A Model-Based Framework for Multi-Adaptive Migratory User Interfaces

Enes Yigitbas(✉), Stefan Sauer, and Gregor Engels

s-Lab – Software Quality Lab, University of Paderborn,
Zukunftsmeile 1, 33102 Paderborn, Germany
{eyigitbas, sauer, engels}@s-lab.upb.de

Abstract. Nowadays users are surrounded by a broad range of networked interaction devices for carrying out their everyday activities. Flexible and natural interaction with such devices in a seamless manner remains a challenging problem, as many different contexts of use (platform, user, and environment) have to be supported. In this regard, enabling task continuity by preserving the user interface's state and adapting it to the changing context of use can help to improve user experience despite possible device changes. The development of such multi-adaptive migratory user interfaces (MAMUIs) involves several challenges for developers that are partially addressed by frameworks like CAMELEON-RT. However, supporting the development of user interfaces with adaptation and migration capabilities is still a challenging task. In this paper, we present an integrated model-based framework for supporting the development of MAMUIs.

Keywords: Model-Based user interface development · Adaptive user interface · User interface migration

1 Motivation

To day users are surrounded by a broad range of networked interaction devices (e.g. mobile phones, laptops, tablets, smartwatches, terminals etc.) for carrying out their everyday activities. Allowing for flexible and natural interaction with such devices in a seamless manner remains a challenging problem, as many different contexts of use (platform, user, and environment) have to be supported.

Figure 1 shows a usage scenario of such a distributed interactive system. Purchasing a train ticket is carried out by the use of different devices which provide different interaction interfaces. In this scenario, the ticket purchase is first prepared on a home PC entering the reservation dates (date and time for round-trip). In transit, the user is able to book additional services (luggage service, seat reservation, hotel reservation at destination) via smartphone. Finally, the printing of the ticket is done at the ticket machine. Focusing on this example scenario, one can see that different devices can be used to access and modify the information provided by the user interface. In this regard, a device change can cause a context change, which has to be considered for an intuitive and flexible task continuation. The major problems in developing cross-device user interfaces for such scenarios are: lack of efficient development methods for generating multiple user interfaces for different platforms, poor adaptation to the context of

© Springer International Publishing Switzerland 2015
M. Kurosu (Ed.): Human-Computer Interaction, Part II, HCII 2015, LNCS 9170, pp. 563–572, 2015.
DOI: 10.1007/978-3-319-20916-6_52

use, and inadequate support for seamless migration of user interfaces across different devices.

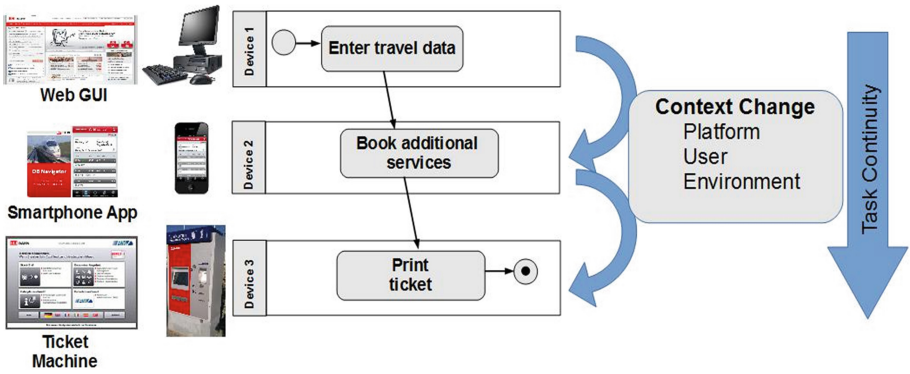

Fig. 1. Example scenario

Regarding the scenario mentioned above, enabling task continuity by preserving the user interface's state during migration and adapting it to the changing context of use can help to improve user experience despite possible device changes. The development of such multi-adaptive migratory user interfaces (MAMUIs) is partially addressed by frameworks like CAMELEON-RT [3]. However, the following factors remain challenging:

- Multi-platform capability: Increase efficiency of multi-platform user interface development across heterogeneous computing platforms (Windows, iOS, Android, Windows Phone etc.).
- Adaptation capability: Establish adaptation mechanisms that enable (semi-) automatic user interface adaptation as reaction to context changes.
- Migration capability: Share an application's user interface and client-side logic across multiple heterogeneous devices in order to support distributed transactions.

In this paper, we present an integrated model-based framework for supporting the efficient development of MAMUIs that enable task continuity and adaptivity to context changes. The paper is structured as following: First, we present related work in the area of model-based development and user interface migration as well as user interface adaptation. Then, we present architectural patterns as basic solution concepts for addressing the challenges of MAMUI development. Based on these architectural patterns, we describe our integrated model-based development framework for MAMUIs. Finally, we conclude with a summary and an outlook for future research work.

2 Background

Focusing on the topic of model-based development of multi-adaptive migratory user interfaces (MAMUIs), multiple aspects have to be taken into account: The model-based development approach for creating UIs and existing approaches for the migration and adaptation of UIs.

2.1 Model-Based Development

Model-based development methods have been discussed in the past for various individual aspects of a software system and for different application domains. This applies to the development of the data management layer, the application layer or the user interface layer [8]. The CAMELEON Reference Framework (CRF) [6] provides a unified framework for model-based and model-driven development of UIs. CAMELEON-RT [3] is an extended version of this framework which represents an abstract reference model for developing the run-time infrastructure for distributed and migratable user interfaces. There are already different approaches such as [4] or [11] which propose model-based development of UIs. These approaches mainly focus on model-based development and its technological implementation aspects. However, aspects like task continuity under device changes, which increase the flexibility of using different UIs, are not sufficiently integrated in the model-based development process.

2.2 UI Migration

UI migration is an important concept to transfer a UI or parts of it from a source device to a target device in order to carry a UI and its state across different devices and to enable mobile, distributed interaction. An example solution for partial Web migration to mobile devices is presented in [7]. Refinements of this concept and architecture proposals for UI migration are described in [15] and [16]. There are also model-based approaches for the dynamic distribution of UIs as described in [12] for example. However, an integrated UI migration process, supporting adaptivity to context changes is not fully covered yet.

2.3 UI Adaptation

Adaptive UIs have been promoted as a solution for context variability due to their ability to automatically adapt to the context-of-use at runtime. Norcio and Stanley consider that the idea of an *adaptive UI* is straightforward since it simply means [13]: "The interface should adapt to the user; rather than the user must adapt to the system." Based on [2] we can generally differentiate between the following types of adaptive UIs:

- **Adaptable User Interfaces** allow interested stakeholders to manually adapt the desired characteristics; example: a software application that supports the manual customization of its toolbars by adding and removing buttons.
- **Semi-Automated Adaptive User Interfaces** automatically react to a change in the context-of-use by changing one or more of their characteristics using a predefined set of adaptation rules. For example: an application can use a sensor to measure the distance between the end-user and a display device, and then trigger predefined adaptation rules to adjust the font-size.
- **Fully-Automated Adaptive User Interfaces** can automatically react to a change in the context-of-use. However, the adaptation has to employ a learning mechanism, which makes use of data that is logged over time. One simple example is a software application, which logs the number of times each end-user clicks on its toolbar

buttons and automatically reorders these buttons differently for each end-user according to the usage frequency.

A classification of different adaptation techniques was introduced by Oppermann [14] and refined by Brusilovsky [5]. UIs with adaptation capabilities have been proposed in the context of various domains and there are also proposals for integrating adaptive UI capabilities in enterprise applications (e.g. [1]). Still, the aspect of automated adaptation is not sufficiently covered during the migration process of a UI regarding to context changes.

3 Architectural Patterns for MAMUIs

In order to support the development of MAMUIs we have identified basic architectural patterns to address the identified challenges: Multi-platform, Adaptation and Migration capabilities.

3.1 Multi-Platform Capability

For increasing the efficiency of multi-platform user interface development it is important to overcome the process of implementing interfaces for M heterogeneous devices with N different contexts while maintaining N*M architectural models and code for all the variants of the same UI. The model-based development process proposes a stepwise approach. At the beginning, models of abstract user interfaces are specified that are then transformed to models of concrete user interfaces. Eventually, the final user interfaces are generated by model-to-code transformations. Such a modeling and generation process is described by the CAMELEON Reference Framework which is illustrated in Fig. 2. The top layer *Task & Concepts* includes a task model that is used for the hierarchical description of the activities and actions of individual users of the user interface. The abstract user interface (*AUI*) is described in the form of a dialogue model that specifies the user's interaction with the user interface independent of specific technology. The platform specific representation of the user interface is described by the concrete user interface (*CUI*), which is specified by a presentation model. The lowest layer of the framework is the final user interface (FUI) for the target platform. The vertical dimension describes the path from abstract to concrete models. Here, a top-down approach is followed, in which the abstract description of relevant information about the user interface (AUI) is enriched to more sophisticated models (CUI) through model-to-model transformations (M2 M). Subsequently, the refined models are transformed (model-to-code transformation, M2C) to produce the final user interface (*FUI*). Based on this architectural pattern, it is possible to enable multi-platform capability for the different UIs that are generated during the development process.

3.2 Adaptation Capability

While the architectural pattern above described, supports multi-platform UI generation, it is not sufficient to develop adaptive UIs, because there are no means to model

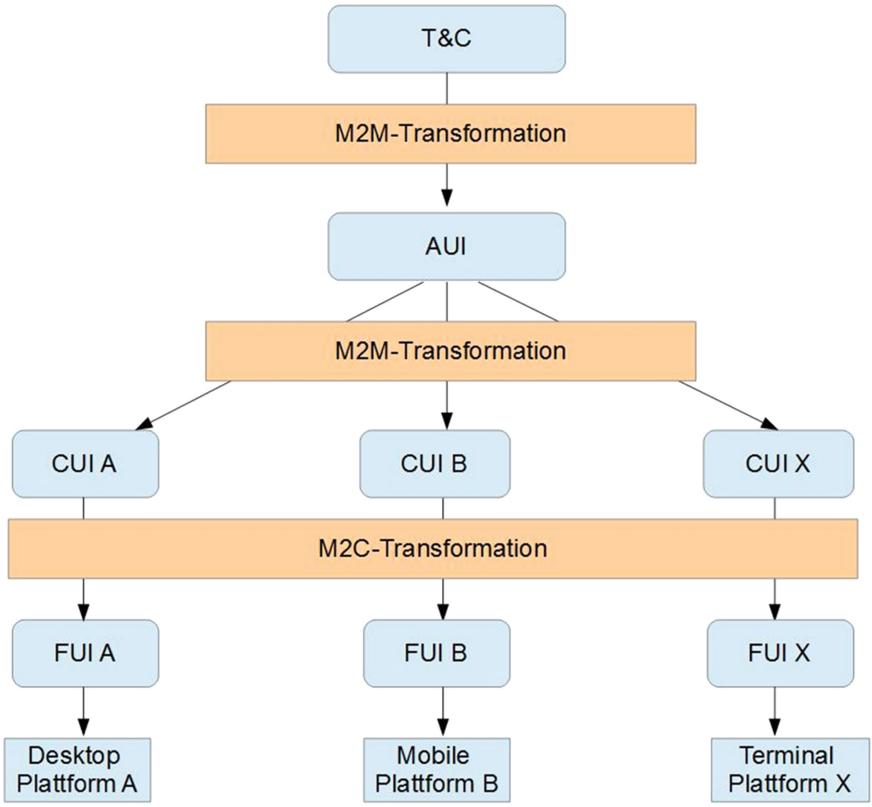

Fig. 2. Simplified CAMELEON reference framework

adaptivity. Nevertheless, the model-based development process offers flexibility to extend it for the development of adaptive components. In this context we have analyzed different architectural concepts for self-adaptive systems such as MAPE-K by Kephart and Chess [9] and the 3-layer reference architecture by Kramer and Magee [10]. As a result of this analysis, we have identified the need for an architectural pattern for representing the adaptation process (see Fig. 3). Such an architectural pattern for UI adaptation can be characterized by an *Adaptation Manager* that monitors the *Managed/Adaptive UI.* The *Managed/Adaptive UI* can run on different platforms (PC, smartphone and terminal). The *Adaptation Manager* consists of five main components that work in the following way: The *Monitor* component is responsible for observing the context information. Context information changes are then evaluated by the *Analyze* component to decide whether adaptation is needed. If so, the planning of an adaptation schedule is done by the *Plan* component. Finally, the adaptation operations are performed by the *Execute* component, so that an adapted UI can be presented. The *Knowledge* base is responsible for storing data that is logged over time and can be used for inferring future adaptation operations.

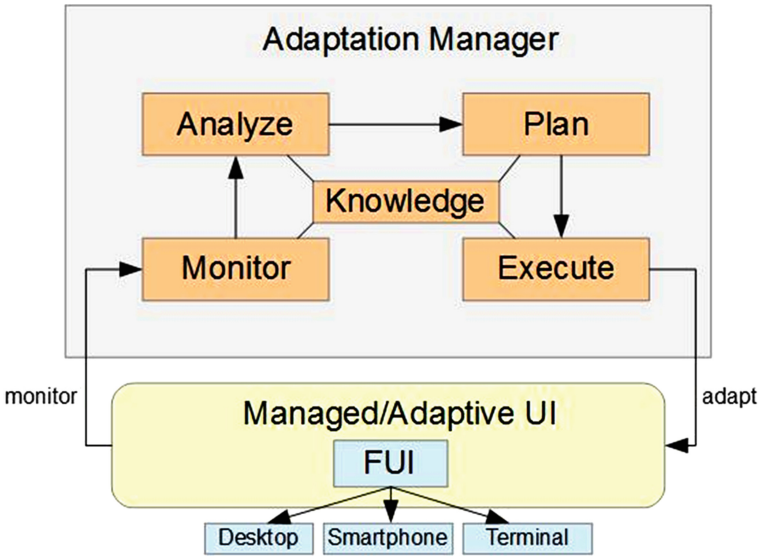

Fig. 3. UI adaptation manager

3.3 Migration Capability

For sharing an application's user interface and client-side logic across multiple heterogeneous devices and in order to support distributed transactions it is important that the UI can be transferred from a source to a target device. For this reason, we have defined a further architectural pattern called UI migration (see Fig. 4).

In order to support this in a seamless manner, we have to enable task continuity during the migration process so that the state of the UI is carried over and the presentation is adapted to the target device and its context. For reaching this goal, the source platform triggers a migration request to the *Migration Server*. This can be done manually by request of the user or automatically based on context information events, like for example, low battery status of a mobile device. The *Migration Server* accepts the migration request to transfer the current *FUI A* to the selected target platform. For supporting the migration process, the *Migration Server* consists of four main components. The *Task Mapping* component is responsible for determining which activities of the task model are supported by the target platform. After establishing a mapping on the task level, the *State Mapping* component captures the state of the migrating *FUI A*. This state is the result of the history of user interactions with the application including previous input data. The *CUI Redesign* component provides a mapping on the CUI level by rearranging the CUI model of the migrating UI for the special needs of the target platform. It considers platform information like display size or resolution for this purpose. Finally, the *FUI Adaptation* component returns as a result of the *Migration Server* a *Context-adapted FUI A'* that is activated on the target platform.

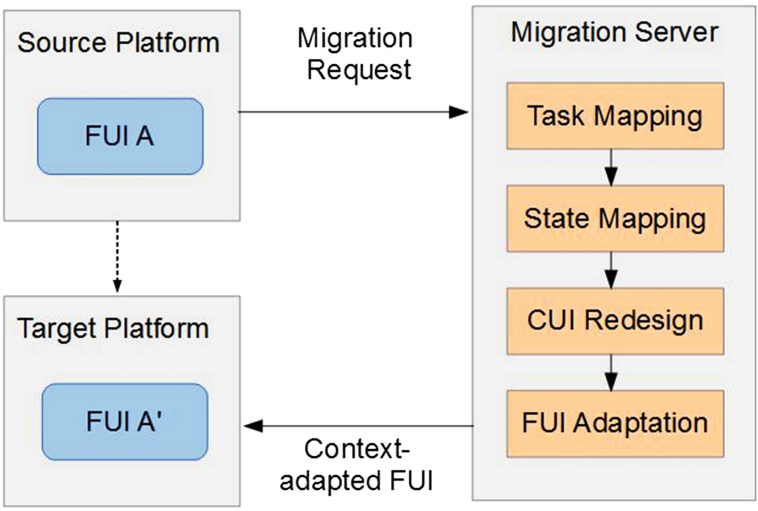

Fig. 4. UI migration

4 Model-based Framework for MAMUIs

In the previous section, we have presented different architectural patterns for developing MAMUIs. While these patterns address basic solution concepts for tackling the different challenges, it is important to design an integrated framework which combines the several aspects of multi-platform capability, adaptation capability and migration capability. For this reason, we propose our MAMUI Reference Framework which supports the development of migratory user interfaces that can extend across a variety of devices and are adaptive to context changes like platform, user and environment. The MAMUI Reference Framework which is depicted in Fig. 5 consists of four main parts: *UI Generator*, *Adaptation Manager*, *Migration Server* and *Context Manager*.

The *UI Generator* is responsible for generating the final user interfaces (FUIs) for the different platforms (Desktop, Mobile, Terminal, etc.). The generation process is based on the CAMELEON Reference Framework where a transformational approach is preceded. At the beginning of the transformation process, task models are created which describe the activities and actions of individual users of the user interface. The task models are then transformed to abstract UI models (AUI) which describe the user's interactions with the user interface independent of a specific technology. In a next transformation step, the concrete UI models (CUI) for the different platforms are created. Finally, by using model-to-code transformations the final user interfaces (FUIs) are generated.

The *Adaptation Manager* is responsible for monitoring the adaptive FUIs and adapting them to the different kinds of context changes. In order to support adaptation mechanisms at different levels of the CAMELEON Reference Framework, the *Adaptation Manager* consists of three adaptation layers. *Task Feature Adaptation* enables changes in the task models such as, for example, minimizing the task feature set based

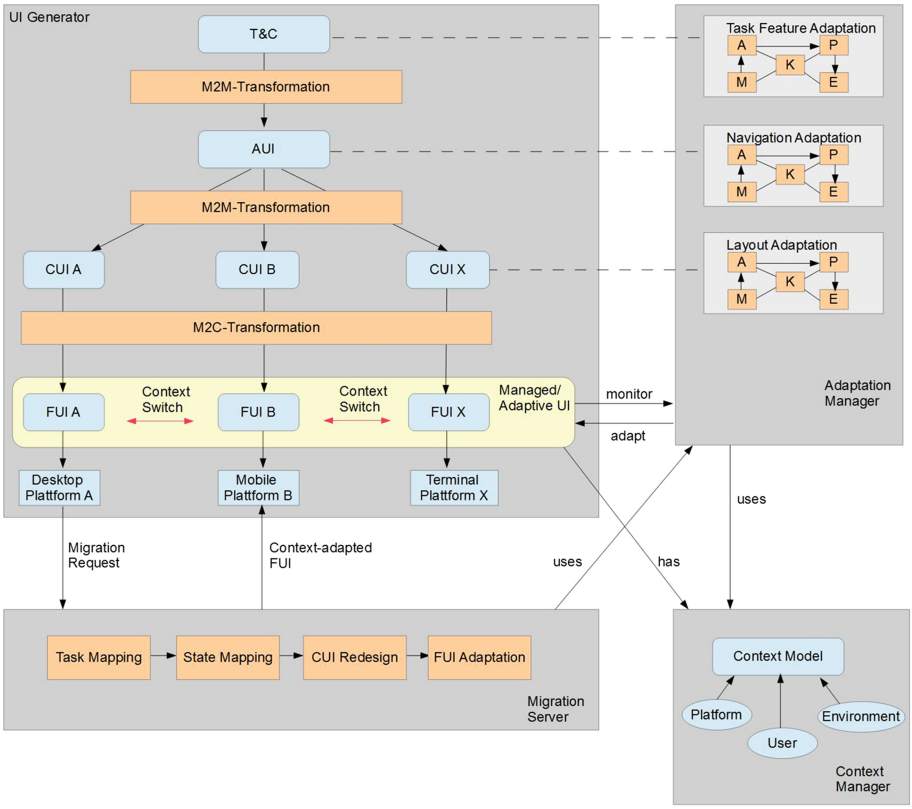

Fig. 5. MAMUI reference framework

on the context information. This means for instance that task activities which are not executable on particular platforms or for some predefined user roles are not represented in the FUI. Similarly, the adaptation layer *Navigation Adaptation* is responsible for changing the navigation flow of the FUI by manipulating the AUI model. Based on the context information, it is also possible to perform changes in the CUI models in order to reach *Layout Adaptation*. For specifying the different changes in the adaptation process, the *Adaptation Manager* makes use of an adaptation model. The adaptation model is encoded in the adaptation layers. In the adaptation model different adaptation rules based on the ECA (Event-Condition-Action) paradigm and the *Context Model* are defined, which are evaluated at runtime to react to the context changes.

While a final user interface (FUI) is running on a particular platform, a context switch may happen by changing the device. For this purpose, a *Migration Server* can accept a migration request from a source platform for migrating the current FUI to a target platform. In order to support the migration process, several steps are provided by the Migration Server as described in detail in Sect. 3.3. First, a task mapping between the source and target platform is done. This is necessary in order to adjust the different task activities so that executable tasks are selected for the target platform. After that, a

state mapping of the user interfaces is established, so that preconfigured UI features and input data are carried over to the target platform. In the next step, the CUI model of the current platform is redesigned in order to adapt it for the target platform considering the context changes. For reaching this goal, the *Migration Server* makes use of the described layers of the *Adaptation Manager*. Finally, the context-adapted FUI is activated for the target platform.

The *Context Manager* operates on a *Context Model* that is divided into a *Platform*, *User* and *Environment* model. The *Context Manager* provides useful context information for the Managed/*Adaptive UI*, so that the *Adaptation Manager* is able to observe context changes that are addressed by related adaptation rules defined in the adaptation model.

With the interplay of its described main components, the proposed MAMUI Reference Framework provides a basic solution concept to support multi-platform, adaptive and migratory UI development. In our ongoing research work, we are currently developing an integrated development environment to address the main parts of the MAMUI Reference Framework. This will include the modelling and transformation of the core UI models at different levels and also the modelling of aspects like adaptation and migration. Future work will include the evaluation of our integrated development environment based on practical example scenarios.

5 Conclusion and Outlook

This paper presents an integrated model-based framework for supporting the development of multi-adaptive migratory user interfaces (MAMUIs). We have referred to a cross-device user interface usage scenario from practice, which served as a basis to show the different challenges in this context, e.g. multi-platform, adaptation and migration capability. In order to address these challenges, we have described different basic solution patterns for supporting the development of MAMUIs. Based on these architectural patterns, we have presented our integrated model-based MAMUI Reference Framework. Our future work will focus on the development and evaluation of a modelling and execution workbench based on the proposed MAMUI Reference Framework.

References

1. Akiki, P.A., et al.: Integrating adaptive user interface capabilities in enterprise applications. In: Proceedings of the 36th International Conference on Software Engineering (ICSE 2014), pp. 712–723. ACM (2014)
2. Akiki, P.A., et al.: Adaptive model-driven user interface development systems. ACM Comput. Surv. **47**(1), 1–33 (2014). Article 9
3. Balme, L., Demeure, A., Barralon, N., Calvary, G.: CAMELEON-RT: a software architecture reference model for distributed, migratable, and plastic user interfaces. In: Markopoulos, P., Eggen, B., Aarts, E., Crowley, J.L. (eds.) EUSAI 2004. LNCS, vol. 3295, pp. 291–302. Springer, Heidelberg (2004)

4. Botterweck, G.: A model-driven approach to the engineering of multiple user interfaces. In: Kühne, T. (ed.) MoDELS 2006. LNCS, vol. 4364, pp. 106–115. Springer, Heidelberg (2007)
5. Brusilovsky, P.: Adaptive Hypermedia. In: User Modeling and User-Adapted Interaction, vol. 11, pp. 87–110. Kluwer Academic Publishers, March 2001
6. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A unifying reference framework for multi-target user interfaces. Interact. Comput. **15**, 289–308 (2003)
7. Ghiani, G.; Paternò, F.; Santoro, C.: On-demand cross-device interface components migration. In: Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI 2010), pp. 299–308. ACM (2010)
8. Hussmann, H., Meixner, G., Zuehlke, D. (eds.): Model-Driven Development of Advanced User Interfaces. Springer, Heidelberg (2011)
9. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
10. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: Proceedings of 2007 Future of Software Engineering (FOSE 2007), IEEE Computer Society, Washington, DC, USA, pp. 259–268 (2007)
11. Link, S., Schuster, T., Hoyer, P., Abeck, S.: Modellgetriebene Entwicklung grafischer Benutzerschnittstellen (Model-Driven Development of Graphical User Interfaces). i-com 6, Nr. 3, Oldenbourg, München, pp. 37–43 (2008)
12. Martinie, C., Navarre, D., Palanque, P.: A multi-formalism approach for model-based dynamic distribution of user interfaces of critical interactive systems. Int. J. Hum.-Comput. Stud. **72**, 77–99 (2014)
13. Norcio, A.F., Stanley, J.: Adaptive human-computer interfaces: a literature survey and perspective. IEEE Trans. Syst. Man Cybern. **19**, 399–408 (1989)
14. Oppermann, R.: Individualisierte systemnutzung. In: Paul, M. (ed.) GI – 19. Jahrestagung, Computergestützter Arbeitsplatz, pp. 131–145. Springer, Heidelberg (1989)
15. Paternò, F., Santoro, C., Scorcia, A.: Ambient intelligence for supporting task continuity across multiple devices and implementation languages. Comput. J. **53**(8), 1210–1228 (2010)
16. Yanagida, T., Nonaka, H.: Architecture for migratory adaptive user interfaces. In: Computer and Information Technology, CIT 2008, pp.450-455 (2008)