Flexible & Adaptive UIs for Self-Service Systems

Enes Yigitbas[1], Stefan Sauer[1]

University of Paderborn, s-lab – Software Quality Lab[1]

Abstract

Self-service systems are technically complex and provide products and services to end users. Due to the heterogeneity of the users of such systems and their short residence time, the usability of a system's user interface (UI) is of great importance. Currently, an intuitive and flexible usage is often limited because of the monolithic architecture of existing self-service systems. Furthermore, today's self-service systems represent the one-and-only endpoint of communication with a customer when processing a transaction. The integration of the customer's personal computing devices, like desktop PC, notebook, and smartphone is not sufficiently covered yet. In order to tackle these problems, we have established a methodology for developing adaptive UIs for multi-channel self-services where a customer may, for example, start a transaction on a PC at home, modify it with the smartphone, and finally finish it at a self-service terminal. In this paper we describe our integrated model-based approach for the development of adaptive user interfaces for distributed multi-channel self-service systems and show its applicability in practice based on an exemplary case study.

1 Introduction

Due to new interaction techniques possible today (e.g. multi-touch or tangible interaction) and distributed interfaces transcending the boundaries of a single device, software developers and user interface designers are facing new challenges. It is no longer sufficient for a self-service system to provide a single “one-size-fits-all" user interface. By nature, customers of self-service systems like automated teller machines, ticketing machines, or postal self-service kiosks comprise a very heterogeneous group of users. Nearly everybody, from the young to the very old, from the technically skilled to the completely unskilled, is a potential user of such systems. Some of these users may have cognitive or physical disabilities regarding vision, hearing, or may be sitting in a wheelchair. Today's personal computing devices provide a level of convenience and user experience, which customers also expect from public self-service systems. Customers do not only want to finish the task at hand within the shortest amount of time, they also would like to have a personalized experience and want to enjoy their interaction with the system. One possible solution is the development of user interfaces, which are personalized in their appearance and which are adaptive to the user's

interaction with the system and the context of use. To make things even more complex, a self-service system should no longer represent the one-and-only endpoint of communication with a customer when processing a transaction. Instead, the customer's personal computing devices, like desktop PC, notebook, and smartphone may all be involved in the process. For example, a customer may start a transaction on a PC at home, modify it with the smartphone, and finally finish it at a self-service terminal. Figure 1 shows such a distributed, networked multi-channel system. The purchase of a ticket is initiated by selecting an appropriate train connection at the PC at home. On the way to the train station, additional seat reservation is booked and arrangements for the luggage are made on a smartphone. Finally, the ticket is printed at the self-service ticket machine.

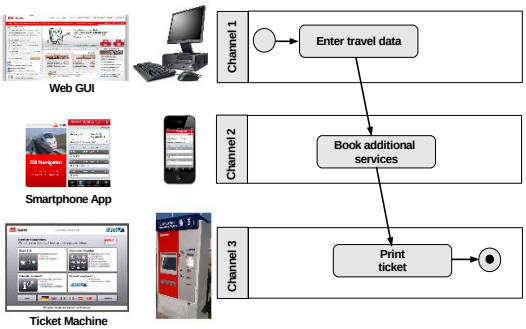

Figure 1: Example Scenario

In existing self-service systems integration of additional channels (Smartphone, Tablet, etc.) with new interaction modalities to interact with a self-service system is often not sufficiently supported. Therefore it is necessary to develop flexible and easy to use user interfaces to improve the monolithic architecture towards a multi-channel system offering

- distributed business and interaction processes enabling different interaction modalities (e.g. graphics, speech, gestures),
- multi-platform capability for integration and usage of heterogeneous devices,
- adaptation of system functionality and user interfaces.

Based on the described example scenario, one can find summarized that there is a need for an integrated model-based development approach in the area of self-service systems. This should include different aspects of a software system, such as functionality, the user interface as well as adaptation and provide a suitable development process for industry.

2 Background and Related Work

Focusing on the topic of model-based user interface development for adaptive self-service systems, multiple topics have to be taken into account: The possibilities and different levels of adapting a system and frameworks for implementation.

2.1 Adaptation

Developing software with a user interface is getting more and more interactive and complex. Developers require a thorough understanding of the users of a system and their needs and pain points to create an adequate user interface. With self-service systems in mind, the time users spend interacting with the system is rather short. Therefore, the interface should be as simple as possible, while providing the necessary information, working within existing processes and matching the needs of a wide range of users (e.g. young people, older people, people with special needs). One possible solution is to create user interfaces with a certain level of adaptation. A classification of different adaptation techniques were introduced by Oppermann (Oppermann 1989) and refined by Brusilovsky (Brusilovsky 2001). For our approach, we distinguish between and define three levels of adaptation, based on the agent that is initiating the process of adaptation:

Adaptable UIs (manual adaptation): The user is able to customize respectively individualize the user interface through layout adaptation (e.g. arranging information fields in a dashboard) or navigation adaptation (e.g. creating shortcuts for often used tasks or views).
Adaptive UIs (semi-automatic adaptation): Due to some sort of authentication, the system is able to identify its users and support them with suggestions (e.g. Amazons recommendation system about potential products of interest (content adaptation)). In addition, it is possible that the system detects if a user has difficulties in achieving his goal. In such a case, the system is able to support a user with suggestions concerning alternative workflows. At any time, a user is able to decide whether or not to accept a suggestion of the system.
Self-adaptive UIs (automatic adaptation): Self-adaptive UIs are based on systems with embedded intelligence. These sensor-equipped systems use a knowledge base or learning system to analyze the behavior of a user in front of it. Thus, it automatically changes some parts or even the whole UI (e.g. adjusting the physical height of a display depending on the size of a user (hardware adaptation), switching to audio guidance for blind people (input/output adaptation), reacting to the emotional status of a user (business process adaptation), rearranging the navigation according to the cultural context (context adaptation), supporting user with interaction problems (workflow adaptation)).

UIs with adaptation capabilities were proposed in the context of various domains (Stephanidis et al. 1998, Jameson 2002). Besides works dealing with the quality of adaptive UIs (Gajos et al. 2008, Lavie et al. 2010), there are also proposals for integrating adaptive UI capabilities into enterprise applications (e.g. Akiki et al. 2014). Nevertheless approaches for the integration of adaptive UIs in the application domain of self-service systems are missing.

2.2 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 technical functionality or for the development of a user interface (Hussmann et al. 2011). The CAMELEON Reference Framework (CRF) (Calvary et al. 2003) provides a unified framework for model-based and model-driven development of user interfaces. There are already different kinds of work,

which propose a model-based development of user interfaces (Botterweck 2006; Link et al. 2008). These approaches mainly focus on model-based development and their technological implementation. However, aspects of self-adaptive software systems for increasing the flexibility of user interfaces for heterogeneous user groups are not sufficiently integrated with the model-based development process.
Existing architectural concepts for self-adaptive systems such as the MAPE-K (Kephart & Chess 2003) and the 3-layer reference architecture (Kramer & Magee 2007) describe the logical concepts for the implementation of adaptive software. Based on the MAPE-K approach existing frameworks like Rainbow (Garlan et al. 2003) or StarMX (Asadollahi et al. 2009) provide a more refined architecture for implementing self-adaptive systems. Geihs et al. develop a framework for the development of self-adaptive and reconfigurable software based on the distributed software architecture of the EU project MUSIC (Geihs et al. 2009). The focus is in particular on the creation of adaptive mobile applications that offer location-based services to the user.
In our approach for model-based user interface development for adaptive self-service systems, we follow a development methodology that integrates several aspects of a software system, such as functionality, the user interface as well as adaptation. Model-based approaches in the field of self-adaptive systems have to be combined with ideas from the field of model-based development of user interfaces.

3 Method and Results

In cooperation with an industrial partner we develop a new methodology for the model-based development of user interfaces for distributed self-service systems. To implement adaptive user interfaces for distributed self-service systems we have designed a target architecture which pays special attention to aspects of adaptation and integrates them into the model-based development process for user interfaces. Figure 2 shows the process/logical view of our target architecture.

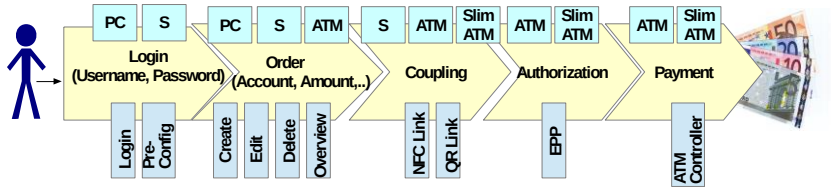

Figure 2: Architecture: Process/Logical View

The workflow of processing a transaction on a multi-channel self-service system is divided into five main steps. In the first step the end user performs a login using his desktop pc (PC) or smartphone (S) to start the interaction with the self-service system. For this purpose the profile manager requests his user name and password and checks whether existing entry data can be found in the user database. After successful login, the customer is able to create, edit and delete his orders on the desktop pc, smartphone or at the ATM. If the order is completed, the coupling between involved devices in the transaction process is established. This is

coordinated by the connection manager which makes use of NFC and QR Code technology to couple corresponding devices like the smartphone and the ATM. In the next step authorization of the user is approved using an encrypting PIN pad (EPP). Finally, the outcome or service of the self-service system is delivered by the ATM or Slim ATM. A Slim ATM is an extremely stripped-down version of an ATM, which cannot be operated without a smartphone. It only consists of a very small display to show simple messages to the user, a pin-pad, and a cash dispenser unit. The advantage of a Slim ATM over a regular ATM is a significant reduction in cost and energy consumption. Furthermore, since user interaction with the Slim ATM is reduced to a minimum, the time needed to wait in line for other customers to complete their transactions is reduced.

In our methodology, we are pursuing a model-based development approach based on the CAMELEON Reference Framework (CRF). In order to support the analysis and design phase during the development of distributed, adaptive user-interfaces we have designed a target architecture which pays special attention to aspects of adaptation and integrates them into the model-based development process for user interfaces. Figure 3 shows the target architecture of our solution based on the CRF approach. At the level of conceptual modeling (Computation Independent Model, CIM) a task model and a user model (representing different user groups or individuals) are created as input for the creation of an abstract model of the user interface at the level of platform-independent modeling (Platform Independent Model-PIM). Another conceptual model describes the context of use in the form of contextual factors, such as the localization of the user (context model). This model is used together with a platform model to implement the model-to-model transformation from the abstract user interface model (Abstract UI-model) into the concrete user interface model (Concrete UI-model) at the level of platform-specific modelling (Platform-Specific Model, PSM). This translation step will be supported by the use of appropriate tools. The concrete user interface model consists of a set of coupled partial models for the respective platforms. With the aid of specific generators and interpreters required for a concrete platform, a suitable user interface will be generated from the respective sub-model, which can then be run on this platform.

This model-driven development approach will also support the dynamic adaptation of the user interface at runtime. For this purpose, an adaptation model is created in addition to a monitoring concept that describes the adaptation of the user interface (as well as the functionality coupled thereto). From the resulting adaptation models the adaptation manager is derived. This is a software component that observes the adaptable software and controls the adaptation according to the adaptation model. For implementing the adaptation steps our target architecture makes use of IBM's MAPE-K loop (Kephart & Chess 2003). The main components of the MAPE-K loop, namely monitor, analyze, plan, execute and knowledge are embedded in our target architecture to provide runtime adaptation. For managing the adaptation process we mapped the MAPE-K components to corresponding components in our target architecture. The monitoring step is implemented by the *Context Monitor* which continuously observes context change information on the context model. This information is then analyzed by the *Context Evaluator* and stored in the *Knowledge* base by the *Data-Logger* component. With the help of the *Knowledge* base an adaptation plan is scheduled by the *Adaptive Engine*, so that corresponding Adaptation changes can be performed by a *UIDL Converter*. The effect of the planned adaptation operations are finally achieved by the *UI Renderer* which generates the adapted final user interfaces.

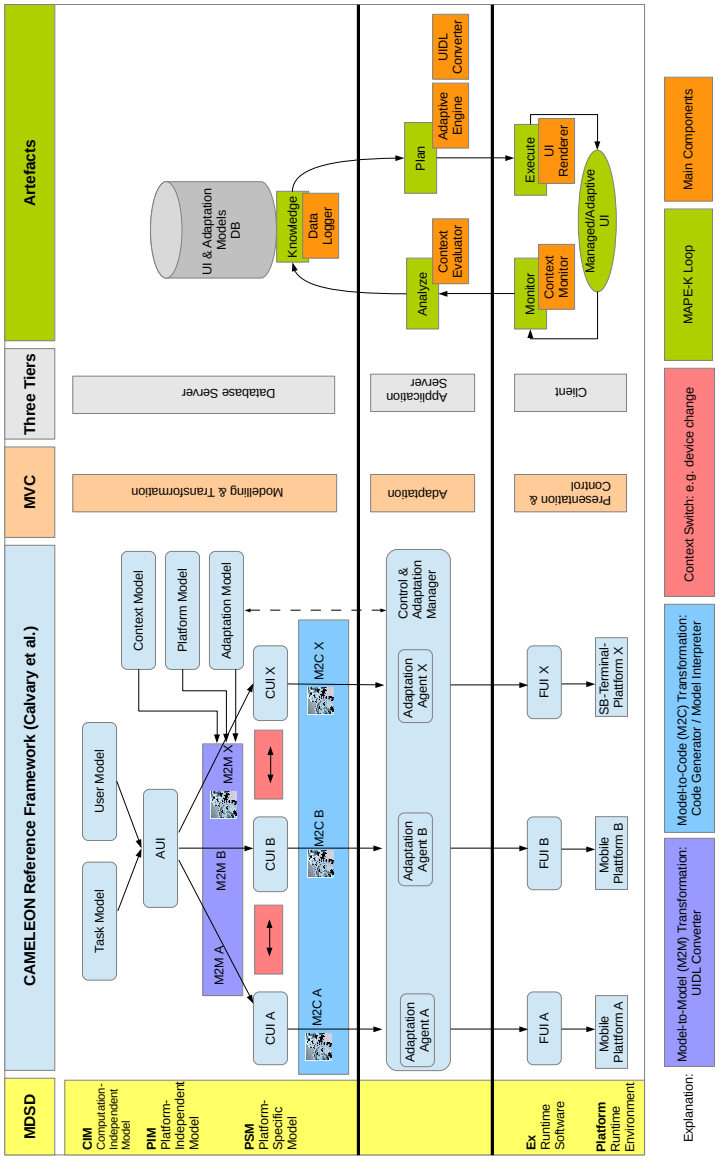

Figure 3: Architecture: Model-based UI Development View

In order to exemplify the idea of adaptation in our target architecture, an example walkthrough is depicted in Figure 4. It shows an exemplary process instance which is processing a transaction on a multi-channel self-service system. After login of the user Elisabeth with her password the *Context Monitor* records relevant context information which is needed to perform the transaction and to provide an appropriate user interface. In this case

her age, transaction starting time, and her device type with the corresponding operating system are observed.

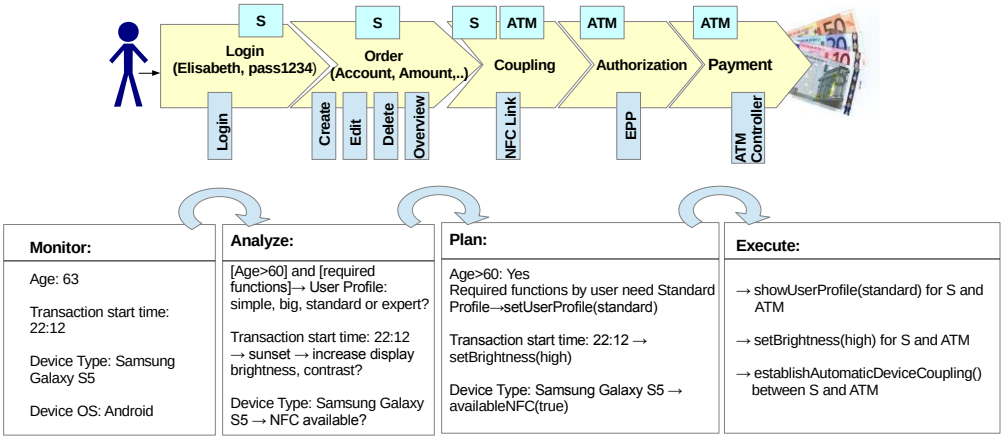

Figure 4: Adaptation: Example Walkthrough

In a next step the *Context Evaluator* analyzes the context information in order to infer possible information for adaptation purposes. In this example the age of the user and often required/used functions are analyzed to decide whether she needs a simple, big, standard or expert user interface profile for the current transaction. The transaction start time is analyzed to evaluate if an increase of the display brightness is needed and based on the used device type it is possible to gain information on the operating system and whether the device supports for example NFC coupling with the ATM. After evaluation of these factors an adaptation schedule is planned. In our process example the user profile is set to the standard mode because Elisabeth often makes use of the functions create, edit, delete and overview. Based on other context information the adaptation schedule also contains a rule to set the brightness to high and to establish automatic device coupling between her smartphone and the ATM.

4 Conclusion and Outlook

In this paper, we presented a concept for the efficient and effective development of adaptive user interfaces. By analyzing use cases and current practice in the application domain of self-service systems we identified the need for tools and techniques, which support the creation of user interfaces that adapt themselves to the user, the environment and the context of use. Based on the requirements a model-based approach for the implementation of adaptive flexible user interfaces was presented for interactive self-service systems. For this purpose we described our target architecture, which extends the CAMELEON reference framework to the aspect of adaptation. Our model-based development process is currently not completely supported by tools and further work will be done on the implementation of tools for the creation of the required models and to automate their translation into a concrete UI expression.

References

Akiki, P. A. et al.: Integrating adaptive user interface capabilities in enterprise applications. In Proc. of the 36th Int. Conf. on Software Engineering (ICSE 2014)

Asadollahi, R. et al.: StarMX: A Framework for Developing Self-managing Java-based Systems. In: Proc. of the Workshop on Software Engineering for Adaptive and Self-Managing Systems, (2009)

Botterweck, G.: A Model-driven Approach to the Engineering of Multiple User Interfaces. In: Proc. of the 2006 Int. Conf. on Models in Software Engineering, Springer, Berlin/Heidelberg (2006)

Brusilovsky, P.: Adaptive Hypermedia. In: User Modeling and User-Adapted Interaction 11, 1-2 (March 2001)

Calvary, G. et al.: A Unifying Reference Framework for Multi-target User Interfaces. In: Interacting with Computers, pp. 289-308 (2003)

Gajos, Z. K. et al.: Predictability and accuracy in adaptive user interfaces. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08), 2008

Garlan, D. et al.: Increasing System Dependability Through Architecture-based Self-repair. In: Architecting Dependable Systems, pp. 61-89. Springer, Berlin/Heidelberg (2003)

Geihs, K. et al.: Modeling of Context-aware Self-adaptive Applications in Ubiquitous and Service-oriented Environments. In: Proc. of the Workshop on Software Engineering for Adaptive and Self-Managing Systems, (2009)

Hussmann, H., Meixner, G., Zuelke, D. (eds.): Model-Driven Development of Advanced User Interfaces. Springer, Berlin/Heidelberg (2011)

Jameson, A.: Adaptive interfaces and agents. In: The human-computer interaction handbook, Julie A. Jacko and Andrew Sears (Eds.). L. Erlbaum Associates Inc., Hillsdale, NJ, USA 305-330, (2002)

Kephart, J. O., Chess, D. M.: The Vision of Autonomic Computing. Computer Vol.36, No. 1, pp. 4150. (2003)

Kramer, J., Magee, J.: Self-managed Systems: An Architectural Challenge. In: Proc. of 2007 Future of Software Engineering (FOSE'07), IEEE Computer Society, Washington, DC, USA (2007)

Lavie, T., Meyer, J.: Benefits and costs of adaptive user interfaces. In: Int. J. Hum.-Comput. Stud. 68, 8 (August 2010)

Link, S. et al.: Modellgetriebene Entwicklung grafischer Benutzerschnittstellen (Model-Driven Development of Graphical User Interfaces). i-com Vol. 6, No. 3., Oldenbourg, Munich (2008)

Oppermann, R.: Individualisierte Systemnutzung. In GI - 19. Jahrestagung, I, Computergestützter Arbeitsplatz, Manfred Paul (Ed.). Springer-Verlag, London, UK, UK, 131-145, (1989)

Stephanidis, C. et al: Adaptable and Adaptive User Interfaces for Disabled Users in the AVANTI Project. In Proc. of the 5th Int. Conf. on Intelligence and Services in Networks: Technology for Ubiquitous Telecom Services (IS&N '98. Springer-Verlag, London, UK, UK, 153-166, (1998)