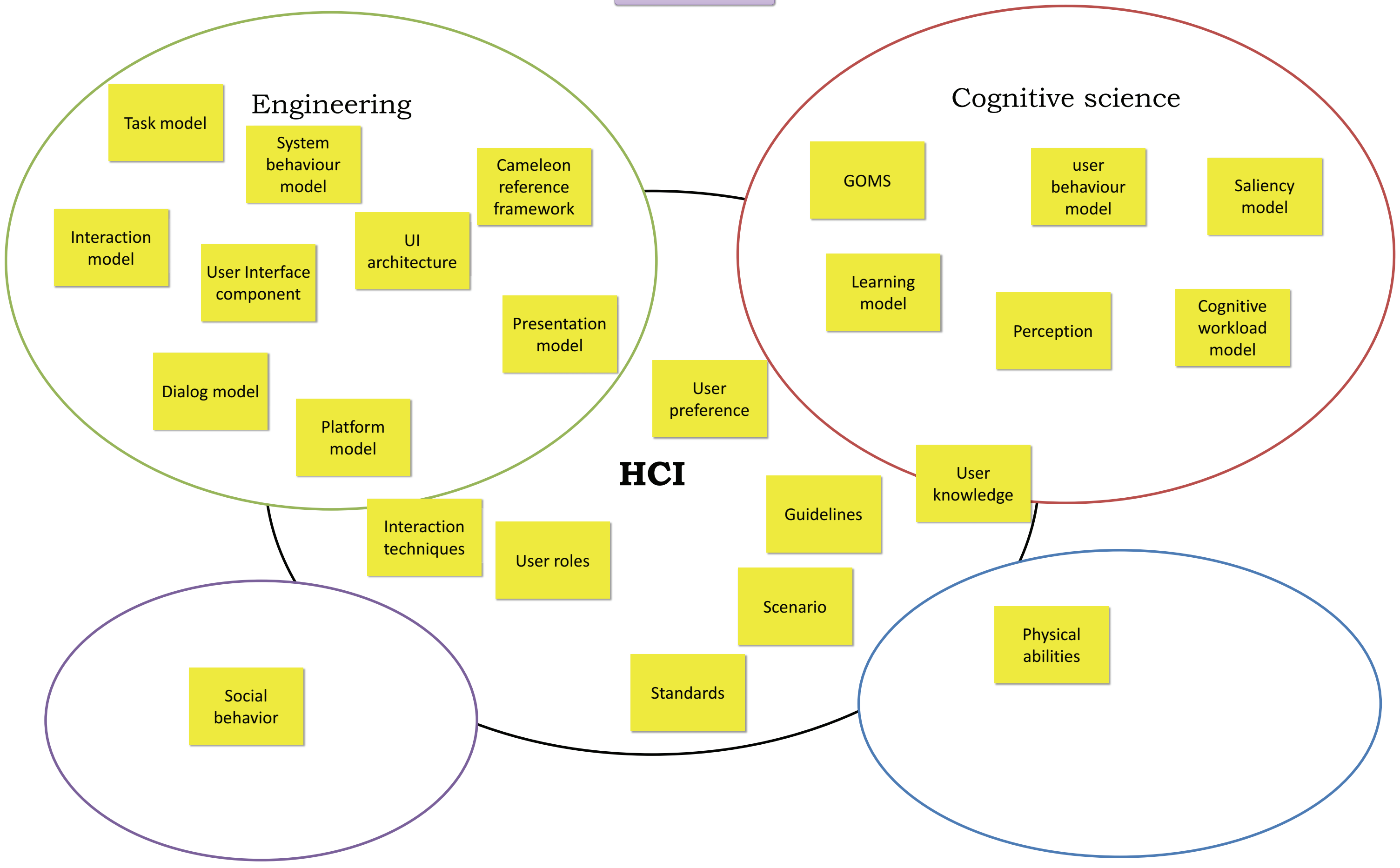
Format

The workshop takes one full day during the CHI 2010 conference. The workshop will consist of short presentations by all workshop participants followed by in-depth group discussions on selected topics. The results of this group work will be presented and discussed in a closing plenary session.

Advanced User Interfaces?

Multiple devices/interaction techniques, safety critical (e.g. airplane cockpit), context-sensitive (e.g. unified mobile industrial operator display) , cooperative, collaborative, post-wimp, ...





Organization

Main contact: jan.vandenbergh@uhasselt.be http://www.lero.ie/mddaui2010