Model-Based Development of Adaptive UIs for Multi-channel Self-service Systems

Enes Yigitbas, Holger Fischer, Thomas Kern, Volker Paelke

Model-based Development of Adaptive UIs for Multi-Channel Self-Service Systems

Enes Yigitbas[1], Holger Fischer[1], Thomas Kern[2], and Volker Paelke[3]

[1] University of Paderborn, s-lab - Software Quality Lab,
Zukunftsmeile 1, 33102 Paderborn, Germany
eyigitbas@s-lab.upb.de, hfischer@s-lab.upb.de
[2] Wincor Nixdorf International GmbH,
Heinz-Nixdorf-Ring 1, 33106 Paderborn, Germany
thomas.kern@wincor-nixdorf.com
[3] Hochschule Ostwestfalen-Lippe, University of Applied Sciences,
Liebigstrasse 87, 32657 Lemgo, Germany
volker.paelke@hs-owl.de

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 is of great importance. Currently, an intuitive and flexible usage is often limited because of the monolithic system 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.

Keywords: MBUID, Self-Service Systems, user interface, model-based development, usability, adaptive user interfaces, user experience

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 (ATMs), 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.

Imagine a distributed, networked Multi-Channel Self-Service System where 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.

Developers of such systems are faced with the following challenges:

1. Manage high complexity: Sharing an application's user interface and client-side logic across multiple heterogeneous devices in order to support distributed business transactions requires changes to the system's architecture.
2. Increase efficiency of multi-platform software development across heterogeneous computing platforms (Windows, iOS, Android, Windows Phone etc.).
3. Implement software that adapts itself to differences in system functionality and user interface.
4. Integrate user centred design into the development process, extending the existing methods to cover the necessary adaptation options.

According to Petrasch [1], the effort of implementing an application's user interface constitutes at least 50% of the total implementation effort. Developing separate applications for each potential device and operating system is neither a practical, nor a cost effective solution. If we also consider multi-modal environments, such attempts tend to become nearly impossible.

Model-based User Interface Development (MBUID) [2] suggests a solution to these problems. In this paper, we will show a concept how MBUID can be combined with self-adaptive approaches to generate user interfaces for Multi-Channel Self-Service Systems.

2 Background and Related Work

Focusing on the topic of model-based user interface development of 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 which can be adapted or adapt themselves to the various skills and preferences of the users. Norcio and Stanley consider that the idea of an *adaptive UI* is straightforward since it simply means that: The interface should adapt to the user; rather than the user must adapt to the system [13]. A classification of different adaptation techniques was introduced by Oppermann [11] and refined by Brusilovsky [8]. UIs with adaptation capabilities were proposed in the context of various domains ([12], [9]). Besides works dealing with the quality of adaptive UIs ([14], [10]), there are also proposals for integrating adaptive UI capabilities into enterprise applications (e.g. [15]).

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 and to the development of a user interface [4].

The CAMELEON Reference Framework (CRF) [3] provides a unified framework for model-based and model-driven development of user interfaces. Several approaches have proposed a model-based development of user interfaces based on the CRF ([5],[6]). Also the aspect of UI adaptation was analyzed within the CAMELEON-project [17] or in the context of UsiXML [16].

However, the combination of both aspects: multi-channel interaction and adaptive UIs is a promising perspective which is not yet fully covered in the application domain of Self-Service Systems. Therefore in our approach for model-based user interface development we combine ideas from the field of MBUID, adaptive UIs and multi-channel interaction.

3 Adaptive UIs for Multi-Channel Self-Service Systems

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) or smartphone (S) to start the interaction with the Self-Service System. 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

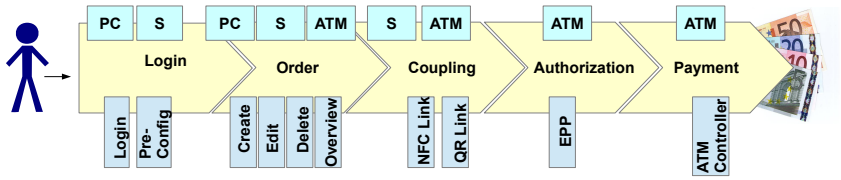

Fig. 1. Transaction workflow on a Multi-Channel Self-Service System

involved devices in the transaction process is established. This is coordinated by a connection manager which makes use of Near Field Communication (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.

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 2 shows the target architecture of our solution based on the CAMELEON 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 (M2M) from the Abstract User Interface Model (AUI) into the Concrete User Interface Model (CUI) 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. It is contemplated to supplement the Adap-

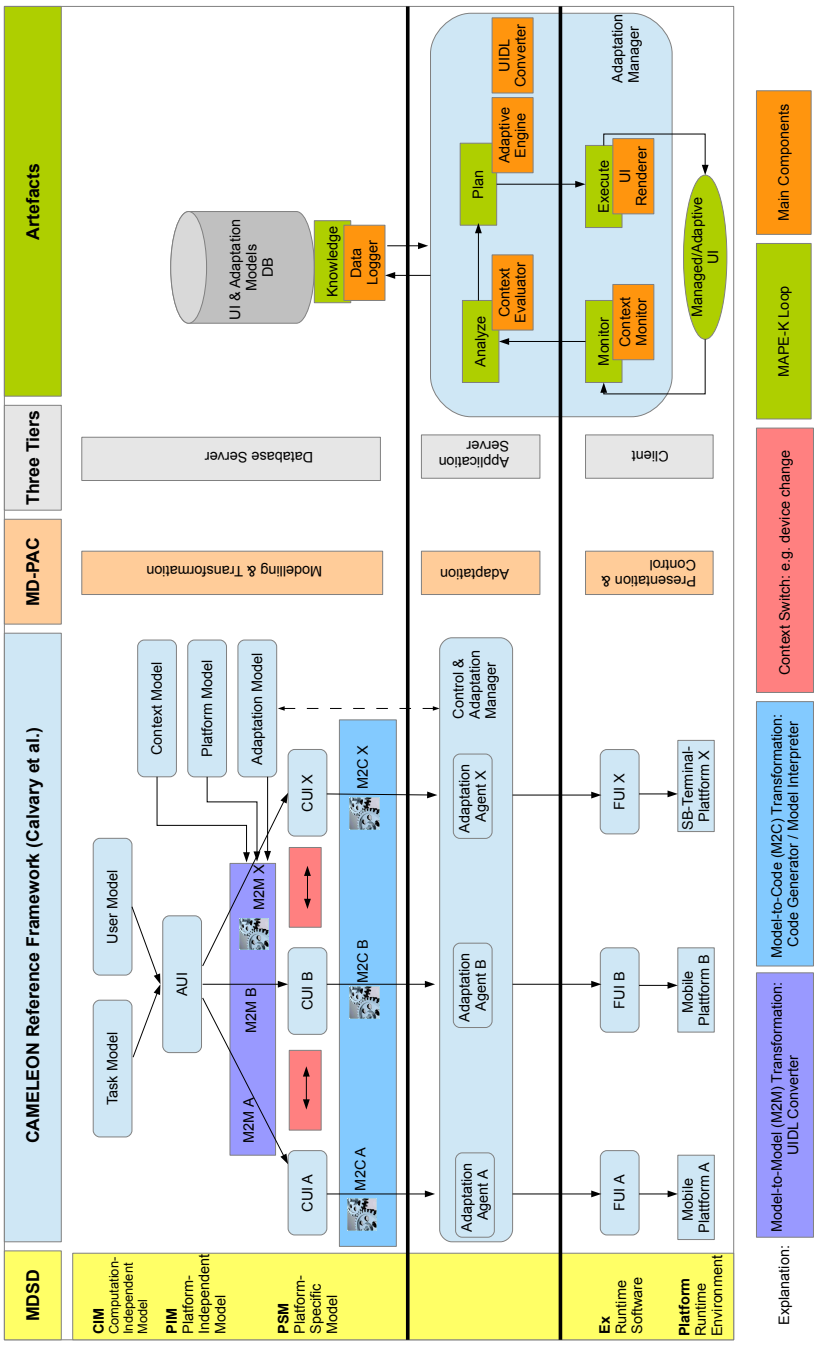

Fig. 2. Architecture: Model-based UI Development View

tation Manager by dedicated sub-components or Adaptation Agents. The latter interoperate on the respective platform instances and are responsible for the adaptation of the user interfaces and their coupled functionality.

For implementing the adaptation steps, our target architecture makes use of IBM's MAPE-K loop [7]. 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.

In order to exemplify the idea of adaptation in our target architecture, an example walkthrough is depicted in Figure 3.

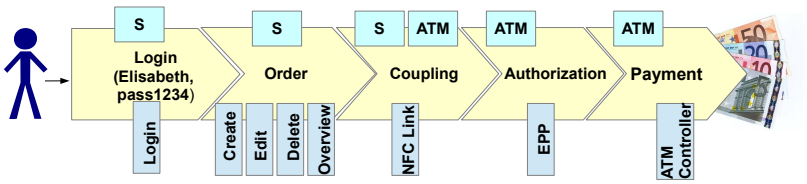

Fig. 3. Adaptation: Example Walkthrough

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. 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 for the current transaction. The transaction start time is analyzed to evaluate if the display brightness shall be increased, 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 evaluating 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 interface solutions that adapt themselves to the user, the environment and the context of use. Based on this operating concept, we established technological and methodological requirements and started to address them with the development of appropriate tools and techniques. Therefore we developed a model-based approach for the implementation of adaptive user interfaces that extends the CAMELEON reference framework. In this connection we described our target architecture, which pays special attention to aspects of adaptation and integrates them into the model-based development process for user interfaces to enrich UIs of Self-Service Systems with adaptivity capabilities. Our framework 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.

Acknowledgement

This paper is based on some of the work within “KoMoS”, a project of the “it’s OWL” Leading-Edge Cluster, partially funded by the German Federal Ministry of Education and Research (BMBF).

References

1. Petrasch, R.: Model Based User Interface Design: Model Driven Architecture und HCI Patterns. In: GI Softwaretechnik-Trends, Mitteilungen
2. Puerta, A.: A Model-Based Interface Development Environment. IEEE Software, 14, 4, S. 40-47 (1997)
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-target User Interfaces. In: Interacting with Computers, pp. 289-308 (2003)
4. Hussmann, H., Meixner, G., Zuelke, D.: Model-Driven Development of Advanced User Interfaces. Springer, Berlin/Heidelberg (2011)
5. Link, S., Schuster, T., Hoyer, P., Abeck, S.: Modellgetriebene Entwicklung grafischer Benutzerschnittstellen (Model-Driven Development of Graphical User Interfaces). i-com Vol. 6, No. 3., pp. 37-43. Oldenbourg, Munich (2008)

6. Botterweck, G.: A Model-driven Approach to the Engineering of Multiple User Interfaces. In: Khne, T. (ed.) Proceedings of the 2006 International Conference on Models in Software Engineering (MoDELS'06), pp. 106-115. Springer, Berlin/Heidelberg (2006)
7. Kephart, J. O., Chess, D. M.: The Vision of Autonomic Computing. Computer Vol. 36, No. 1,(2003)
8. Brusilovsky, P.: Adaptive Hypermedia. In: User Modeling and User-Adapted Interaction 11, 1-2 (March 2001)
9. 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)
10. Lavie, T., Meyer, J.: Benefits and costs of adaptive user interfaces. In: Int. J. Hum.-Comput. Stud. 68, 8 (August 2010)
11. Oppermann, R.: Individualisierte Systemnutzung. In GI - 19. Jahrestagung, I, Computergesttzter Arbeitsplatz, Manfred Paul (Ed.). Springer-Verlag, London, UK, UK, 131-145, (1989)
12. 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, Springer-Verlag,(1998)
13. Norcio, A.F., Stanley, J.: Adaptive Human-Computer Interfaces: A Literature Survey and Perspective. In: IEEE Transactions on Systems, Man, and Cybernetics, 19, pp.399408, (1989)
14. 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
15. 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)
16. Mezhoudi, N.: User interface adaptation based on user feedback and machine learning. In Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion (IUI '13 Companion). ACM, New York, NY, USA, 25-28.
17. Balme, L.: CAMELEON-RT: A Software Architecture Reference Model for Distributed, Migratable, and Plastic User Interfaces. Proc. of 2nd European Symposium on Ambient Intelligent EUSAI'2004, LNCS, Vol. 3295. Springer-Verlag, Heidelberg, 2004, pp. 291–302.