The end-user vs. adaptive user interfaces

Veit Schwartze, Frank Trollmann, Sahin Albayrak
DAI – Labor
Ernst Reuter Platz 7
Berlin, 10781Germany
+49 30/314 - 74064, 74048, 74001
{Veit.Schwartze, Frank.Trollmann, Sahin.Albayrak**}**@dai-labor.de

ABSTRACT
In smart environments, applications can support users in their daily life by being ubiquitously available through various interaction devices. Applications deployed in such an environment, have to be able to adapt to different context of use scenarios in order to remain usable for the user. For this purpose the designer of such an application defines adaptations from her point of view.

Because of situations, which are unforeseeable at design time, the user sometimes needs to adjust the designers' decisions. For instance, the capabilities and personal preferences of the user cannot be completely foreseen by the designer. The user needs a way to understand and change adaptations defined by the designer and to define new adaptations. This requires the definition of a set of context of uses and adaptations applied to the user interface in this situation. For this reason supportive user interfaces should enable the user to control and evaluate the state of the adaptive application and to understand "What happens and why?"[1] In this paper, we describe the requirements and function of a supportive user interface to evaluate and control an adaptive application, deployed in a smart environment.

Keywords
Context aware applications, end-user support, adaptation- and situation definition

INTRODUCTION
Applications, which are deployed into smart environments, often aim to support the users in their every-day life. Such applications must be able to adapt to different context of use scenarios to remain useable in every situation. The large set of possible properties of devices leads to an infinite number of possible situations which cannot be considered at design time completely.

For instance there is a large set of heterogenic displays for graphical user interfaces, which differ in their aspect ratio, resolution and input possibilities. In addition, each user has different abilities or disabilities as well as a personal taste. Such preferences cannot be predicted or categorized in a reliable way at design time. The ability of the user to distribute user interface elements to different devices also raises the problem of multi-application scenarios.

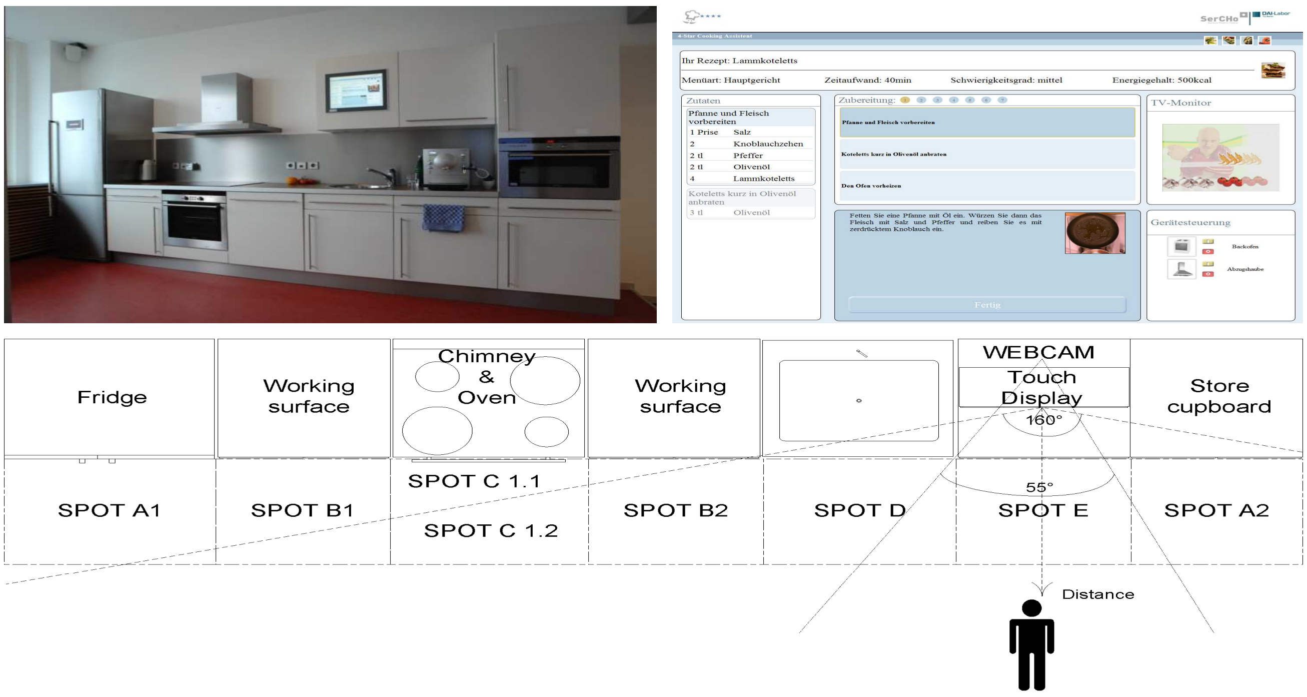
This raises the need for the user to understand and control adaptations of the application at runtime in order to personalize it to her liking. Following, we want to describe the requirements and functions of a supportive user interface, to enable the user to evaluate and control user interface adaptations.

The next section describes the problem in more detail by an example application. This is followed by the requirements that have to be achieved by a supportive user interface. The section work in progress then gives an overview about the layout- and adaptation model, which are needed to generate the position, size and style for each user interface element and to change these layout dimensions to a specific situation. The conclusion summarizes the paper and describes the next steps.

PROPLEM DESCRIPTION
In this section we illustrate the problem space by an example of a cooking assistant. Afterwards we derive problems that have to be solved within the scope of adaptive user interfaces.

The cooking assistant is an application that enables the user to search for recipes and supports her while cooking them. During the cooking process the cooking assistant is able to control the devices in the kitchen. We deployed the cooking assistant into a real kitchen environment like depicted in Figure 1 top-left. The main screen, shown in Figure 1, top-right, guides the user through the cooking steps and provides help if needed. The bottom half of Figure 1 illustrates several spots corresponding to the different working positions and user tasks in the kitchen.

[1] Direct manipulation vs. interface agents, Shneiderman, B. & Maes, P. Interactions, ACM, 1997, 4, 42-61


4-Star Cooking Assistent

Ihr Rezept: Lammkoteletts

Menüart: Hauptgericht Zeitaufwand: 40min Schwierigkeitsgrad: mittel Energiegehalt: 500kcal

Zutaten

Pfanne und Fleisch vorbereiten

1 Prise	Salz
2	Knoblauchzehen
2 tl	Pfeffer
2 tl	Olivenöl
4	Lammkoteletts

Koteletts kurz in Olivenöl anbraten

3 tl	Olivenöl

Zubereitung: 1 2 3 4 5 6 7

Pfanne und Fleisch vorbereiten

Koteletts kurz in Olivenöl anbraten

Den Ofen vorheizen

Fetten Sie eine Pfanne mit Öl ein. Würzen Sie dann das Fleisch mit Salz und Pfeffer und reiben Sie es mit zerdrücktem Knoblauch ein.

Figure 1: The kitchen with the cooking assistant running on a touch screen (top-left), the main screen of the cooking assistant (top-right), and the location spots defined by the context model (bottom).

In [4], we define different automatic adaptations, to adapt the user interface to specific situations, defined by working steps, to support the user while operating in the kitchen. Two examples are:

- Distance-based adaptation: While cleaning dishes the user wants to learn more about the next step. A video helps to understand what has to be done. Depending on the users distance to the screen, the layout algorithm increases the size of video element to improve the legibility. In this case the distance of the user to the interaction device is used to calculate the enlargement factor for this element.
- Spot-based adaptation: While using the cooking assistant, the user is preparing ingredients, following the cooking advices and controlling the kitchen appliances on a working surface. Because it is difficult to look at the screen from this position, shown in Figure 1 bottom, the important information (Step description and the list of required ingredients) are highlighted.

The described adaptions can improve the interaction with the application but the user is not able to influence the adaptations or to interfere, which can lead to frustration and the denial of the application. For instance, if the user is concentrated on the ingredients list or the textual step description and the size of these elements is scaled down. This problem space can be divided into the evaluation and control of the system state and behavior.

Incomprehensible adaptations can lead to confusions for the user. The user has little knowledge about the state of the system and its internal representation of the environment, user and platform characteristics. Therefore, it is hard for her to comprehend why a specific adaptations has been applied. It is not only important to know why something happens but rather how to influence the behavior of the user interface generation. At design time unknown environment conditions and user characteristics leads to the wish to adjust adaptations at runtime e.g. button size to the preference, capabilities or rule of the actual user. For example a user with a color blindness or degeneration of the macula[2] may wish to adjust the contrast and the font size to improve the visibility and readability of the user interface. In a similar case, left-handed users may wish to adjust the position of interaction elements (e.g. buttons) so their hands don't hide important information during interaction.

Additionally, supportive user interfaces can allow the user to define individual distributions, which leads to free space or multi-application scenarios. These problems must be solved. The next section defines the requirements of an approach to enable the user to adjust, interfere or define new adaptations.

[2] That means the loss of vision in the center of the visual field (the macula) because of damage to the retina.

REQUIREMENTS

The requirements of a supportive application are derived from the need to evaluate the state of the system and to control the behavior of the adaptation algorithm. They are divided into:

- An approach, to define the layout of an application and the adaptations to different context of use scenarios and
- The support of the end-user to change these adaptations to their preferences.

As aforementioned, heterogeneous interaction devices, sensors and appliances makes the development of user interfaces for smart environments a challenging and time-consuming task. To reduce the complexity of the problem user interface developers can utilize models and modeling languages. User interfaces generated from models at design time often fail to provide the required flexibility because decisions made at design time are no longer available at runtime. To handle this issue, the use of user interface models at runtime has been suggested [6].

The approach shifts the focus from design to run time and raises the need to support the end-user by the development and personalization of applications. A meta-user interface offers an abstract view to the state of the system and provides an interface to influence its behavior. In [1] the system provides access to the task and the platform model, at which the platform model shows the interaction devices currently available in the home. Like the described approach, the supportive user interface should visualize the user, environment, and platform information of the running system in a simple way. Also the situations and corresponding adaptations (system and user initiated) should be transparent to the user. This means, the adaptation rules representation must describe in detail why and how the user interface changes and enable the user to interfere. To make the execution of user interface adaptations more comprehensible for the user, feedback should be provided like the animation of user interface changes.

Additionally, the user needs a way to delete or adjust layout adaptations rules and thus change the situation precondition and the adaptation. A preview of the changes avoids wrong decisions. The definition of new adaptation rules requires the selection of context variables, their accuracy and range of values which accurately describe the situation. Following, the user defines the executed adaption. First she has to select the layout dimension (size, orientation, containment) she wishes to influence, following she selects a specific statement and the changes realized by the layout generation algorithm. Furthermore, some statements need parameters e.g. a statement, defines the size of a button, which depends on the width of the finger.

The state of the realization is described in the next section.

WORK IN PROGRESS

In our implementation the components that realize adaptations of user interfaces, which can be adjusted at runtime, are the layout and the adaptation model, both based on a model@runtime [6] approach to use the same model at design and run time.

Additionally, we have done the first steps to expand the approach of a meta-user interface described in [3] to provide a simple way to adapt the layout generation algorithm to the needs of the user.

Layout model

The layout model defines the structure of the user interface and spatial relationships between user interface elements. It consists of the user interface structure and a set of statements. The user interface structure is determined by a tree-like hierarchy of Containers and UI-Elements. Containers can contain a set of nested containers and nested elements. User interface elements are the visible parts of the user interface structure and can present information to the user. The statements describe the size, style and spatial relationships between the user interface elements.

The approach differs from previous approaches in two general aspects. First of all, we interpret the design models, such as the task tree, the dialog model, the abstract user interface model and the concrete user interface model. We derive the initial structure of the user interface and suggest statements influencing the spatial relationships and size of user interface elements from this information. Therefore we propose an interactive, tool-supported process that reduces the amount of information that needs to be specified for the layout. The tool enables designers to comfortably define design model interpretations by specifying statements and subsequently applying them to all screens of the user interface. The layout model editor is described in [7] in more detail.

Furthermore, different to other layout generation approaches like [2], we create a constraint system at runtime. A sub tree of the user interface structure marks the user interface elements that are currently part of the application's visible user interface and a set of statements regarding these nodes is evaluated and creates a constraint system solved by a Cassowary constraint solver. The result of a successful layout calculation is a set of elements, each consisting of the location (an absolute x, y coordinate) and a width and height value.

Adaptation model

The adaptation model describes possible situations and the corresponding adaptations of the layout model of the application. For this purpose, the adaptation model consists of adaptation definitions. Each adaptation definition consists of a tuple of a situation, describing when the rule should be applied and an adaptation rule, describing how the layout model is adapted. The adaptation rules may cause changes to the user interface structure and may also add, modify or delete statements.

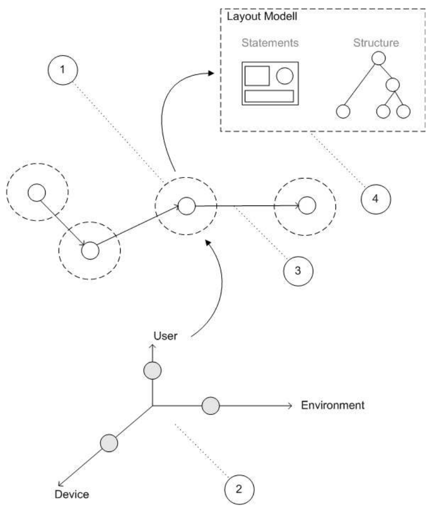


Figure 2: Example graph of layout model adaptations

In the center of Figure 2 an example of an adaptation graph is shown. Each node (①) defines a state of the layout model (④) and each edge (③) a set of adaptation rules to transform the layout model to a state, applicable for a specific situation (②). A situation is determined by a certain state of the user, device and environment.

Additionally, we have done first steps to define a supportive user interface.

Supportive user interface

The supportive user interface should provide a way, to understand the context information representation within the system and allow the manipulation of the user interface generation and adaptation algorithm.

To match the requirements defined above, a supportive user interface should hide the complexity of the interaction space (various sensors gathering information about the environment, heterogenic interaction devices and user characteristics) from the user. Also the complexity of situation definition and recognition must be encapsulated. Accordingly, the situation description, the adaptation definition must be as simple as possible but as complex as necessary. The user must be able to define powerful adaptations but shouldn't be overstrained. A way to do this is to derive semantic information from the user interface models to visualize the effected elements on the screen. To preview the user interface changes, the supportive user interface application simulates the layout model changes and visualizes the result of the calculation to the user.

In [5] we use the information derived from the concrete user interface model (e.g. all button elements) and allow the user to define a statement which influences the size of these elements. A screenshot is shown in Figure 3.

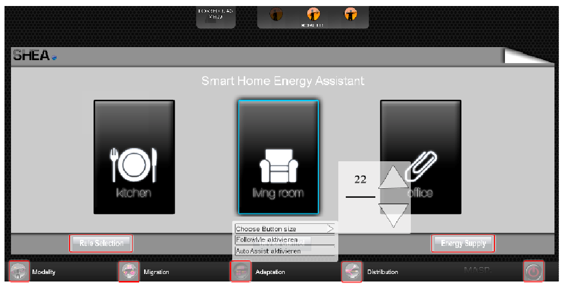


Figure 3: Supportive user interface screenshot

The supportive user interface application adds a statement to the layout model and triggers the recalculation mechanism to update the user interface of the application.

CONCLUSION

In this paper, we have defined the requirements of a SUI to control and evaluate the state of the adaptive application and have shown first steps of implementation.

In the future, we plan to increase the ratio of automatic statements derived from the user interface models for the layout generation process. Additionally, we take the domain model objects influenced by the user interface elements into account. The resulting set of statements reduces the amount of designer defined statements. At run time, the situation recognition and the adaptation algorithm must be evaluated, especially the handling of imperfect (e.g. inaccuracy, incompleteness, conflicting) context information and the user interface adaptation over the time.

Last but not least, we have to implement the SUI concepts and prove the acceptance of our approach by user studies. Additionally, because the user doesn't want to define all adaptions manually, we want to explore the possibilities of machine learning algorithms to reduce and simplify the definition of adaptations.

REFERENCES

1. Joelle Coutaz. Meta-user interfaces for ambient spaces: Can model-driven engineering help? In Margaret H. Burnett, Gregor Engels, Brad A. Myers and Gregg Rothermel, editors, End-User Software Engineering, number 07081 in Dagstuhl Seminar Proceedings. Internationales Begegnungs und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007.
2. Christof Lutteroth, Robert Strandh, and Gerald Weber. Domain specific high-level constraints for user interface layout. Constraints, 13(3):307 - 342, 2008.
3. Dirk Roscher, Marco Blumendorf, and Sahin Albayrak. Using Meta user interfaces to control multimodal interaction in smart environments. In Gerrit Meixner; Daniel Görlich; K. Breiner; H. Huÿmann; A. Pleuÿ; S.

Sauer; J. Van den Bergh, editor, Proceedings of the IUI'09 Workshop on Model Driven Development of Advanced User Interfaces, volume 439 of CEUR Workshop Proceedings, ISSN 1613-0073. CEUR Workshop Proceedings (Online), 2009.

4. Veit Schwartze, Sebastian Feuerstack, and Sahin Albayrak. Behavior sensitive user interfaces for smart environments. In HCII 2009 - User Modeling, 2009.

5. Veit Schwartze, Marco Blumendorf and Sahin Albayrak. Adjustable context adaptations for user interfaces at runtime. In Proceedings of the Working Conference on Advanced Visual Interfaces, pages 321 - 325, 2010.

6. Gordon Blair, Nelly Bencomo, and Robert B. France. Models@ run.time. Computer, 42(10):22$^{\perp}$ 27, Oct. 2009.

7. Sebastian Feuerstack, Marco Blumendorf, Veit Schwartze, and Sahin Albayrak. Model-based layout generation. In Paolo Bottoni and Stefano Levialdi, editors, Proceedings of the working conference on Advanced visual interfaces. ACM, 2008.