Workshop Report: Model Driven Development of Advanced User Interfaces (MDDAUI)

Andreas Pleuß[1], Jan van den Bergh[2], Stefan Sauer[3], and Heinrich Hußmann[1]

[1] Institute for Computer Science, University of Munich,
Munich, Germany
{Andreas.Pleuss, Heinrich.Hussmann}@ifi.lmu.de
[2] Expertise Centre for Digital Media, Hasselt University,
Hasselt, Belgium
Jan.VandenBergh@uhasselt.be
[3] Institute for Computer Science, University of Paderborn,
Paderborn, Germany
sauer@upb.de

Abstract. This paper reports about the workshop *Model Driven Development of Advanced User Interfaces* (MDDAUI) which was held on October 2nd, 2005 at the *MoDELS/UML 2005* conference in Montego Bay, Jamaica. We introduce the topic of the workshop and give an overview about the workshop's structure. Then we summarize the accepted contributions and finally we provide an overview about the workshop discussion and its results.
It is intended to provide a follow-up event of this workshop in the next year.

1 Workshop Topic

The user interface of an application is often one of the core factors determining its success. Existing approaches for user interface development provide abstract and platform independent models for basic widget-based user interfaces. This workshop deals with model driven development of advanced user interfaces.

Today there is an increasing demand for user interfaces with high usability whereas the covered functionality gets more and more complex. Applications often provide more intuitive interaction techniques as well as tailored and customizable representations of information. Complex information is presented in individual and interactive graphic objects. Techniques like animation or 3D visualization are used to achieve a more comprehensive or attractive presentation. Some user interfaces also use additional perception channels beside graphics, e.g. speech or haptic output. In addition, the usage of temporal media types, like animation or sound, and the combination of different modalities lead to synchronization and dependency issues. Moreover, even within a single modality, different devices are used for different purposes.

While on the one hand such a broad spectrum of presentation, perception, and representation media has been established, it is on the other hand necessary

to figure out a possible configuration which best addresses the user's needs. An appropriate level of usability often requires customization of the user interface. Some applications also require an automatic adaptation to their current runtime context, like location or available devices.

Significant work has been done addressing these issues for an advanced development of user interfaces and further research is still under progress. For application development in general, some of the most important state-of-the-art concepts are the Unified Modeling Language and the Model Driven Development (MDD) paradigm. Thus, the workshop involves the areas of software engineering and human-computer interaction and aims to integrate the required methods and concepts of advanced user interface development into MDD. The goal is to support a model driven development of applications under comprehensive consideration of features of advanced user interfaces. The target application area is not limited to classical business applications and may include for example games, simulations, and infotainment or edutainment applications.

2 Spectrum of Submissions and Participants

Interested participants were asked to submit a short position paper of four pages length in double-column format. We received 13 submissions from which 9 have been accepted according to the reviews of the program committee. As various work has already been done in the area of user interface development, we concentrated on papers which demonstrate the experience of the authors in this field and clearly focus on the workshop topic.

The resulting spectrum of participants included people from human-computer interaction domain as well as people working mainly in the field of model driven development. Besides people from academia, we had also participants working in industrial context.

As a result, the presentations included a broad spectrum of views on the workshop topic. In particular the literature references in the accepted papers provide a very comprehensive overview over existing work relevant in this research area. Most accepted papers propose a concrete solution for some of the open problems while a few are position papers which summarize the state-of-the art and outline critical problems and challenges in model driven user interface development.

3 Workshop Schedule

The workshop took one day during the week of the MoDELS/UML 2005 conference. We provided sufficient time for in-depth discussion, scheduling the presentation of papers mainly within the two morning sessions. A selected number of papers were presented as long presentations while all other accepted papers were presented in short presentations.

The talks in the two morning sessions were structured according to the following scheme: The first session was used for the presentation of approaches

with a more general usage of models and model-based techniques for user interface development. The second session was used for papers which address more explicitly the techniques and standards of model driven development like *model driven architecture* (MDA). Finally, two short talks which presented more specific approaches opened the discussion sessions in the afternoon.

The detailed program can be found on the workshop webpage [1].

4 Presented Papers

The following section summarizes the papers presented on the workshop. All papers are published in [2].

4.1 A. Boedcher, K. Mukasa, D. Zuelke: Capturing Common and Variable Design Aspects for Ubiquitous Computing with MB-UID

This paper describes a model-based development process for useware, i.e. hardware and software components used for operating technical systems, e.g. in a factory. The process allows developing the user interface for different devices in a consistent way by identifying commonalities and variabilities of the interfaces. The models in the process are described using an XML language called UseML.

4.2 A. Wolff, P. Forbrig, D. Reichart: Tool Support for model-based Generation of Advanced User-Interfaces

The paper shows a model driven approach for user interface development under consideration of existing models like task models, dialogue models, and abstract user interface models. In particular, the paper focuses on the tool chain to support the models and the transformations between them. To enable an evolutionary development process, the tools apply to an "edit by replacement" approach which allows the developer to keep the connections of model elements throughout the different levels of abstraction in the development process.

4.3 S. Basnyat, R. Bastide, P. Palanque: Extending the Boundaries of Model-Based Development to Account for Errors

This approach addresses the domain of safety-critical interactive systems by integrating information about possible erroneous user behavior into the models. Therefore, the authors introduce an extension for task models. As an extended system model they use an existing approach called ICO (Interactive Cooperative Objects). Finally, they define the relationships between the extended task models and the ICOs.

4.4 N. Sukaviriya, S. Kumaran, P. Nandi, T. Heath: Integrate Model-driven UI with Business Transformations: Shifting Focus of Model-driven UI

This position paper provides an overview about current problems and challenges in practical application of model driven user interface development. Therefore, the authors explain several problem fields based on their experience with the integration of user interface design with model-based approaches. For the goal of integrating user interface development and model-based business process modelling, the authors show commonalities between these two tasks and conclude that business analysis can act as a useful starting point for user interface development.

4.5 J.S. Sottet, G. Calvary, J.M. Favre: Towards Model Driven Engineering of Plastic User Interfaces

This paper presents a model driven approach for the development of plastic user interfaces. With the term plasticity the authors refer to user interfaces which are adaptable to the context of use specified by the respective target platform, user, and environment. They consequently apply the state-of-the-art techniques of model driven development. In this way, they aim on the one hand to overcome known problems of monolithic code generators for user interfaces but on the other hand also to gain new insights – with user interfaces development as a very complex application domain – in the research area of model driven engineering.

4.6 T. Schattkowsky, M. Lohmann: Towards employing UML Model Mappings for Platform Independent User Interface Design

This paper introduces a model driven approach for the development of user interfaces based on an extended UML class diagram, called information model. The information model contains the information to be provided by the system augmented with abstract information whether the user is allowed e.g. to create, delete or edit this information during runtime. On that base, transformation rules are described which finally lead to the specification of a concrete platform specific user interface.

4.7 J. Van den Bergh, K. Coninx: Using UML 2.0 and Profiles for Modeling Context-Sensitive User Interfaces

The paper presents a UML profile for context-sensitive user interfaces. It introduces stereotypes to support the modelling of common user interface models, like task models and dialogue models, with UML 2.0. In addition, it presents a context model to specify the context of the application, like the user and the environment. The elements from the context model can then be used to specify the influence of the application's context on its behaviour.

4.8 R.I. Bull, J.M. Favre: Visualization in the Context of Model Driven Engineering

The paper proposes a model driven approach to realize complex visualizations of large data. Therefore, it provides explicit metamodels for common complex graphical visualizations. A complex graphical visualization of a system's data can thus be generated by defining transformations between the models of the source data and the target graph. As an example, the authors show the generation of visualizations for source code.

4.9 C. Nill, V. Sikka: Modeling Software Applications and User Interfaces Using Metaphorical Entities

The paper discusses the use of metaphors to design interactive software. In this way it proposes to combine the concepts from user interface metaphors and the concepts from the "Tools and Materials" approach from software engineering domain. The authors propose to use the metaphors as a kind of conceptual patterns for an application's user interface.

5 Workshop Discussions

The sessions in the afternoon were mainly used for discussions. The discussions started with a collection of relevant general research questions in the whole group. We considered three complexes of questions as most relevant for the model driven development of (advanced) user interfaces:

- Models: Which are the adequate models? Which models are more important in which situation and what are some important points of attention?
- Process: What is an adequate development process? How to integrate prototypes? How to incorporate the user?
- Validation: How to validate the proposed approaches? How to validate the models?

On that base we formed two discussion groups to go more into the details. Each group consisted of eight workshop participants. The first group discussed on models for advanced user interfaces, the second group discussed on the model driven development process for (advanced) user interfaces. The third topic, validation, was not further discussed at this workshop.

5.1 Models

The discussion about the kind of models that are adequate in model driven development of advanced user interfaces was started by the observation that the answer to this question would probably be dependent on the kind of application. Therefore an attempt was made to define a limited number of categories of

user interfaces. The discussion resulted in three categories identified by typical examples: “wordprocessor”, “website”, “first person shooter”.

The first category, identified by the wordprocessor, has a user interface built around a document or object. Other examples are graphics editors, spreadsheets or integrated development environments. The manipulation of a central object or document is the main purpose of an application. Much of the functionality is dependent on the state of the object (e.g. selection of a word, paragraph, figure, table or no selection at all).

The second category, identified by the website (but also database applications), has a central role in the interface for different kinds of user-tasks. When lots of tasks are available, they are probably arranged around user roles or user interests, not around a central object.

The third category, identified by the first-person shooter, contains applications that are proactive, and often offer an immersive experience. These applications typically do not wait for user interaction, but rather act on their own and the user reacts to changes. Simulation software, action games and role playing games can be put in this category.

The discussion then continued on what makes the user interfaces of these applications inherently different. Four different criteria were determined: the influence of time on the behaviour of the application, the importance/structuring of user-tasks, the line between application logic and user interface, and the nature of the “controls”.

Table 1 gives an overview of the perceived characteristics for each combination of categories and criteria. The score given for task importance should be regarded as follows: High task importance denotes that one can easily establish a number of steps or actions that have to be taken to reach a certain goal. One can therefore create a clear and highly structured task models organized in a tree structure for the given category of application. The medium score is given to applications like the word processor, which typically offer a lot of functionality. A task model of such an application would consist of a lot of tasks that can be executed in parallel or in an order-independent way. Such tasks typically have one or more preconditions which depend on the results of tasks that are performed at an earlier moment in time, such as a piece of text has to be selected before it can be made bold. Another example can be that some information had to be put on the clipboard (precondition) before it can be pasted somewhere in the document. A low score is given for user interfaces where it is very hard to establish a direct correlation between the high-level goals and the low-level actions. One could say that the scores are inverse proportional to the number of ways to solve a problem through the user interface of the application.

The timing of interactions has different importance for different types of applications. The time it takes to do a certain action is in general not very important in the first two categories of user interfaces/applications, while this is in many cases crucial for applications of the last category, since the state of the application changes whether or not a user interacts with it.

	Category 1	Category 2	Category 3
Task importance	medium	high	low
Time dependency	low	low	high
Separation of UI	medium	high	low
Controls	specialized, standard	standard	custom, no direct clues

Table 1. Characteristics for the different categories

The degree of separation between the user interface and the application logic also depends on the category of application. This is also shown by the type of controls that are used to realize the user interface of applications in each category. The applications in the second category can do with standards controls. Applications of the first category require a single specialized and complex control. In the last category of applications, the user interface is usually for the greatest part custom-made, with highly specialized controls or actions that are only accessible through short-cuts (no visual or other clues that those actions are available).

During the discussion it became clear that different categories of user interfaces can be discerned which have different requirements for models that would be used to describe them. It was, however, noted that the categorization that was made is not absolute and that it would need additional research to confirm that these categories are really different. A last remark was that one application can have different parts that belong to different categories. For example, the general configuration part of a 3D racing game application is part of category 2, while the part responsible for game play is of category 3 and the part to configure the car is of category 1.

As a general conclusion we can say that it is very probable that there is no single set of models optimal for every kind of user interface. It is however necessary to investigate the compatibility of the models to enable model driven development of complete applications, including the user interface.

5.2 Process

The discussion about the model driven development process for (advanced) user interfaces focused on problems of the practical application of model driven approaches. Thereby we considered the knowledge of those participants with experience as user interface designer in industrial context. We found the following research question as an extract of the most important issues regarding the model driven development process in practice:

> In the field of human-computer interaction and user interface design, well-established techniques exist to incorporate on the one hand creativity and on the other hand the user into the user interface development process. This observation yields the question how to incorporate the user/prototypes/creativity into the model-based development process?

In this view, we have elaborated the following properties of the model driven development process for user interfaces:

Models as Transmitter of Information. The role of models from the viewpoint of user interface design is to act as a kind of transmitter for the essential information about the user interface. A first aspect here is the transmission of information between different tools in the development process. With regard to user interface design, the involved tools here are not only modelling tools, but also tools to support the creative aspects mentioned in the question stated above. These are not only common user interface builders; often more sketching and drawing oriented tools like Photoshop are used to freely create ideas about a possible user interface design.

A second, possibly closely related, aspect is the transmission of information between different stakeholders within the development process. The supposed user of the system is often interested in the concrete layout, represented e.g. by screenshot-like images or prototypes, to discuss ideas and details about the system. The software developer, however, is interested in an abstract view on that part of the user interface which interrelates with the application logic.

In such a scenario, the models can act as the common information base for all the different representations and views on the user interface, containing the important essence of the overall decisions and hiding irrelevant or too much detailed information of respective development artefacts. For example from a screenshot of the user interface, the involved abstract user interface elements may be the only important information to be provided for further development steps. Other models may specify the number of different presentation units (e.g. screenshots) required for the overall system and the links between them.

Resulting Benefits. The discussion above led us to the core benefits of a model driven process in the view of such a design-oriented process. As the models act as transmitter for information, they can increase efficiency by preventing to create very similar information many times during the development process. Examples are hi-fi mock-ups or prototypes which build up on low-fi mock-ups like sketches: While the concept is derived from foregoing low-fi mock-ups, the realization itself has to be started from scratch again. If the required essence of the earlier development steps is captured within models, it will be possible to generate starting points for the next development steps.

Another benefit of models is their potential as communication tool between different stakeholders. This is especially important the context of user interface development, as it usually involves additional stakeholders (in addition to the software designer and the customer) like e.g. a user interface designer or, more specifically, a graphic designer. The models can also act as a kind of contract between them.

Certainly, general advantages of models are likewise significant for user interface development, like quality assurance, model-based validation, and documentation.

Requirements on models. Finally, we discussed the resulting requirements on models to achieve a process which integrates experts, methods, and tools from user interface design area into model driven development. First, the models have to be flexible enough. This means in particular that models are allowed to be incomplete during the development process, to enable the creative design process and not to force designers to make decisions earlier than necessary.

Second, the modelling language must provide a high degree of usability for stakeholders who have to deal with the respective model. For example, models which have to be read or created by user interface designers should not base too much on technical concepts. A possibly intuitive notation and the recognition of well-known concepts in the modelling language can strongly increase the general acceptance of the language and thus the success of the whole process.

6 Conclusion

The workshop about Model Driven Development of Advanced User Interfaces was a first event to bring together knowledge from the fields of user interface development and model driven engineering. We received a broad spectrum of high quality contributions from people with very different backgrounds within this area. Thus, the submitted papers provide a comprehensive overview about work and literature relevant in this field.

Based on the success of this workshop, it is intended to aim for a follow-up event in the next year. A possible future direction is a more detailed analysis of the differences and commonalities between the available concepts.

7 Acknowledgements

We thank the workshop participants for their high quality contributions as well as the program committee members for their valuable reviews.

8 References

References

[1] MoDELS 2005 Workshop on Model Driven Development of Advanced User Interfaces, Workshop Webpage, 2005 `http://www.edm.uhasselt.be/mddaui2005/`

[2] Pleuß, A., Van den Bergh, J., Hußmann, H., Sauer, S.: Proceedings of Model Driven Development of Advanced User Interfaces. CEUR Workshop Proceedings, Vol. 159, 2005, `http://ceur-ws.org/Vol-159`