13 Consistency in use through Model based User Interface Development

Marcus Trapp University of Kaiserslautern Gottlieb-Daimler-Straße67653 Kaiserslautern, Germany marcus.trapp@informatik.uni-kl.de

Martin Schmettow,Fraunhofer IESE Fraunhofer-Platz 167663 Kaiserslautern, Germany martin.schmettow@iese.fraunhofer.de

In dynamic environments envisioned under the concept of Ambient Intelligence the consistency of user interfaces is of particular importance. To encounter this, the variability of the environment has to be transformed to a coherent user experience. In this paper we explain several dimension of consistency and present our ideas and recent results on achieving adaptive and consistent user interfaces by exploiting the technology of model driven user interface development.

INTRODUCTION

The common progress towards increasingly available, possibly mobile services exists obviously everywhere. There are a lot of devices that facilitate our tasks in business and in everyday life. Usually we have to cope with many different types of those devices, which may even realize complex services. For these reasons it is difficult to be able to use all of their specific capabilities because we have to learn how to use these devices efficiently. Moreover, users want to use these devices in many different environmental contexts. This includes the brightness and the noise level of the ambience. For example, if we have a dark and noise ambience, we do not want to use speech user interfaces or graphical user interfaces (GUI) on devices that do not have active backlight. Another important thing that constitutes the environmental context is the current state of the service and the whole system. A user interface for a service may look very different if a system is in an emergency state instead of a normal state. Using devices efficiently gets even more difficult if several users want to use it at the same time, particularly if they want to use different services [10]. Thus, the diversity of usage situations, as a combination of users, services, contexts, and devices, is getting more and more complex through the enormous number of possible values of every single factor (see Table 1).

Since especially ambient computing environments are characterized by openness, heterogeneity, and dynamics, it is important to take into account the fact of not knowing all kind of possible usage situations at development time of the system, and thus, the user interface.

As today's devices are heterogeneous particularly pertaining to their usability and integration features nowadays sophisticated users with specific skills are required. In order to let common users benefit from these devices, an intuitive way of usage and integration is needed to shorten the initial phase of skill adaptation. Therefore, consistency in use as a key factor of usability is the most important factor in user interface design for these new ambient applications.

DIMENSIONS OF CONSISTENCY

Several dimensions of consistency in user interfaces can be identified which have different importance in specific usage situations.

The first consistency dimension addresses the Look & Feel of an application's user interface. This refers, for instance, to wording, used colors, shapes, layout, typefaces, etc., and the behavior of dynamic elements such as buttons, boxes, and menus in GUIs.

The Look & Feel is a characteristic of a specific interaction device. We characterize an interaction device as a combination of computer hardware, an operating system, and a user interface toolkit. Therefore, this consistency dimension is most import if many different tasks can be performed on a single device (rows 2 and 6 in Table 1). Independent from the kind of task a user wants to perform, a consistent and thus, familiar Look & Feel should be provided. User interface guidelines like [2] and user interface development tools like GUI builders [4] try to support user interface designers in achieving this consistency dimension at development time.

Nr.	# User (in context)	# Interaction-Device	# Task
1	1	1	1
2	1	1	many
3	1	many	1
4	1	many	many
5	many	1	1
6	many	1	many
7	many	many	1
8	many	many	many

Table 1. Usage Situation Types.

If one specific task can be performed using many different devices (rows 3 & 7 in Table 1), the Look & Feel of the task's user interface should be as consistent as possible with respect to the device characteristics. For example, if a button is labeled "Submit order" in a GUI, the command that initiates the submission of an order should also be "Submit order" in a speech user interface.

As mentioned above, ambient systems are characterized by openness and dynamics. Thus, it is very likely that it is unknown at development time what specific device will be used to perform a specific task a runtime. Under these circumstances a user interface designer has to face the huge problem of developing a user interface that uses the appropriate Look & Feel for all devices that may be used to perform the task.

The second consistency dimension deals with the performance of the tasks itself, independent from the kind of interaction device a user uses to perform a specific task. The temporal order of the steps needed to perform the task should be in principle as consistent as possible with respect to the device characteristics (rows 3 and 7 in Table 1). For example, before you can playback a video you first have to select a video file. This temporal order of steps should be the same in a GUI and in a speech user interface.

The third consistency dimension deals with the spatial order of available tasks at the same time and thus, the navigation to and selection of a specific task. This is especially important for ambient systems as the number of available tasks at the same time is very large as a result of their strong networking capabilities (row 2,4,6, and 8 in Table 1).

Today's user interface development approaches typically focus on the temporal ordering of tasks, leaving as unaddressed question of how we manage the spatial arrangement of tasks around us. "Thereby, we are spatially located creatures: we must always be facing some direction, have only certain objects in view, be within reach of certain others. How we manage the spatial arrangement of items around us is not an afterthought: it is an integral part of the way we think, plan, and behave" [8]. We use the available space to structure actions and to facilitate the selection of interactions and information.

We propose to structure the available tasks in three categories: personal tasks, role tasks, and room tasks.

- Personal tasks are associated with a specific user. These are tasks that s/he wants to be able to perform anytime independently from her/his current role or her/his current location (for example, information about the current time).
- Role tasks are associated with the current role of a user. For example, the tasks useful for a company's facility manager are different from the tasks useful for the company's CEO.

- Room tasks are associated with a specific location. For example, the tasks that can be reasonably performed in a kitchen are different from the tasks that can be reasonably performed in a home cinema.

Therefore, we propose to order all tasks that are available at the same time spatially from personal tasks over role tasks to room tasks. To ensure consistency the spatial order of available task for all user interfaces should always follow this order. Thus, a user has always the same natural navigation to available tasks.

PREVIOUS WORK

We gained a lot of experience in developing user interfaces for ambient systems during the development of an ambient system that assists cycling groups during their training [7]. The cyclists can interact with the system in several ways using different graphical user interfaces (using several bike computers with small to large displays), or a speech user interface (using headsets). Additionally, they could change the interaction device during the training dynamically [3]. We also developed an adaptive speech user interface that provides dynamic self adaptation according to the physical stress condition of the cyclist [9]. Thus, the cyclist's voice can be recognized even if s/he is nearly breathless.

We realized that it is an essential prerequisite for building usable interfaces for ambient systems to change from object orientation (devices) to task orientation. Additionally, as consequence of the complex diversity of possible usage situations it is obvious that it is unscalable to implement a user interface for each usage situation by hand. Thus, an automated solution like multiplatform generation is necessary.

In order to cover these facts we are using a model based user interface generation approach. We base our work mainly on the UsiXML approach, a modern user interface specification transformation language [11].

Our vision is to bring the user interface construction completely to run time. In a first step we set up a tool chain for generating XHTML user interfaces completely from models on a “requirements” level. In order to ensure the ergonomic quality (usability) of generated user interfacess, we enhanced the UsiXML default transformation path [11]

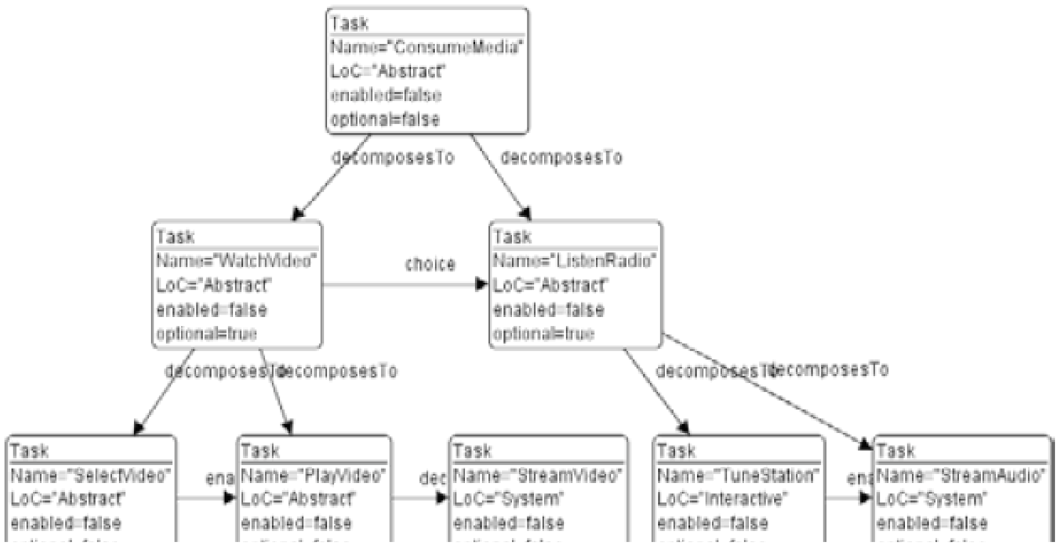

Figure 1 Home entertainment domain task model (simplified).

with formalized usability patterns [1]. One of our findings is that adding ergonomic rules to the transformation path not only enhances the usability of the generated user interfaces, but also results in truly adaptive user interfaces, which optimize themselves to the environmental context and user characteristics.

Another related work yet started is to bring the changing functionality of dynamic service environments into a single and consistent user interface. This can be achieved by enhancing the transformation path upfront by computing a situational valid task model based on the current and local availability of services. This approach will be outlined in the following.

DYNAMIC TASK MODELS

The basic idea in our approach is that task models, which are originally meant for describing the user-system interaction, will also serve for describing the functionality a single service provides to a service ensemble.

When a new device is installed in or becomes otherwise available to the environment, it is registered by an infrastructural service and announces its functionality by a task model fragment (we call this the device functionality model (DFM)). These tasks are then added to the current task model, which we call room task model (RTM), and will be made available to the user by the user interface generation engine outlined above. We call this the additive approach, because it already brings dynamic functionality to the user interface generation, but in a mere additional sense.

As outlined above, we expect dynamic service environments to exploit synergies between services, which basically means that adding two services A and B could result in a (maybe not earlier thought of) service C. For today we have to assume, that there exists a comprehensive model, which covers all available services (here: A and B) and their compound services (here: C). This seems to be realistic when thought of as a domain specific but vendor independent standard. Indeed, something similar already exists for the domain of home entertainment, namely the UPnP AV standard [12].

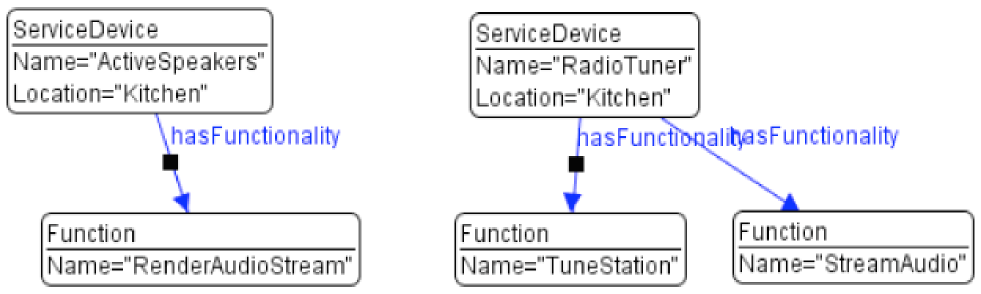

Figure 2 Device functionality models for active speakers and radio tuner.

The domain model in our approach also has the form of a task model, accordingly it is called domain task model (DTM). The RTM is then computed by deleting all branches out of the DTM, which can not be fully instantiated by DFM fragments. In this subtractive approach of dynamic task models the DTM can also describe compound services that can be activated, when all components are instantiated by services.

After this procedure a single task model with all available services is fed to the user interface generation engine. Since every task is transformed by the same transformation rules (enhanced by ergonomic rules) user interfaces can be expected that are consistent in Look & Feel, temporal order, and spatial order.

In the following example, we show that the computation of the RTM is quite straight forward using the standard means of model based user interface development with UsiXML.

EXAMPLE

In our example we imagine a home entertainment environment, which consists of several services, which are localized and which announce their functionality with DFMs:

- different multimedia sources
- output devices

The whole environment is controlled by a personalized control device (CD), where the generated user interface is rendered (e.g. a PDA). The user interface is generated by the model-driven UI generation engine outlined above. But here the engine is fed with a dynamically generated RTM.

More concrete the environment consists of the following service devices (SD): a video player, a high definition wall mounted screen and active speakers in the living room and active speakers and a radio tuner in the kitchen. The DTM states the two main tasks (see Figure 1): (1) "Watch video" or (2) "Listen radio". Both tasks require a source device (e.g. tuner or video recorder) and an audio renderer (active speakers) to be available. This is stated in the DTM by system tasks. Given that the above mentioned devices announce their functionality as DFMs (see Figure 2), the room task model can be computed for both rooms.

We implemented the transformation rules using attributed graph grammar (AGG) [5], the preferred transformation language for UsiXML models. Our transformation follows the following five steps from DTM and DFM to RTM:

- InstantiateTasks: Tasks where a matching SD functionality exists in the same location like the CD are mapped to this functionality and enabled
- EnableAnchTasks: Tasks with all non optional subtasks enabled are set to enabled
- DisableDeadBranches: Enabled tasks with disabled non optional ancestor task are set to disabled Instantiations from SDs to disabled tasks are removed
- DeleteDeadBranches: All disabled tasks are removed

As a result the main task “Watch video” is activated in the living room and “Listen radio” in the kitchen. We also exploited the support of compound services in the subtractive approach since the task “Watch video” is instantiated by two different devices for video and audio rendering.

These localized task models can then be fed to the user interface generation engine, which will then send the generated user interface to the users CD (e.g. a PDA). In this scenario at least two vendor specific CDs (e.g. remote controls) are substituted by one integrated.

CONCLUSION

After explaining several dimension of consistency and their importance for ambient systems, we have argued that the model based user interface development can be exploited for such dynamic service environments, where the consistent integration of different services becomes very critical for the user experience. This is an interesting fact, because the model driven approaches are often motivated by the idea of adaptivity, for instance, changing the appearance of the user interface dependant on characteristics of user and context. There even exists some findings that adaptivity of user interfaces is sometimes harming consistency by resulting in too variable behavior [6].

We believe that with dynamic user interfaces the use of elaborated ergonomic rules will result in adaptive interfaces that are pleasant for the user and not confusing. We also demonstrated that harmful variability can be made consistent by means of model based user interface development.

ACKNOWLEDGMENTS

This work was supported in parts by the BelAmI project of Fraunhofer IESE - which in turn is funded by the German Federal Ministry of Education and Research (BMBF), the State of Rhineland-Palatinate's Ministry of Science (MWFFK), and the Fraunhofer Gesellschaft e.V.

REFERENCES

[1] Apfelbacher, F. Formalisierung von ergonomischen Regeln zur Verbesserung der Benutzerfreundlichkeit von adaptiven Benutzungsschnittstellen. Diploma Thesis TU Kaiserslautern, 2006.

[2] Apple Human Interface Guidelines http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/index.html

[3] Bartelt, C., Fischer, T., Niebuhr, D., Rausch, A., Seidl, F., and Trapp, M. Dynamic Integration of Heterogeneous Mobile Devices. Proceedings of the Workshop in Design and Evolution of Autonomic Application Software (DEAS), 2005.

[4] Eclipse Visual Editor Project http://www.eclipse.org/vep/WebContent/main.php

[5] Ermel, C. and Schultzke, T. The Agg Environment: A Short Manual. TU Berlin, http://tfs.cs.tu-berlin.de/agg/ShortManual.ps

[6] Findlater, L. and McGrenere, J. A Comparison of Static, Adaptive, and Adaptable Menus. CHI, 2004.

[7] Jaitner, T., Trapp, M., Niebuhr, D., and Koch, J. Indoor-Simulation of Team Training in Cycling. International Sports Engineering Association (ISEA) Conference, 2006.

[8] Kirsh, D. The intelligent use of space. Journal of Artificial Intelligence, 73(1-2), 31-68, 1995.

[9] Klus, H. Conception of Intelligent Applications. Diploma Thesis TU Kaiserslautern, 2005.

[10] Kray, C., Wasinger, R., and Kortuem, G. Concepts and issues in interfaces for multiple users and multiple devices. Workshop on Multi-User and Ubiquitous User Interfaces (M3UI '04), 2004.

[11] Limbourg, Q. Multi-Path Development of User Interfaces. PhD Thesis Université catholique de Louvain, 2004.

[12] UPnP-Forum, MediaServer V 1.0 and MediaRenderer V 1.0. http://www.upnp.org/standardizeddcps/mediaserver.asp