SmartMote: A model-based Architecture for context-sensitive User Interfaces in Future Factories

Marc Seissler*, Kai Breiner, Mathias Schmitt*, Samuel Asmelash***, Johannes Koelsch*****

**German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany*
(e-mail: {surname}.{name}@dfki.de)
***Fraunhofer Institute for Experimental Software Engineering IESE, Kaiserslautern, Germany*
(e-mail: kai.breiner@iese.fraunhofer.de)
****University of Kaiserslautern, Kaiserslautern, Germany*
(e-mail: {asmelash; koelsch}@rhrk.uni-kl.de).

Abstract: Ubiquitous information access within intelligent environments – like the *SmartFactory*^{KL} – will become more and more important in everyday life. Universal mobile interaction devices will increase the flexibility of the service men during their maintenance task. Although this greater flexibility can improve the maintenance processes it also results in new problems for the users (e.g. information flood, remote function controls) and developers (e.g. multi-platform development) that have to be addressed to guaranty the efficient development of usable, interactive systems for a safe human-machine interaction in such environments. In this article we present a model-based architecture for the design of context-sensitive mobile user interfaces that allow the abstract specification of an run-time adaptive user interface. Based on the model-based architecture, a prototypical implementation of a model interpreter – the SmartMote – and an industrial use case are presented that present the use of a run-time adaptive user interface and the feasibility of the presented approach.

Keywords: Model-Based Design, Usability Engineering, Ambient Intelligence, Mobile Interaction.

1. INTRODUCTION

Mobile devices such as smartphones, tablet PCs and also notebooks are already integrated into daily life fulfilling various purposes. These devices work seamlessly together since they are able to share (e.g., using the cloud) and gather information (e.g., using positioning services). By incorporating all these features, the devices are able to provide much more functionality than similar stand-alone versions. This new era of smart devices and "ubiquitous computing" was already described in the vision of Marc Weiser (Weiser, 1999).

As computational devices are getting smaller and are produced for being used by a broader audience, they also found their way into modern production facilities. While taking profit of the new characteristics of these devices future production environments will be much more dynamic and intelligent as compared to current rather static facilities. Profiting from the distributed computational power future production environments will become "smart". One example is the *SmartFactory*^{KL}, which is a demonstration environment for the interplay of future technology, to increase the flexibility of industrial production processes (Zuehlke, 2010).

Due to the popularity of standardized communication protocols and wireless communication technologies (such as TCP/IP and Bluetooth) mobile interaction systems can be seamlessly integrated into the production environments, enabling new forms of human machine interaction. Mobile universal interaction devices (see Fig. 1) for example offer a location independent information access and control of production processes that increases the flexibility and productivity during maintenance tasks. In the case of being used in a maintenance scenario, the single devices do not need to be approached directly as all relevant information is also available remotely.

This makes such devices attractive for being used in maintenance and repair tasks.

Fig. 1. Examples for mobile universal interaction devices for production environments

Besides the easier information access mobile interaction devices can be used to implement new interaction scenarios. E.g. in case of an incident, the service men can be assisted by the mobile interaction device with the context relevant information that is frequently used in the particular situation. This information can be in the event of a malfunction of a field pump the provision of a description of steps the maintenance staff has to follow to repair the pump.

Another example for an advanced usage scenario is the

adaptation of the information by the user interface according to the location of the maintenance staff. While maintaining a robot arm, there is the chance that the unsafe zone needs to be entered to conduct certain physical activities. At the same time the robot can reach the user which may lead to injuries or death. This needs to be (and can be) detected by the smart production environment via indoor positioning systems, which in consequence has to deactivate all potential dangerous functions of the smartphone that will lead to human damage.

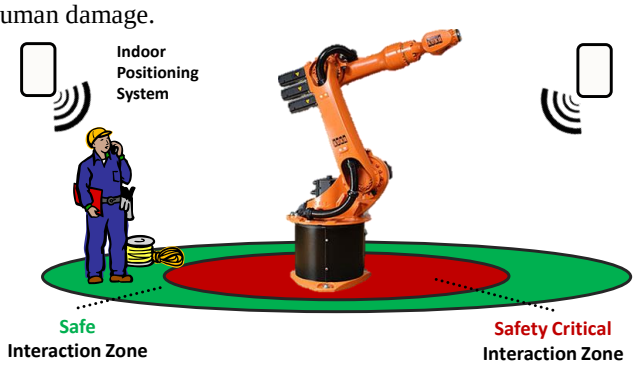

Fig. 2. Service staff approaching a robot arm and switching from the "safe zone" (green – outer circle) to the "unsafe zone" (red – inner circle), which is detected by the positioning system.

Despite all advantages of ubiquitous and mobile interaction devices in smart production environments there are also challenges that have to be addressed to enable the implementation of the before mentioned usage scenarios. Besides the need of an adequate infrastructure (e.g. indoor positioning system, wireless technology) systematic and scalable development processes are demanded to support the developers during the designs phase. To guarantee the design of usable interactive systems, user centred-development processes that involve the users during the system design are vital.

The remainder of the paper is structured as follows:

In section 2 we present the related work in the field of model-based development of context-sensitive user interfaces followed by our model-based architecture in section 3. Afterwards, in section 4 we present the software architecture of our model-interpreter prototype that is used to generate the run-time adaptive user interface. In section 5 we give a summary of the lessons learned in this project and give an outlook about challenges that have to be addressed in future work.

2. STATE OF THE ART

One basic principle of software engineering is the "separation of concerns". This principle supports the scalability of development processes and is implemented within the paradigm of model-based user interface development (MBUID). Here, multiple declarative models are used to describe different aspects (e.g., presentation, behaviour and context) of the user interface and user interaction. These models are structured within an architecture according to their level of abstraction.

One of the most recent model-based architectures is the CAMELEON Reference Framework (CRF) (Calvary et al., 2003) which describes four layers of abstraction to structure the models according to a user centred development process.

According to the CRF, many reference implementations have been proposed that allow the model-based description of user interfaces. In (Meixner et al., 2011) a model-based architecture for the design of multi-platform UIs is presented. In this approach, task models represent the initial design models in which the users tasks are specified in an hierarchical task model, specified with the Useware Markup Language (useML) (Mukasa et al., 2004; Meixner et al., 2009). Based on this model, transformations are used to derive modality independent abstract user interfaces and the platform independent concrete user interface describing the presentation and behaviour aspects of the UI. In the last step the final user interface is generated for a specific target toolkit (e.g. Java Swing, HTML).

To enable the design of context-sensitive, run-time adaptive user interfaces, new approaches have been presented that use additional context, adaption and user models to describe the user interface adaptations that should be performed during run-time. For the description of the use context extend task models like the Room-based Use Model (RUM) (Goerlich et al., 2007) (Breiner et al., 2009) and MARIA (Paternò et al., 2011) have been introduced. This model is capable of defining situation specific to the user's tasks. While in this approach the usage situations can be described on a very high abstraction level, the approach only supports an implicit description of the presentation and behavior aspects and adaptation effects that should be performed during run-time.

In the Multiple-Access Service Platform (MASP) (Blumendorf et al., 2008), explicit models for the description of the context and adaptation strategies of the UI are used which offers a greater flexibility for the developers.

Nevertheless, most of the presented approaches suffer from different problems that hinder their direct application in the domain of production automation for the design of context-sensitive universal interaction devices.

In our previous work we identified the lack of expressiveness and redundancies between the models as a critical shortcoming of the earlier approaches (Seissler et al., 2010). The limited expressiveness resulted in the generation of user interfaces with a low usability while the redundancies between the models had a negative effect on the extensibility of the model renderer which generates the run-time adaptive user interface.

In the next section we present a model-based architecture that addresses the before mentioned issues and gives the UI developers a greater flexibility in designing context-sensitive user interfaces for smart production environments.

3. A MODEL-BASED ARCHITECTURE FOR CONTEXT-SENSITIVE USER INTERFACES IN FUTURE FACTORIES

To support the developers during the design of context-sensitive user interfaces a model-based architecture that allows the separated description of the user interfaces

presentational, behavioural and adaptation aspects has been developed.
The architecture specifies three core-models that allow the explicit specification of the run-time adaptive user interface. According to the CRF the user interface is described via a modality- and platform-independent abstract user interface (AUI) model that is refined by a platform independent, concrete user interface (CUI) model. The run-time adaptations that have an effect on both models are defined in a separated adaptation model.

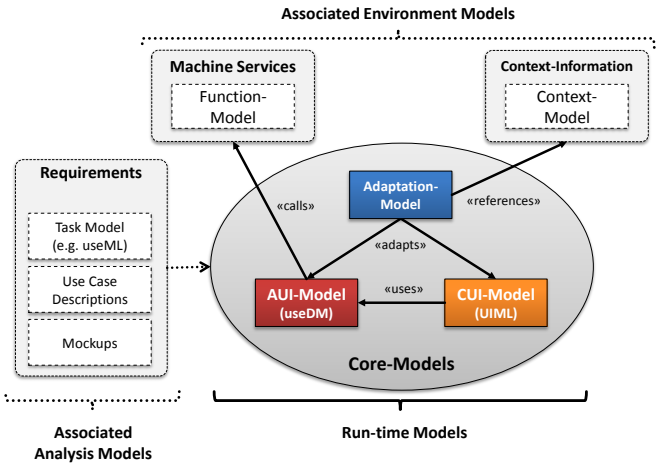

Fig. 3. The model-based architecture

Associated analysis models are used for the informal and semi-formal documentation of the user requirements to give a task-oriented perspective on the user interface.
Besides the associated analysis models the architecture consists of an explicit functional model that specifies the backend services of the machines and devices within the factory. Within the associated context model the interaction zones of the factory are described that are used to trigger the user interface adaptations. For the acquisition of the location information an external "interpretation server" (see (Stephan et al., 2010)) is used that aggregates location information from different sensors and sends the triggers to the run-time architecture.
Since the AUI, CUI and adaptation model represent the core of the model-based architecture for the design of run-time adaptive user interfaces these models are presented in the following subsections.

3.1 Abstract User Interface (AUI) Model

The abstract user interface (AUI) model is used for the modality-independent description of the interactions. For the specification of the AUI model we introduced the XML-based "Useware Dialog Modelling Language" (useDM) (Seissler et al., 2012) that extends the main concepts of the "Dialog and Interaction Specification Language" (DISL) (Bleul et al., 2004) for an effective description of context-sensitive UIs.

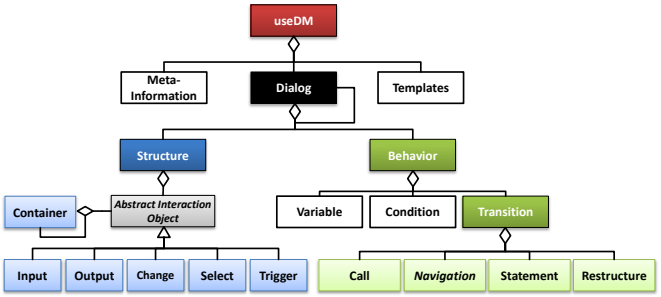

Fig. 4. The meta-model of the Useware Dialog Modeling Language

In useDM Dialogs are used to describe the static UI structures and presentational aspects. The contents of a dialog can be in turn described by six abstract and modality-independent interaction objects. Five of the abstract objects ("input", "output", "change", "select" and "trigger") are based on the "elementary use object" of useML (Goerlich et al., 2007; Meixner et al., 2011), which represent the basic modality-independent information exchange between the user and the machine. The "container" element helps to group and structure the abstract interaction objects in hierarchies. Additionally, containers allow assigning further semantics to the sub-elements. For instance, navigational objects that are grouped in a container can later be mapped onto a consistent layout by specifying one rule for all elements of this container.
While the presentation model part of useDM allows specifying the static aspects of the UI e.g. the structure and content, the dialog model part is used to describe the dynamical aspects. The dialog model part in useDM is based on an Event-Condition-Action (ECA) concept which allows the specification of holistic behaviours. The "Behaviour"-Element represents the entry-point into the dialog model and is further refined by global variables and conditions, which can be referenced in transitions. Transitions represent the core of the behavioural description. Using these elements, we can describe complex UI behaviours by means of four different action types.
The "call"-Element is used to link the UI with the field devices and machine functions, which are implemented in the functional model of the architecture and which are accessed by interfaces in the architecture.
The "statement"-Element is used for setting attribute values of interaction objects in the presentation model (e.g. title of an interaction object) as well as variable values within the dialog model.
The "navigation"-Element enables the specification of absolute and relative dialog changes in the UI. The absolute navigation allows the specification of dialog-id based navigations. On the other hand, relative navigations are used for the specification of generic navigations in reusable dialogs by using navigation symbols (e.g. "next", "parent", etc.)

3.2 Concrete User Interface Model (CUI-Modell)

After the definition of the AUI the information and functions specified in the AUI model have to be mapped into a modality-specific concrete user interface (CUI). This CUI

represents the UI in a specific target modality (e.g. graphical or vocal UI) but still abstracts from the platform-specific aspects. For a graphical UI this means that e.g. the abstract interaction objects have to be mapped onto concrete interaction objects (e.g. a "trigger"-element can be mapped onto a "button"-element) and arranged in a layout. The CUI then can be further refined by adding additional, modality-specific design information (e.g. colour schemes, pictures, icons).
For the platform-independent description of the graphical concrete UI we use in our model-based architecture the User Interface Markup Language (UIML) 4.0. Since in our architecture the CUI only is used for the refinement of the information architecture that is specified in the AUI model, the language had to be extended by an explicit mapping mechanism. This mapping mechanism is used to interlink the AUI model elements with the CUI model elements via rules. This helps to avoid redundancies between the AUI and CUI models and increases the flexibility of the model based architecture. The developers can manually specify their own mapping rules without having to change the implementation like in other concepts (see e.g. (Goerlich et al., 2007)).

3.3 Adaptation Model

For the specification of the run-time adaptions of the UI an explicit adaption model has been developed that interconnects the context-information of the associated "Interpretation Server" (Stephan et al., 2010) and the AUI- and CUI-model elements. Like in the CUI model, rules are used to specify how the situations described in the context model affect the design aspects of the UI. Since the adaptation rules can address elements of both models, adaptations on different abstraction layers of the UI can be modeled. Modality-independent adaptations, which support the context-sensitive information filtering, have an effect on the AUI model, while modality-specific adaptation rules that have an effect on the information presentation (e.g. color, size, layout), are specified for the CUI model.
While each of the three presented core models describes one aspect of the UI, they all together allow the description of the context-sensitive UI. For the run-time adaption of the user interface a model interpreter is needed. In the next section the SmartMote, a model-interpreter for the presented model-based architecture is presented.

4. SMARTMOTE – A RUN-TIME ADAPTIVE UNIVERSAL REMOTE CONTROL

To test the feasibility of the model-based architecture a prototypical model interpreter has been designed that supports the generation of the run-time adaptive UI (Seissler et al., 2012). Based on the architecture and the model interpreter we've designed a context-sensitive UI that supports the service personal in a maintenance scenario. In this scenario the context-sensitive UI is used for the interaction with a cyber-physical production system (CPPS) that has been presented by the DFKI at the international Hannover Fare (HMI) 2012. This CPPS consists of four independent production modules which demonstrate the automated production of an intelligent key-finder. The UI allows the interaction with two production modules of the CPPS and their field devices. To present the core-benefits of the context-sensitive UI the interaction with the Assembly-Module, which handles the housing and the PCB of the key-finder, and the Picking-Module, which handles the product via a robot, is presented.
For the situation-aware information presentation both production modules have been augmented via two distinct interaction zones that are modeled within the "Interpretation Server". These interaction zones are interlinked in the adaptation model with according adaptation rules that implement the run-time adaptations for the AUI and CUI model.
The result of the UI modeling is depicted in Fig 5.

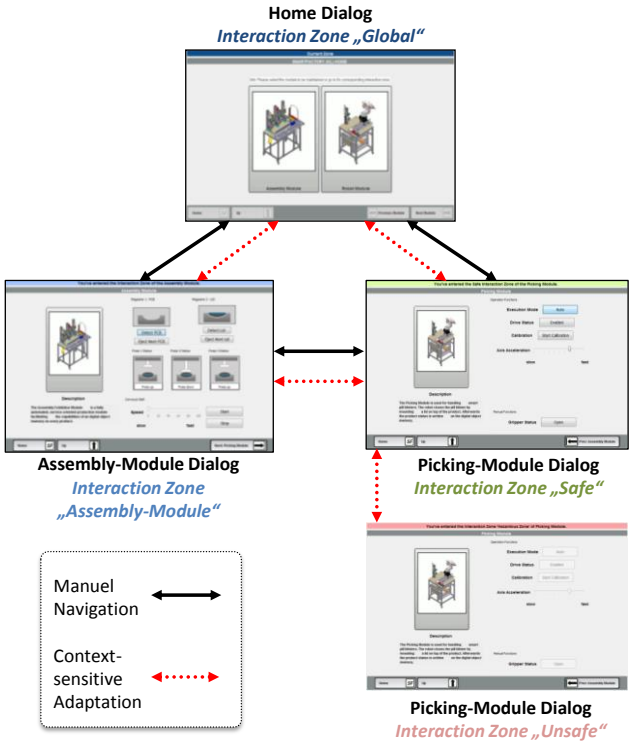

Fig. 5. The generated context-sensitve UI

The users are supported by two different adaptations that have been incorporated in the UI. A dynamical navigation between the dialogs "Assembly-Module" and "Picking-Module" has been specified which automatically presents the correct dialog to the user according to her current location. While this comforts the interaction in larger industrial settings it also has a positive effect on the safety of the mobile interaction. Additional adaptions can be used to enable/disable certain functionalities (e.g. robot arm) that demand for a continuous, visual feedback. Therefore, if the user leaves the interaction zone of the Picking Module (Interaction Zone "Unsafe") and has no direct sight on the robot, the manipulation functions are disabled (see Fig. 5).
Adaptation rules that have an effect on the CUI model have been used to give the users a feedback for the adaptations. The adaptation of the color scheme of the status bar has been used in this use case to offer feedback about the current interaction zone and the respective adaptation that has been

triggered by the system. This makes the adaptation process for the users more transparent since they can see when and in which interaction zone an adaptation has been performed.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented a model-based architecture for the development of context-sensitive user interfaces which support mobile maintenance tasks in smart production environments. Due to the use of three independent, but interlinked, user interface models the separated description of the presentation, behavioral and adaptation aspects is enabled. To show the feasibility of the architecture, a demonstrator of a model-interpreter, the SmartMote, and a prototypical context-sensitive UI for a cyber-physical production system has been developed.

To give the users a better support during the interaction with the universal interaction device, further context sources can be integrated into the architecture. One possible enhancement which seems to be fruitful is the integration of status information from the production system itself (e.g. "big data"). Having more precise information from the single field devices in the production process, the user can be triggered from the system, when there seems to be an anomaly in the system that has to be fixed before a failure occurs.

However, for the industrial application of this approach other issues like safety of the wireless communication or the user authentication that have been neglected in this concept.

ACKNOWLEDGMENTS

Our work as well as the "GaBi" project is funded by the German Research Foundation (DFG) reference number ZU 79/16-1 and -2 as well as RO 3343/1-1 and -2. Additionally we thank Malte Brunnlieb, Philipp Diebold, Markus Kleine, Thilo Rauch, Peter Reuter and Lars Scherer for their work contributing to this project.

REFERENCES

Abrams, M. et al., 1999. UIML: An Appliance-Independent XML User Interface Language. In *Proc. of the 8th Int. World Wide Web Conf*, Elsevier, 1695-1708.

Bleul, S. et al., 2004. Multimodal Dialog Description for Mobile Devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces.* (Costabile, M. (Ed.)), ACM Press, New York, NY, USA.

Blumendorf, M. et al., 2008. Multimodal user interfaces for smart environments: the multi-access service platform. In *Proceedings of the working conference on Advanced Visual Interfaces (AVI '08)*. (Levialdi, S. (Ed.)), 478–479. ACM Press, New York, NY, USA.

Breiner, K. et al., 2009. Run-Time Adaptation of a Universal User Interface for Ambient Intelligent Production Environments, In *Proc. of the 13th Int. Conf. on Human-Computer Interaction*, 663-672. Springer.

Calvary, G. et al. 2003. A Unifying Reference Framework for Multi-Target User Interfaces, *Interacting with Computers*, 15(3), 289-308.

Goerlich, D. and Breiner, K., 2007. Intelligent Task-oriented User Interfaces in Production Environments. In *Proceedings of the the workshop on Model-Driven User-Centric Design & Engineering (MDUCDE'07), 10th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine-Systems*, (Goetz Botterweck (Ed.)), CEUR-WS.org.

Meixner, G. et al., 2011. Model-Driven Useware Engineering. In *Model-Driven Development of Advanced User Interfaces : Model-driven development of advanced user interfaces*. (: Hussman, H.; Meixner, G.; Zuehlke, D. (Ed)) 1-26. Springer, Berlin.

Seissler, M. et al., 2010. Using HCI Patterns within the Model-Based Development of Run-Time Adaptive User Interfaces. In *Proceedings of the 11th IFAC/IFIPS/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, (Vanderhaegen, F. (Ed)) 477–482. IFAC-PapersOnline.

Seissler, M. et al., 2012. SmartMote – A run-time adaptive universal control device for ambient intelligent production environments : W3C Working Group Submission. URL: http://www.w3.org/wiki/images/0/06/2012-02-09_SmartMote_Architecture_useML_useDM.pdf - Last checked on October 22, 2012.

Stephan, P. et al., 2009. Evaluation of Indoor Positioning Technologies under industrial application conditions in the SmartFactoryKL based on EN ISO 9283. In *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 09)*.870–875. IFAC-PapersOnline.

Stephan, P. System architecture for using location information for process optimization within a factory of things. In *Proceedings of the Third International Workshop on Location and the Web*. 1-4. ACM Press, New York, NY, USA.

Paternò, F. et al., 2011. Engineering the authoring of usable service front ends. In *Journal of Systems and Software* 84(10), 1806–1822

Weiser, M., 1999. The computer for the 21st century. In *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3), 3-11.

Zuehlke, D., 2010. SmartFactory - Towards a factory-of-things. In *Annual Reviews in Control* 34(1), 129–138.