"Enhancing interaction with supplementary Supportive User Interfaces: Meta-UIs, Mega-UIs, Extra-UIs, Supra-UIs ..."

Proceedings of the 1st International Workshop on

Supportive User Interfaces : SUI 2011

at the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems

Pisa, Italy - June 13, 2011

Edited by:

Grzegorz Lehmann,
DAI-Labor, TU-Berlin.
Alexandre Demeure,
University of Grenoble, INRIA, LIG
Mathieu Petit, Gaëlle Calvary
University of Grenoble, CNRS, LIG

http://www.supportiveui.org

Table of Contents

~

Enhancing interaction with supplementary Supportive User Interfaces (UIs): Meta-UIs, Mega-UIs, Extra-UIs, Supra-UIs …

Alexandre Demeure
University of Grenoble, LIG
INRIA, 655 av. de l'Europe, 38334
St Ismier Cedex, France
First.Last@inrialpes.fr

Grzegorz Lehmann
DAI-Labor, TU-Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
First.Last@dai-labor.de

Mathieu Petit, Gaëlle Calvary
University of Grenoble, CNRS, LIG
385, av. de la bibliothèque
38400 St Martin d'Hères, France
First.Last@imag.fr

ABSTRACT

In order to improve the interaction control and intelligibility, end-user applications are supplemented with Supportive User Interfaces (SUI), like meta-UIs, mega-UIs, helping or configuration wizards. These additional UIs support the users by providing them with information about the available functionalities, the context of use, or the performed adaptations. Such UIs allow the user to supervise and modify an application interactive behavior according to her/his needs.

Given the rising complexity of interactive systems, supportive UIs are highly desirable features. However, there is currently no common understanding of types and roles of supportive UIs. Enabling concepts and definitions underlying the engineering of such UIs are also missing. In order to fill this gap, the workshop seeks a discussion with a broad audience of researchers, who have experience with the design and development of supportive UIs.

Author Keywords
Supportive User Interfaces, UIs quality, explanative UIs, help systems, awareness of the context of use, meta-UI, mega-UI, supra UI.

ACM Classification Keywords
H.5.2 [User Interfaces]: Ergonomics, Graphical user interfaces (GUI), Prototyping, User-centered design, Evaluation/methodology. D.2.2 [Software Engineering]: Design Tools and Techniques, User Interfaces.

General Terms
Design, Human factors, Algorithms.

INTRODUCTION

Enabling technologies make it possible to create more and more complex systems in terms of functional core, new interaction techniques and context-of-use dynamics. Coming along with systems complexity, the users require a better understanding and control of their applications.

In the aftermath of "pervasive intelligibility" researches [5], this workshop focuses on human-computer interaction and more specifically on the engineering of user interfaces to foster intelligibility and control. User interface intelligibility has been approached from different perspectives. The concept of "Meta-UI" has been introduced as a metaphorical UI to control and evaluate the state of interactive ambient spaces [1]. Other works focus on self-explanatory user interfaces, and make it possible for the end-user to understand the design of the user interface [4]. The Crystal tabletop prototype has been developed to handle a complex platform composed of components like TVs, robots, picture frames, etc. [3]. Crystal provides the users with intelligible UIs to control the media distribution and the component discovery.

Such research projects exemplify the notion of **supportive UI**. In a broader context this workshop aims to identify and classify the supportive UIs that may enhance the interaction (e.g., by rendering the workflow in e-government applications or making it possible to the end-user to see the available platforms in the surrounding and redistribute the UIs him/herself). These include Meta-UIs [1], Mega-UIs [2], self-explanatory UI, Supra-UIs and others. The goals of the workshop are to:

- Define the concept of supportive UI,
- Elicit the dimensions of supportive UIs through a taxonomy that would cover both the abstraction and presentation of supportive UIs,
- Discuss the properties supportive UIs should convey,
- Explore how to integrate supportive UIs into development processes and Model-based UI development,
- Identify the key research stakeholders for further research.

To that end, examples of points of discussion could be:

- What is the added-value for the users? Which one is the border between UI and supportive UI? Do UIs for help, personalization or end-user programming belong to supportive UIs?

- Are supportive UIs parts of the original UI? Are they generic or do they require application-specific features or rendering?
- How to take benefit from model-based approaches to integrate supportive UIs by design?
- How to evaluate supportive UIs?

The relevance of the workshop is two-fold: first, to improve the quality of UIs, and to reconcile research areas (e.g., model-based approaches, end-user programming).

ORGANIZATION

Alexandre Demeure is assistant professor at the University of Grenoble. His main research interests include plasticity of UIs, software architecture for HCI, multitouch interaction and creativity support. **Grzegorz Lehmann** is a PhD student at the Technische Universität Berlin. His research focuses on the utilization of runtime and executable models for developing ubiquitous UIs**. Mathieu Petit** is a post doctoral fellow at the University of Grenoble. His current research focuses on model description and automated transformation to design plastic UIs. **Gaëlle Calvary** is professor at the University of Grenoble. Her research area is about UI plasticity to ensure UI quality along the variations of the context of use. She mostly explores model-driven engineering.

FORMAT

We propose a one-day workshop with six working hours, excluding the breaks. Our goal is to facilitate a combination of presentations, demonstrations, discussions and community building.

Candidate participants must submit a short paper or a position statement. The short paper describes experiences, ongoing work or results related to the workshop's topic. We encourage submissions including video demonstrations. A position statement describes requirements or issues the participant encounters when designing and/or implementing supportive UIs, as well as desirable solutions from the author's point of view.

In order to focus the discussion on supportive UIs concepts and design, the organizers will select the most prominent themes relative to the workshop topic from the set of accepted papers. The authors will be asked to mainly focus their presentations on these relevant themes.

At first, the participants will introduce themselves. Each introduction should include a short statement about the favorite problem to tackle during the workshop. After the introductions, Jérémie Melchior, from Université Catholique de Louvain (Belgium) will give an introduction speech about quality properties for intelligent UIs. The workshop will then focus on reviews and discussions of topics emerged from the position papers. The selected papers will be presented in two one-hour slots.

After the lunch break, participants will be split into groups structured around the core topics provided in the papers and statements. Afterwards, the groups will report back to the plenary forum. The following is a tentative schedule for the workshop, time given in working hours, excluding breaks:

0:00-0:15	Introduction by the organizers
0:15-45	Brief introduction talk by each participant, using predefined template (e.g., background, experience, favorite problem)
0:45-1:15	Invited talk : "Quality properties of intelligent interfaces" by Jérémie Melchior
1:15-2:15	Selected paper presentations
2:15	Break
2:15-3:15	Selected paper presentations
3:15-3:30	Summary and presentation of afternoon works
3:30	Lunch
3:30-5:00	Breakout groups – initiation
5:00-6:30	Plenary discussion on group results, future agenda and follow-up activities

PROGRAM COMMITTEE

- Jean Vanderdonckt
- Gerrit Meixner
- Joëlle Coutaz
- Kris Luyten
- Peter Forbrig
- Marco Blumendorf
- Melanie Hartmann
- Natalie Aquino
- Oscar Pastor
- Victor Lopez
- Dominique Scapin
- Philippe Palanque
- Marco Winckler
- Audrey Serna
- Dirk Roscher

WEBSITE AND CONTACT

http://www.supportiveui.org/ ; chairs@supportiveui.org

REFERENCES

[1] Coutaz, J. **Meta-User Interfaces for Ambient Spaces**. In Proc. of the 5th Int. Ws. on Task Models and Diagrams for Users Interface Design: TAMODIA 2006, pp 1-15, Coninx, K., Luyten, K. and Schneider, K. A. (eds.), Springer LNCS 4385. Hasselt, Belgium, October 23-24, 2006.

[2] Sottet, J-S., Calvary, G., Favre, J-M. and Coutaz, J. **Megamodeling and Metamodel-Driven Engineering for Plastic User Interfaces: MEGA-UI**. In Human-Centered Software Engineering, pp 173-200, Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (eds.), Springer Human-Computer Interaction Series. 2009.

[3] Seifried, T., Haller, M., Scott, S. D., Perteneder, F. Rendl, C., Sakamoto, D. and Inami, M. **CRISTAL. Design and implementation of a remote control system based on multi-touch system**. In Proc. of the 4th Int. Conf. on Interactive Tabletops and Surfaces: Tabletops 2009. ACM. Banff, Canada, November 23-25, 2009.

[4] Garcia Frey, A., Calvary, G. and Dupuy-Chessa, S. **Xplain: an editor for building self-explanatory user interfaces by model-driven engineering**. In Proc. of the 2nd Int. Symp. on Engineering Interactive Computing Systems: EICS 2010, pp 41-46, ACM. Berlin, Germany, June 19-23, 2010.

[5] Vermeulen, J., Lim, B. Y. and Kawsar, F. **Proc. of the Int. Ws. on Intelligibility and Control in Pervasive Computing**. Held in conjunction with the 9th Int. Conf. on Pervasive Computing : Pervasive 2011. St Francisco, CA, USA, June 12-15, 201.

Building Supportive Multimodal User Interfaces

José Coelho
LaSIGE, University of Lisbon
Campo Grande Edificio C6 Piso 3 1749-016
Lisboa, Portugal
+351 21 750 05 32
jcoelho@lasige.di.fc.ul.pt

Carlos Duarte
LaSIGE, University of Lisbon
Campo Grande Edificio C6 Piso 3 1749-016
Lisboa, Portugal
+351 21 750 05 19
cad@di.fc.ul.pt

ABSTRACT

In this paper, we describe and discuss solutions capable of helping in the development of supportive multimodal user interfaces. Based on the specifications and design of European Union funded project GUIDE (Gentle User Interfaces for Elderly People), we show how it is possible to use several modalities of interaction as well as adapting UIs, as a mean of providing users with ideal interaction in every application, and preventing or resolving errors resulting from missed or wrong user-device inputs.

Keywords
Supportive multimodal user interfaces, adaptation, GUIDE, UI translation.

INTRODUCTION

In this paper we are going to introduce some mechanisms present in the ongoing GUIDE project and which are intended to help developers in the implementation of supportive user interfaces.

GUIDE Project

GUIDE[i] aims to offer multimodal interaction to elderly (and disabled) users with the goal of simplifying interaction with a television (TV) and set top box (STB) based system. By pointing to the screen, making gestures, issuing speech commands, interacting with a Tablet PC, using the remote control, interacting with an Avatar or simply making use of user intuition for combined interaction with several of these modalities, the GUIDE framework makes fitting interaction to users' characteristics and preferences, possible and also for impaired users to interact with the TV.

In what concerns supportive interaction, the use of Avatars is explored with the goal of offering users, a persona with whom they can relate to, while interacting with the system. The Avatar will work like someone who explains to users the interaction steps to be done in order to execute tasks, and will help them getting out of "trouble" after an error has been generated while using the system. More, the existence of generic, as well as content-specific, speech commands as a possibility of interaction makes intuition a reality in GUIDE. Additionally, pointing interaction using a video based gesture tracking sensor is helped by cursor adaptation techniques which makes easier the selection of content on the screen, also helping in supporting interaction.

This diversity of devices and modalities of interaction, will offer users the flexibility to use whatever medium they find more appropriate given a specific context, at the same time as they benefit from visual (text, images, video and animations), audio (speech, and other sounds) and haptic feedback (vibration). These multimodal capabilities are in fact, the first step to a supportive interaction.

Considering the variety of differences present in elderly users and their preferences when using a system like this, GUIDE will cluster it's users in different User Profiles (UPs) - transparent to every user - where data concerning preferences and constraints of interaction are saved. By making use of each UP, GUIDE will try to adapt User Interface (UI) elements to fit every user.

In addition to providing supportive use, GUIDE framework supports UI adaptation for every application running. Moreover it aims at providing this support requesting reduced extra effort from developers. Since it is not expectable to have developers providing different versions of applications for users with different characteristics GUIDE will develop tools to "translate" a "standard" UI into tailored UIs for every type of user. The extra effort asked of developers consists in identifying each UI interactive component using WAI-ARIA[ii] semantic tags. With that information, GUIDE will abstract UI characteristics, and save them in an Application Model (AM) (one for every application), making adaptation of UI components possible at run-time.

Problem Description

Nowadays, most UIs lack capability in guiding users to an adequate and efficient interaction [1], when ideally "the UI must guide the user in accomplishing a task the application was designed for" [4] by providing help and appropriate feedback about features, tasks, modalities and contexts of interaction. If a user is not capable of perceiving an application and reacting to errors while interacting, more sooner than later he or she is going to abandon its use, and adopt a more usable application. Unsatisfied users are going to prefer a better supportive interface which can fit and adapt to his or her characteristics. If this is true for the so called typical users, for elderly users this is even more relevant. Because these users are usually characterized by having one or multiple impairments (example: hearing difficulties, visual incapacity, motor constraints, etc.), adequate interaction is only possible when the system is capable of adapting its UI components and modalities of interaction to these users' specific characteristics.

Therefore, in the development of supportive multimodal user interfaces for elderly or impaired users, several questions need to be answered so that an appropriate application and interaction can be implemented:

- How to let your users know how to interact?
- How to know your users?
- How to help users after a mistake has been identified?
- How to present content and interaction possibilities in the most suitable way to the users?

In the remainder of this paper, we describe the approaches followed in GUIDE to try to offer solutions to the questions identified above, by supporting multimodal interaction and UI adaptation for elderly users when using a TV and STB based system. Special interest also goes to the way this framework provides every application with the possibility of adapting to different contexts of interaction, and to the presentation of ideas on how it could be possible for these types of users to personalize UI presentation and interaction while preserving usability.

ANSWERING THE QUESTIONS

How to let your users know how to interact?

For an efficient interaction to be a reality, users need to have knowledge about the available ways for performing each task. They have to know to the full extent all the possibilities and modalities when confronted with different difficulties and contexts of interactions. Only by understanding how they can interact, they can make the most of the interface being presented and understand how to use all the features provided by the application. For example, if a visual interface with a menu is presented on the screen, and the user doesn't know he or she can speak the name of a specific button for making a selection, a lot of time can be lost by performing the task using alternative modalities (the only ones the user has knowledge about) like selecting the button by pressing remote control keys in a certain sequence or by pointing to the screen with the remote control.

GUIDE will try to instruct the users before they start interacting with any of the framework applications. For this, it will use an application called the User Initialization Application (UIA) to give the user a clear understanding of the possible ways of interaction. Users will be guided through the experimentation of the various modalities of interaction available in the framework, like pointing to the screen, issuing speech commands, pressing remote control buttons, etc.. For this purpose, the UIA will present on the screen a tutorial with scripted animations of how to perform different gestures, informing the user of the set of speech commands he or she can issue for achieving typical tasks, and providing instructions about how to interact with other components of the system like the Avatar engine, the Table PC, etc.. In all this process the user has an active role, learning by experimentation of every interaction modality and device.

How to know your users?

For the users to understand an interface and know how to interact with it, it really helps that the interface knows the user in advance. Only knowing beforehand what are the users preferred ways of interacting as well as the users' impairments and difficulties makes it possible to build or adapt the interface for appropriate and efficient user interaction. For example, if the system doesn't have enough information about the user to know that he or she is blind and presents a visual interface to him or her, no interaction will occur at all, and the system will not be used. In a second example, if the user prefers to interact using pointing and the system presents a simple visual interface that only receives remote control input, he or she will be less motivated to use and adopt that system (and a higher probability of making errors during interaction exists).

GUIDE will try to collect information about its users before they get to interact with any of the system's applications. To this end, the UIA will also be used for collecting data about users. Every time a new user starts using the system, the UIA is presented on the screen combined with audio output (covering possible situations of severe audio or visual impairments) and the user is asked to perform a series of tasks concerning his or her capabilities. In a first instance, the user is "registered" in the application using name, and facial and vocal characteristics, so that from that point on, every time he or she wants to use the system the correspondent UP can be loaded based on these properties. Next, the application tries to understand if the user has some visual impairment by presenting text on the screen and asking for user feedback (figure 1) (e.g. presenting a sentence and expecting for user to adjust the font until he feels comfortable reading, and then asking user to read the sentence out loud to make sure he is in fact seeing it well). If the user passes this test, different configurations of text font and buttons, as well as several background and text colors, are tested out to understand his or her preferences regarding visual interfaces. If the user fails the test, the text font size is raised in a screen-by-screen basis until there is the understanding of how severe is the user visual impairment.

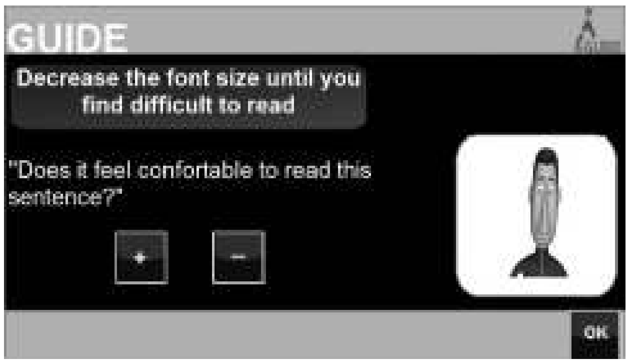

GUIDE
Decrease the font size until you find difficult to read
"Does it feel confortable to read this sentence?"
+
−
OK

Figure 1: *UIA prototype. Example of visual test where the user has to read out loud the text presented on the screen, and increase or decrease the text size to his or her preferences.*

For every other modality of interaction, similar tests are presented to the user, and data about user impairments and preferences is collected. For example, the user is asked to perform different gestures, or asked to point to different locations on the screen to understand motor capabilities, to repeat out loud what he heard to understand hearing capabilities (figure 2 top), and asked to play memory and interpretation "games" with the goal of testing his or her cognitive capabilities (figure 2 bottom).

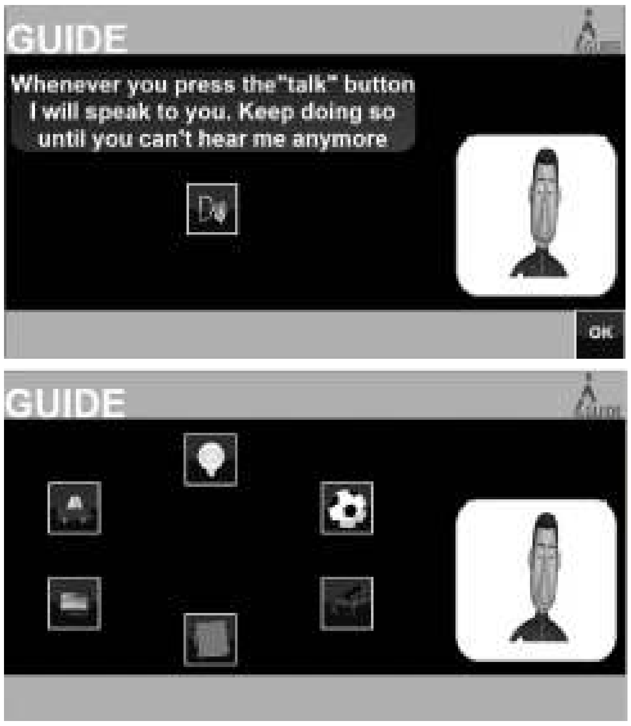

GUIDE
Whenever you press the"talk" button I will speak to you. Keep doing so until you can't hear me anymore
OK
GUIDE

Figure 2: *UIA prototype. Examples of audio (top) and cognitive (bottom) tests presented to GUIDE users.*

From the results obtained in GUIDE user trials and from discussions with developers, we also know to be extremely important that UIA application must be presented to users in form of a simple and quick tutorial, so that elderly don't feel like they are being evaluated. If UIA takes too long, users will also lose interest, and will not want to use the system.

User information can be collected explicitly with the UIA, but also implicitly through run-time analysis of the user interaction logs. After the user has gone through all the UIA process, he or she starts interacting with different applications. Information concerning every task performed and modality used is saved by the system in logs. A rule-based inference motor will analyze this data and makes conclusions about user preferences and difficulties (for example, if the user makes consecutive errors when pointing to the screen for selection of a menu button, the system concludes he or she has difficulties using that modality and tries to increase the size of the buttons before suggesting a change in the modality of interaction). These conclusions enrich the data collected in the first process.

All data collected by the UIA and run-time processes are saved in a user model and used to adapt every application running on the framework [2].

How to help users after a mistake has been identified?
A supportive UI is one which tries to be aware at all times if a user is lost in the interaction, or if he or she is having too many interaction errors to be enjoying an efficient use of the application. Accordingly, one of the biggest challenges when guiding the user in the interaction, it's how to identify or perceive that the he or she is lost and when is the application or interaction generating errors. Only after identifying that, the application can then try to help the user and suggest alternative ways to achieve a desired goal. This is, however, a difficult task because at run-time a lot of dimensions are involved. If the user mistakes or misinterprets the interface structure and meaning, it can by itself result in interaction mistakes. There are also a lot of possible errors caused by changes in the context of the interaction, like the physical and social aspects of the environment. For example, if a user is interacting using speech input and the noise in the room increases, the system can fail to interpret the command issued because of the background noise, or a wrong command can be recognized instead (this can also happen when another person is speaking to the user at the same time of interaction).

Interaction mistakes will be identified in GUIDE by analyzing the interaction in run-time and by watching for unrecognized inputs. Because in this framework users can interact with UIs through different modalities (and devices), in a singular way or in a combined fashion, the system has to be alert for many different errors like:

- Unrecognized commands issued when speech input is performed.
- Selection of meaningless coordinates (coordinates not related with any UI interactive content) when pointing with finger.
- Unrecognized gestures performed by the user.
- Errors resulting from remote control commands.

- Repeated errors when interacting with each device or modality (consecutive errors could suggest a switching of modalities is required).
- Long periods with no selection registered but with screen navigation occurring (may suggest that the user is lost, or doesn't know what to do).
- Errors resulting from incomplete fusion of input modalities.
- Contradictory instructions from simultaneous input of different modalities.
- No input received after system started a task requiring user feedback.

Additionally, every time a change in context of interaction occurs, the system has to be alert for periods of inactivity or for unexpected inputs, and using the interaction logs the system tries to prevent some errors from happening when there is clear understanding of what are the causes.

A supportive UI has to be capable of helping the users every time there is a mistake in the interaction [4]. However, in modern applications help is a capacity "created ad-hoc" [4] meaning it was previously generated and it does not cover run-time situations not originally foreseen by the designers. For this reason, UI design does not cover every situation where a user needs help for responding to UI or interaction difficulties. Therefore helping the user is not something easy to do in a predefined manner before the user starts using the system, and requires some run-time "intelligence" from the supportive system or interface. For example, if a user is using speech input for menu navigation and his or her dog enters the room and starts barking, the system will receive a series of consecutive unrecognized inputs and the user will be in a situation that was not taken care off in the design process, which can result in aborting the interaction with the application.

As it is strongly based on multimodal interaction, one of GUIDE's ways of helping users after a mistake has been identified will rely on suggesting to the user a change in the modality of interaction. This change is however, based on each user preferences and characteristics firstly identified by the UIA and logs of interaction, as well as it is based in the context of interaction and task being performed at that moment[2]. So, as the user has already "ranked" modalities of interaction by preference (and based on constraints), every time an error results from repeated errors interacting with one single modality, another is suggested to the user, who accepts it (or rejects it) in order to continue the interaction. This will also be the procedure every time a change in the context of interaction happens [2] (for example, when the dog starts barking, the system won't recognize the barks as speech commands – rather, barks will be interpreted as background noise - and will suggest to the user continuing interacting using pointing).

Another way of helping users is to present to them relevant information related with the context of the error they have just made, like presenting alternative modalities of interaction and showing how to use them when a change in the context of interaction happens, showing information related with the task they are performing every time there are errors in the recognition of modalities or long pauses in the interaction (for example when the user is pointing and trying to select an area on the screen where there are no interactive UI items, show him or her where the buttons are by highlighting them). However, GUIDE main focus is helping users proceed with the interaction in an alternative way even when it's not possible to detect the cause of the error.

Finally, every time an error occurs, the Avatar engine will also be called for a more "personal" interaction between the system and the user (meaning, the Avatar presents the explanation of the error to the user, shows how changing modalities can solve the problem or just points the user to using an alternative modality when an error arises). In this way, it's almost like together they can find a solution to the problem or "find a way out" of the mistake.

How to present content and interaction possibilities in the most suitable way to the users?

The main problem with developing interfaces for elderly or disabled users is the great diversity existing in terms of user characteristics and user impairments. It is common for an elderly user to have more than one impairment (for example, poor hearing and poor vision), as it is usual to observe a lot of differences between each of these users. This means that what is good for one user can also, and at the same time, be inappropriate for several others. For example, an elderly user with hearing difficulties can interact with a visual interface without any problem, but one with severe visual impairments cannot, and need an interface with audio input and output for efficient interaction. However it is not expectable that developers will implement different versions of the same application, so the framework has to ensure the ways of interaction are adapted to the user characteristics.

GUIDE will offer elderly users adaptation mechanisms capable of adapting UI elements to each user characteristics. After the user has gone through the UIA and the system has collected enough information, the user is assigned to one UP [1]. Using the information about each user, GUIDE adapts each UI to fit the UP interaction patterns. This is only possible because GUIDE asks for extra information in each application development, so every UI is implemented using HTML, JavaScript, and CSS languages to what the developers add WAI-ARIA[ii] annotations providing semantic information about UI components. In this way, for every application, GUIDE will derive and keep an Application Model (AM), which is nothing more than an abstract interface that saves information about the structure of the UI and identifies each UI element present. This facilitates adaptation to different interaction contexts as well as to different types of users (users that belong to different UPs), because every time a user calls for an application, the system uses its application model and considering the interaction context and user characteristics, modifies UI elements not appropriate for the user. For instance, when a user with visual impairments calls for an application formed by a visual menu and some text content, GUIDE consults its AM and "knowing" the user characteristics as well as "observing" no change in the interaction context, loads the UI increasing the size of the buttons originally defined and uses audio and visual output modalities.

In what concerns the developers control over this UI adaptation, GUIDE will adopt one of three adaptation schemes depending on the level of freedom given by the developer to change the application original properties (CSS and HTML): In "Augmentation", GUIDE won't be able to change any UI components, only making some overlay of output modalities (for example, adding audio output to a visual interface); in "Adjustment" GUIDE has permission to adjust UI component parameters as well as also making "augmentation" (for example, adding audio output to a visual interface and also changing UI colors for a higher-contrast); and finally in "Replacement" the developer gives total control to GUIDE, making possible the substitution of UI components as well as "augmentation" and "adjustment" (for example, adding or removing buttons, as well as adjusting colors and adding audio output to a visual interface).

Additionally, all interfaces must be capable of listening for user commands at any time of the interaction so that modifications to the interaction and presentation can be done at run-time, if the user is not satisfied with the current configuration. For example, if a user says "bigger buttons" or makes a gesture to increase the volume, the interface must adapt and reflect these changes (by reloading the UI or modifying output parameters).

CONCLUSIONS

For the development of supportive multimodal user interfaces to be a reality, we have to make sure that the user's characteristics are known to the application. As well, the application has to be capable of instructing the users about all the ways of interacting with it, and make sure that adaption and UI help is presented to users in a personalized fashion. GUIDEs UIA, multimodal interaction and UI translation and adaption, were presented in this paper as possible solutions which can help in the deployment of supportive user applications without asking much more additional effort from the developers.

REFERENCES

1. Biswas, P., Langdon, P.: Towards an inclusive world – a simulation tool to design interactive electronic systems for elderly and disabled users. Proc. SRII, 2011.
2. Coelho, J., Duarte, C.: The Contribution of Multimodal Adaptation Techniques to the GUIDE Interface. In: Stephanidis, C. (ed.): Universal Access in HCI, Part I, HCII 2011, LNCS 6765, pp. 337-346. Springer, Heidelberg (2011)
3. Garcia Frey, A., Calvary, G. and Dupuy-Chessa, S. Xplain: an editor for building self-explanatory user interfaces by model-driven engineering. In Proc. of the 2nd Int. Symp. on Engineering Interactive Computing Systems: EICS 2010, pp 41-46, ACM. Berlin, Germany, June 19-23, 2010.
4. Myers, B. A., Weitzman, A. J. Ko., and Chau, D. H.. Answering why and why not questions in user interfaces, in CHI'06: Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 397-406, New York, NY, USA, 2006. ACM

[i] GUIDE– Gentle User Interfaces for Elderly People. http://www.guide-project.eu/.

[ii] http://www.w3.org/WAI/intro/aria

Opening the Box
Meta-level Interfaces Needs and Solutions

Alan Dix
Talis, 43 Temple Row, Birmingham, B2 5LS, UK
and Birmingham University, Edgbaston, Birmingham, UK
alan@hcibook.com
http://www.hcibook.com/alan/papers/SUI2011-meta/

ABSTRACT

This paper begins by considering reasons why some form of meta-level interface may be required for modifying or exploring existing user interfaces, from obvious functional reasons of customisation and personalisation to more political and social goals such as education and empowerment. The paper considers examples of systems developed by the author and others, and uses these to present a number of techniques and principles for effective meta-interactions. Some of these concern more surface manipulation, and others deeper levels of code and meta-descriptions of the application and UI. It concludes that meta-interaction may be a key element for future liberal society.

Keywords
customisation, personalisation, end-user programming, end-user empowerment, appropriation

1. INTRODUCTION

The topic of this workshop brings together a number of areas on which I have worked or that have been of personal concern. This paper will discuss some of these areas of concern and then look at general principles and techniques that can be used to address them.

2. WHY META?

While it hardly needs stating for this workshop, to many it may seem that meta-level interactions are simply the preserve of the hobbyist or techie. However, they are both ubiquitous and of broad benefit.

2.1 Customisation and Personalisation

Of course meta-level user interfaces are common. Every time a user drags a palette to the side of the screen, selects a ringtone or modifies the style definition in a document, she is engaging in an adaptation of the user interface. However, we also know that beyond a few examples like this few users actually customise despite having problems or gripes that could be dealing with through simple selection of options (for example, turning off some of the 'smart' features in Word). Improving even these basic features can have a major impact on user experience.

2.2 Appropriation

In particular "plugability and configuration" is one of the design principles for appropriation [9]. Indeed several of the design principles discussed in [9] are related to meta-level user interactions; while appropriation is possible using the interface as given, the user has greater flexibility if she can peek under the hood (design principle "provide visibility") and tinker inside ("plugability and configuration") and share the results with others ("encourage sharing").

2.3 End-user Empowerment

One advantage of appropriation is the sense of ownership and empowerment it engenders. A sense of control is important for well being, and the act of tinkering gives this, whether to improve the user interface for its original purposes, or make it do something completely novel.

While this is important for all users it is particularly relevant for those in developing countries, or the disadvantaged in developed countries, who can be doubly disadvantaged in a world where access to information is central to economic and political power [1].

Existing technology can be appropriated by traditionally disadvantaged groups; for example, Jensen reports how mobile phones allowed fishing boats in Kerala, southwest India, to obtain higher prices for their catches [12] and we have all seen the impact of social media in recent popular uprisings across North Africa and the Middle East.

However, if those closer to need are in a position to create, modify or adapt existing software and hardware the results are likely to be more appropriate than tools designed primarily for an urban, middle-class, western environment. This may be the end user, but Marsden et al. argue the case for 'human access points', local experts, in their case local health workers, who are given the tools to create and adapt mobile-phone administered questionnaires [16]. Prompted by various workshop discussions [17, 20], we have explored the potential for a range of mobile phone-based adaptations including compete coding via the mobile-phone screen [10].

2.4 Education

Often modifications to user interfaces require a high degree of expertise; so education is needed in order to use them. However, if well designed, meta-level interactions hold the potential to be a means for education in themselves; as generations of children who have fiddled with old car engines can testify. Education, of course, also contributes to empowerment.

The Query-by-Browsing (QbB) intelligent database interface is an example of this. QbB generates SQL: queries based on user record preferences, but then reflects this back to the user both by highlighting the records selected by the query and by exposing the query itself [7]. The user can comprehend the system via the concrete record selections, but in the process learn the SQL that produce it (although not the machine learning algorithms which generate the queries).

2.5 Privacy and Auditability

The control of privacy settings in social applications such as Facebook, has become a big issue. Höök also argues that this is an issue likely to be important in future ubiquitous computing applications [11]. Indeed the very openness in low-level architecture required for rich context-sensitive features in itself creates privacy issues [8]. Many approaches to privacy, in ubiquitous computing and elsewhere, focus on restricting information flow. However I have long argued that it is the eventual *use* of the information that is most critical [6]; that is systems that expose what happens to information both currently (visibility) and in the past (auditability) are far more likely to support the user's ability to manage information disclosure.

2.6. Comprehensible Behaviour and Trust

Closely related is the issue of trust, not just for financial and or personal security, but also at a mundane level of whether we decide to use particular application features. This is especially important when systems make choices automatically for us. The kind of openness needed to allow a user to adapt a system is very similar to that needed to allow a user to believe in what it is doing already.

The record listings in Query-by-Browsing [7] are an example of this as they may be comprehensible to the user, even if the SQL is not, giving the user confidence that the query will continue to be appropriate for unseen records. Another example is MICA, which makes suggestions for GUI customisation based on user activity, but also "*includes a description of why MICA is making recommendations and how it generated them*" [5], precisely to support Hook's "predictability and transparency" principle [11] and so engender trust.

3. TECHNIQUES AND PRINCIPLES

So if meta-level investigation and modification is a good thing, how can it be achieved?

3.1 Cost and Benefit – When it happens

Sometimes people don't customise because they don't know how. However many experts do not customise their interfaces even if they complain about the things that are wrong! The key problem is not lack of understanding but lack of immediate benefit. We are creatures who heavily discount the future; effort now for future gain is hard. If customisation can be made closer to the point of use it becomes more likely. One example are dialogues that ask for a decision, but have a tick box to say "always do this". This is effectively asking you set a preference, but at a point in time when you are in the middle of doing the requisite action. The benefit is clear and the cost (in terms of clicks and mental effort) low. Furthermore this is all set within the context of a concrete example of use (see also next point)

3.2 Progressive Disclosure –Where It happens

The preferences and customisation of many applications are buried in a "preferences" menu item far away from the actual interaction. Somewhere in a preferences panel you set parameters whilst guessing vaguely what they might be about. However, others connect customisation closer to the thing it affects. Back in 1995, Marsden [15] advocated the advantages of a systematic policy suggesting a 'screw' metaphor where every component has a small screw icon in the bottom right hand corner. Clicking the screw 'undoes it' revealing the circuitry within, and potential the ability to unscrew other sub-components (see Figure 1).

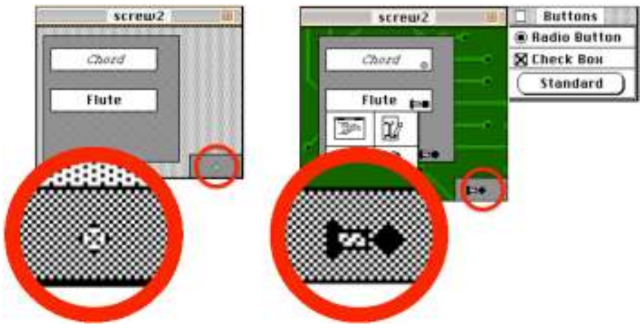


Figure 1. Screw Metaphor from [15]
(a) screw in – UI (b) screw out – metaUI

Today in the Apple Dashboard just such a mechanism is found on widgets. Instead of a screw a little 'i' for information icon, clicking it 'turns around' the widget showing settings behind. Strangely the iPhone reverted to a special place for settings rather than associating them closely with their application.

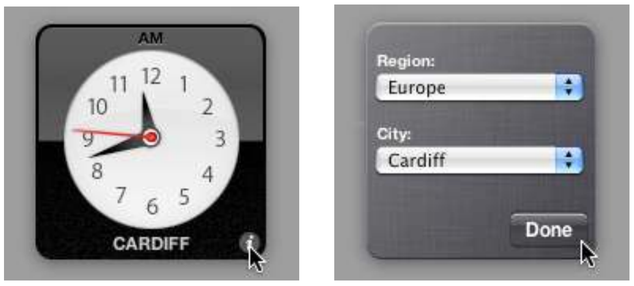


Figure 2. Mac OS Dashboard widget
(a) front – UI (b) back– metaUI

3.3 Tools of Revelation

A similar approach is to use some form of external 'tool' for meta-level modifications. This happens in the real world; Figure 3 shows a stud detector, which detects the wooden studs in a wall so that you can screw into them. The wooden structure is hidden behind plasterboard and wallpaper, but the stud detector reveals it – the "provide visibility" appropriation principle [9] in the physical world.

Figure 3. Wall Stud Detector

Note that "provide visibility" does not mean the same as Nielsen's "visibility of system status" evaluation heuristic [19], as this usually refers to the essential information about the system for normal use. Instead, if systems reveal a little more (such as a mobile phone showing signal strength not just whether or not a call can be made), then the user can use this in unexpected ways (such as waving the phone about to seek out better signal).

Beaudouin-Lafon's 'instrumental interaction' [2] and in particular Toolglasses [3], follows the same principle as the stud detector advocating the use of 'instruments' as a means for modifying and interacting with objects.

3.4 Smooth Transitions

When creating means for user to modify their environment there is often a temptation to try to do everything – the spectre of Turing equivalence rises and before long a simple end-user customisation tool becomes a full-blown and complex programming language. The effort to produce something that could, *in principle*, do everything often ends up with something that, *in practice*, is good for nothing. However, the alternative is often to have very different means for simple and more complex modifications, so that users hit barriers; for example, moving from Excel formulae to Visual Basic.

Mathematicians face a similar problem when modelling 'differential manifiolds' curved spaces such as the surface of the Earth or the curved space-time of general relativity. They effectively paper the curved space with flat Euclidean surfaces (which are easier for a mathematician to handle), but if you try to use a single flat surface there is at least one point where things go very wrong, like the place where the foil is all folded up at the end of an Easter egg. Instead mathematicians use a collection of small patches, which overlap in a 'smooth' manner.

One can envisage customisation working like this, with different levels of customisation (perhaps ending up at open-source code), where the two ends (use and coding) have a huge gulf between them, but where each pair of successive levels overlap with an easy transition. This sounds like a hard problem, but there are examples that achieve this to varying extents. HyperCard had a smooth transition from use to customisation and then to programming. In consequence, many who would never consider themselves programmers created complex HyperCard applications. Xerox Buttons were another example, where a non-technical user might just use the button, then peek at its code and change a file name, and perhaps, over time, start to understand some of the code that drove the familiar user-interface actions [14]. Could the Excel formula to VB step be more like this?

3.5 Ease of collaboration

Another of the appropriation principles is "encourage sharing" [9]. In Nardi and Miller's classic study of spreadsheet use [18], they describe the collaboration between Buzz and Betty

"When Buzz helps Betty with a complex part of the spreadsheet such as graphing or a complex formula, his work is expressed in terms of Betty's original work. He adds small, more advanced pieces of code to Betty's basic spreadsheet: Betty is the main developer and he plays an adjunct role as consultant."

The fact that spreadsheets have relatively smooth transitions (at least between levels of formula use) make this collaboration possible. Note especially that Betty is able to do a lot herself, and probably extends this over time (education). Furthermore Betty is able to determine her own level and understand when to seek help.

Spreadsheets, by their nature allow them to be passed around. It is far rarer to see other kinds of configurations shared. In UNIX systems, a lot of configuration is in text files, such as .login or .profile, and expert users will move these around. However, it is near impossible to simply take one person's Word settings and apply them to another users machine. Xerox Buttons [14] were a simple idea, a button that executed some Lisp code, but were surprisingly powerful, in part because you could *mail them round*, creating a community. Maker cultures emerge when people can share ideas and, even better, artefacts.

3.6 From Configuration to Code

Spreadsheets, Xerox Buttons, Query-by-Browsing and HyperCard are all examples where the user can move in steps from doing things to raw coding. When looking at near-end-use development, one of the design lessons was "reduce the gap between design and execution" [10].

"*In general, bridging the gaps between environment and language, design and use, test and bug report [...] features found in many end-user or near-use software such as spreadsheets (eliding data, code and execution), Yahoo! Pipes (design close to execution), and programming by example (use is design)*"

At Talis we are working on tools to bridge this gap for linked open data [4] as exposed, for example, in data.gov.uk. This is building on Callimacus, where RDFa embedded in a web page turns it into a UI generation template, opening up application building to ordinary web developers [13].

3.7. Meta-Representations for Meta UIs

As well as being the subject of user interaction, semantic data of some form seems to be a key element of future user interactions. Whether mashing data for the web or connecting digital devices in the living room, effective meta data about devices, applications and their interactive potential seems an essential start point for more flexible machine initiated activity, for machine activity to be explicable, and for users to be able to interrogate and modify it. Model-based user interfaces are clearly one way to achieve this, but there could be other solutions, similar to the way applications expose meta-information for Apple Scripting on Mac OS or via COM on Windows.

4. CONCLUSIONS

We have discussed various principles and methods for meta-level interactions., and also some of the reasons why this is 'a good thing'. As we enter an era of open data and mashups the ability to digitally tinker seems not just a hobby, but a key enabler of a broad-based civil society.

REFERENCES

1. Beardon, H., Munyampeta, F., Rout S. and Maiso Williams, G. ICT for Development, Empowerment or Exploitation: Learning from the Reflect ICTs project. ActionAid. (2005) http://www.actionaid.org.uk/1413/ict_for_development_empowerment_or_exploitation.html
2. Beaudouin-Lafon, M. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. in *Proc. of the CHI '00.* ACM Press, (2000) 446–453.
3. Beaudouin-Lafon, M and Mackay, W. Reification, polymorphism and reuse: three principles for designing visual interfaces. in *Proc. of AVI '0)*. (2000) ACM Press ,102-109.
4. Bizer, C., Heath, T. and Berners-Lee, T. Linked Data – The Story So Far. *International. Journal on Semantic Web and Information Systems*, 2009.
5. Bunt, A., Conati, C., and McGrenere, J. Supporting interface customization using a mixed-initiative approach. In *Proc. of IUI '07*. (2007) ACM Press. 92-101.
6. [Di90] Dix, A. (1990). Information processing, context and privacy. in *Proc. of INTERACT'90*, (1990) North-Holland. 15–20. http://hcibook.com/papers/int90/
7. [DP94] Dix, A. and Patrick, A. (1994). Query By Browsing. Proc. of IDS'94: *The 2nd International Workshop on User Interfaces to Databases*, (Lancaster, UK, 1994), Springer Verlag. 236–248. http://hcibook.com/alan/papers/QbB-IDS94/
8. Dix, A. Beyond intention – pushing boundaries with incidental interaction. Proc. of *Building Bridges: Interdisciplinary Context-Sensitive Computing*, (Glasgow University, 9 Sept 2002) http://hcibook.com/alan/papers/beyond-intention-2002/
9. Dix, A. 2007. Designing for appropriation. In *Proc. of HCI2007 Volume 2.* (2007). BCS, 27–30. http://www.bcs.org/content/conWebDoc/13347
10. Dix, A., Kozhissery, R., Ravichandran, R. and Dayanand, D. Content Development Through the Keyhole. in *Proc. of EISE2009, Expressive Interaction for Sustainability and Empowerment*, (2009) 67–78. http://hcibook.com/alan/papers/EISE2009-Keyhole/
11. Höök, K. Steps to take before intelligent user interfaces become real. *Interacting with Computers*, 12 (2000) 409--426,.
12. Jensen, R. The Digital Provide: Information (Technology), Market Performance, and Welfare in the South Indian Fisheries Sector, *Quarterly Journal of Economics*, 122(3), (2007) 879–924.
13. Leigh, J. and Wood, D. RDFa as a Query Language. *Semantic Technology Conference*. (June 2010)
14. MacLean, A., Carter, K., Lövstrand, L., and Moran, T.. User-tailorable systems: pressing the issues with buttons. In *Proc. CHI '90.* (1990) ACM Press, 1990, 175–182
15. Marsden, G. Overcoming Design and Execute Modes in User Interface Design Environments. in *Proc. of HCI 95 people and Computers* (1995), 133-137
16. Marsden, G., Maunder, A. and Parker, M. People are people, but technology is not technology. *Phil. Trans. R. Soc. A* (2008) 366, 3795–3804.
17. *Mobile Design Dialog*. (Cambridge. 3–4 April 2008) webpage: http://www.cs.swan.ac.uk/mobdesign/ Mobile Design Dialog discussion: http://mobiledesigndialog.nexo.com/
18. Nardi, B.and Miller, J. An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work (CSCW '90).* (1990) ACM, Press, 197–208.
19. Nielsen, J. Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods*, (1994) John Wiley & Sons, New York, USA.
20. *Winter School on Interactive Technologies*. (HP Labs in Bangalore, 2nd & 3rd February 2009). UK-India Network on Interactive Technologies. http://www.ukinit.org/02122008/winter-school-interactivetechnologies

A Classification of Self-Explanatory User Interfaces

Maximilian Kern, Marco Blumendorf, Sahin Albayrak
DAI-Labor
Technische Universität Berlin
Ernst-Reuter-Platz 7
10587 Berlin, Germany
forename.surname@dai-labor.de

ABSTRACT
In this paper a definition of Self-Explanatory User Interfaces (SEUI) is proposed. Furthermore, existing approaches on SEUIs are classified by identification of their significant features. Derived from these features, challenges and open issues are elaborated. Then, advantages of a model-based approach for the development of SEUIs are given. Finally, a conclusion is given with an outlook on an ultimate SEUI from the author's perspective.

Keywords
assistance, guidance, self-explanatory UI, adaptive UI, meta UI, MDUI

INTRODUCTION
The term Supportive User Interface (SUI) has been introduced recently and still needs to be defined clearly. In our work, we understand the term supportive as the goal to support the user while interacting with a user interface. Support thus aims at the ability of a user interface to provide optimal interaction capabilities and the necessary configuration options therefore as well as help for the user to understand the rationales of a user interface, provide a context-sensitive help if the user is lost in navigation or requests help explicitly. In this paper we focus on Self-Explanatory User Interfaces as a subtype of SUIs that especially emphasizes the help as explanatory features of SUIs.

The earliest approaches for built-in support on an interactive system emerged around 1966 with the HELP system developed under the Genie project [12]. The HELP system provides answers to questions about commands and entities available on a UNIX based terminal window. While such approaches were restricted to low computing performance at this time, the ongoing technological improvements enables recent assistants being capable of understanding, interpreting and speaking human language, capturing and considering context information and learn from users by observing their interaction. In the following, we propose a definition of SEUIs. Furthermore, we clarify the term SEUI and classify existing approaches by analyzing their features. Afterwards, challenges of the development of SEUIs are discussed. Then, we discuss how SEUIs can benefit from model-based development. Finally, a conclusion on SEUIs is given with an outlook to an ultimate SEUI.

A DEFINITION OF SEUI
Self-explanatory user interfaces in general are characterized and thus, can be defined by the ability to reason on the application state and generate additional explanations or useful hints of higher value which support users in fulfilling their desired task faster. Therefore, SEUIs introspectively read out information hidden from the actual user interface and evaluate them. By these hints, the user gains deeper insight of the rationales in terms of purpose and structure of the application [6]. Advanced SEUIs are generic by means of that they adapt at runtime to the current context-of-use and they are not bounded to a specific domain. In this manner, their characteristics conform to those of meta UIs and thus, can be comprehended as a kind of meta UI. By taking the idea of an SEUI being able of accessing and reasoning on artifacts of other applications or domains, SEUIs can be thought of to be an ever-present agent or companion who intermediates between the user and the applications. Depending on its mightiness, it is not only giving hints generated out of the underlying application but is also able to interact on behalf of the user. The agent could make use of natural language processing (NLP) and understanding (NLU) and the user can establish a dialog with him. Users could then accomplish their task by cooperatively talking to the agent. For instance, in [13] an information-seeking chat bot is presented. This chat bot supports a tourist resided in Potsdam to find sight-seeing places and gain background information related to those places such as architects, historical persons, entrance fees and public transports. It integrates an ontology with topic maps applied as the discourse of the dialogue with the user.
Furthermore, this approach utilizes templates for generating natural utterances which wrap the requested information.

FEATURES OF SEUIs
Existing approaches on self-explanatory user interfaces differentiate mainly in the way, how they appear to the user (Appearance) and how they are activated (Trigger). Furthermore, they can be distinguished by the type of knowledge base they are using and their scope or

mightiness. Figure 1 gives an overview of the identified features. These features are discussed in greater detail in the following sections.

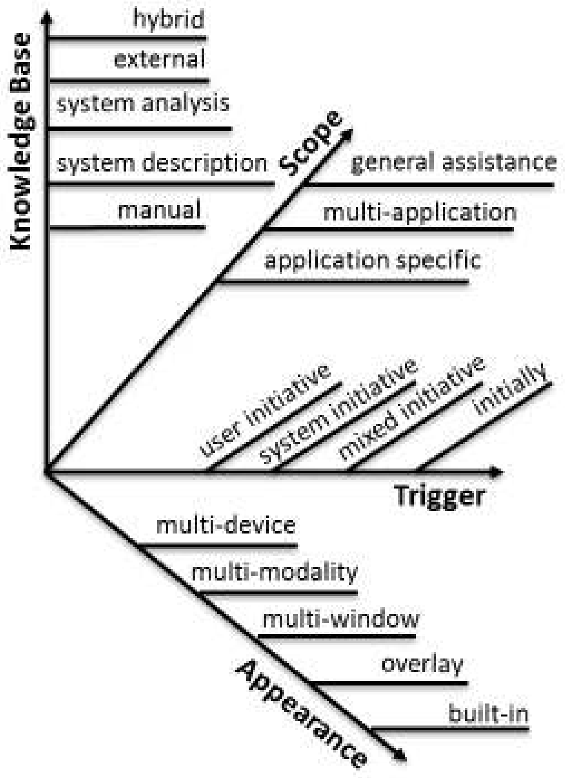


Figure 1. Overview of identified features of SEUIs

Appearance

The appearance of SEUIs is manifold. However, we can distinguish 5 basic ways of interaction:

1. Multi-device: shows the assistant on another device.
2. Multi-modality: utilizes one modality for the UI (e.g. graphics) and another one (e.g. voice) for assistance.
3. Multi-window: combines UI and assistance on one device and modality e.g. by using multiple windows, different voices or split screens.
4. Overlay: puts the assistance over the application which makes it easier to directly refer to specific elements.
5. Integration: integrates the assistance as part of the application so that the user perceives it as part of the application.

An example for multi-window, more in detail a split screen mode was applied in the DiamondHelp system introduced in [11], where the user still remains able to manipulate the underlying user interface directly. The user can choose between a ’guided’ interaction in form of a chat with the system or ’unguided’ interaction by interacting with classical user interface elements such as buttons, labels, etc.. Overlay mode is emphasizing the character of meta UI by overlaying the guided user interface in order to reach the user. This mode was applied to the MASP Guide [8] and is illustrated in Figure 2.

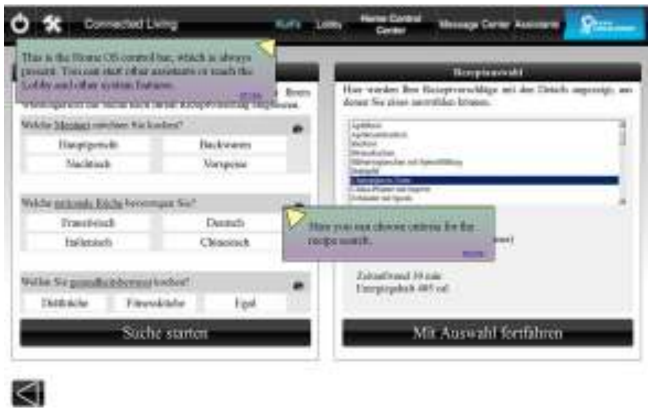


Figure 2. MASP Guide in overlay mode

Triggers

SEUIs either propose hints to the user pro-actively (system initiative) or the user is explicitly asking for help (user initiative). A third mode is called mixed-initiative which is a combination of both. A proactive SEUI, where the system takes over initiative, needs to recognize when support is actually required by the user. In order to be able to recognize the need for guidance, one option is to observe interaction history of a specific user and reason on the collected information. In [1] for instance, task models are used to connect sequences of observed user interactions to abstract tasks. Based on this information, possible interactions of users are predicted and could be proposed as a solution to the user. In [7] an approach for initial help is presented, which helps users using an application for the first time, i.e. it initially gives hints on startup. An important and reasonable issue for system initiative is to keep support decent in the sense of that the user is not flooded with hints and suggestions on what he is able to do next. In detail, system-initiated, self-explanatory user interfaces subtly appear in the moment, the user is lost in navigation or explicitly requests help. For the case the user explicitly requests help by asking for instance “Why does the menu bar appear all the sudden on the right hand side?”, the SEUI may find the reason by analyzing the adaptation history and finds that the user is right-handed and switched using a touchscreen and they should not cover the user interface with her right arm. The crystal framework proposed in [9] enables the user to ask a wide variety of why-questions, the answer is generated by introspection of the current state of the application.

Knowledge base

Another aspect is the source from where to retrieve information for giving the user desirably useful hints. One option is that the designer or developer of the SEUI is manually identifying possible critical states of the user interface at design time. Practically, due to the nature of adaptive user interfaces, this is difficult since the designer might not be able to foresee each state of the application during runtime (Even if she could, she should prevent critical situations at design time by revising the design of the application.). Thus, it is preferable that the supportive user interface is giving generic support during runtime. At this time, the SEUI can retrieve information either from the system description or from an external resource, e.g. the Internet. The former option would require that the system description offers more information than is held on the surface of the user interface and this information is available during runtime. By this way, hints are generated out of hidden artifacts of the system description. The latter option represents a bigger challenge since the information on the Internet needs to be matched to a machine-processible structure, i.e. a structure which is comprehensible by the SEUI. For this purpose, the use of some kind of ontology matching or well-formed source is inevitable. In [4] a hybrid approach “The Companions” is introduced, which is able to incorporate knowledge retrieved from local resources, but also from a social network or news site into a local rdf-based knowledge base (KB). In order to give the user the impression of talking to a human, the face of the avatar is displayed. The system was designed to enrich photo albums with semantic information about recognized people and places such as their relations or detailed information.

Scope

The previously mentioned possibility of retrieving information from an external resource yields to another aspect of SEUIs - the scope of an SEUI. Generated hints might be more useful to the user when the SEUI has knowledge which goes beyond the intended domain of the application, i.e. it has also knowledge of other applications and their domains. For instance, for an interactive application for preparing recipes, the SEUI gives reasons if a step is not feasible due an electric device is missing, which is controlled by another application for device management. Mightiness of an SEUI is addressing the potential of controlling the application itself or other applications. For instance, if a user asks for a missing device, the SEUI can implicate that the user wants to use the device and activate the device in the device management application. General assistance applies for fully generic SEUI approaches. Such approaches require no certain structure from the guided application.

CHALLENGES

Based on the identified features, we can identify various challenges for the development of SEUIs. The major general issue of giving support to the user is the understanding of the user and their needs. Getting this right is crucial so the user actually feels supported rather than annoyed. The users are playing the key-role in HCI, so they should not be displeased by the amount of hints and the moment hints are communicated by the system.

This directly leads to the appearance of the SEUI. It should please the user without disturbance and therefore needs to be well designed and provide the necessary integration into the application depending on the needs. Learning from many bad examples of help systems, it seems advisable to provide some kind of adaptation and personalization capability, which allows the continuous adaptation, based on the users behavior, and also requires the continuous monitoring of adaptation results and the performance of the help system in terms of user satisfaction.

Looking at the triggers to start the assistance, system-, user- and mixed initiative also pose different challenges. A system-initiative SEUI needs to be aware of situations, where users are not certain of how to proceed, and then find a reason (and a solution) in order to solve the users’ problem. For instance, Microsoft’s Paper Clip discourages users due to the lack of information about the context-of-use, i.e. it is not aware of the context. For user-initiative SEUIs the major issue lays in the ambiguity of a user’s utterance, the system has to rely on the terms of the current domain, current task and the discourse of the user interface, i.e. it needs to be aware of the system state.

The issue of ambiguity then also refers to the knowledge base (KB) of an SEUI. As discussed earlier, the usage of an ontology or presumption of certain structures of the KB is inevitable. Then, the challenge is accounted to the quality of the ontology matching algorithm and the way of extracting and processing information. Furthermore, this quality depends also on the fineness of the world knowledge and common knowledge for SEUIs with knowledge which goes beyond the intended domain of the application.

Relating to the scope of SEUIs, there might not be one best way for supportive UIs. It depends on the needs of the user, the usage situation and the application if SEUIs are integrated parts or separate applications. Being external applications, this however also poses requirements on the application in terms of traceability of the current state and access to design information and semantic meaning of elements. An application might need to conform to a specific structure in order to integrate self-explainability. This has direct impact on the effort for application developers/designers, which should be ideally minimal. Thus, the challenge is to develop an open or standardized programming/controlling interface for applications in order to ease integration of SEUIs and access application knowledge.

A MODEL-BASED APPROACH TO SEUI DEVELOPMENT

From our point of view, model-based development comes along with major advantages in order to cope with previously mentioned challenges. Models provide explicit information about the application state and the contextual space instead of weaving information in unstructured program code. For the sake of separation of concern, information is held in several models each covering a certain aspect (e.g. context model, interaction model, abstract UI model, concrete UI model, final UI model, etc.). An SEUI can access this information easily and needed information can be retrieved from these models. For inferring on semantics, the SEUI benefits from the self-explanatory nature of models. The MASP has built-in features for monitoring the application state and interactions [2], which lower the development effort for recognizing trigger situations of an SEUI. Another model-based approach on Automated Usability Evaluation (AUE) described in [10] is simulating a user model at run-time in order to identify lacks in usability. This approach could also be applied in order to identify problematic states of an application during runtime and provide hints to the user (for system-initiative SEUIs). Models have been proposed and utilized as basis for adaptive systems [2][3][5]. Regarding the appearance, an SEUI integrated into such systems needs to be as adaptive as the surrounding environment.

CONCLUSION AND OUTLOOK

Self-explanatory user interfaces raise supportiveness of user interfaces significantly. We have proposed a definition for SEUIs, which is "the ability (of a user interface) to reason on the application state and generate additional explanations or useful hints of higher value which support users in fulfilling their desired task faster.". It was stated that SEUIs mainly differentiate in their activation mechanism (user-/system-/mixed-initiative, initially), their appearance (multi-device, multi-modality, multi-window, overlay, built-in), their knowledge base (manual, system description, system analysis, external, hybrid) and their scope (application specific, multi-application, general assistance). We are conscious that our classification is not completive but consider it as a first step towards a better understanding of SEUI as a special kind of SUI. The challenges and open issues on SEUI lay in the design and the understanding of users and their needs. Furthermore, it was elaborated, how development of SEUIs can benefit from a model-based approach.

As a conclusion, the ultimate SEUI from our perspective is a companion, which is ubiquitously accessible and provides useful hints at any time. It would only take initiative if a user needs help and would incorporate knowledge beyond the current application's domain. For retrieving external information, it would apply approved algorithm for matching terms against ontologies. In order not to allocate space on the screen, the user could communicate entirely via voice, but it remains optional for overlay mode. Moreover, the SEUI would act in the same way as an expert knowing your personal needs and observing any of your interactions.

REFERENCES

1. Bezold, M. Describing user interactions in adaptive interactive systems. In UMAP (2009), 150–161.
2. Blumendorf, M., Lehmann, G., and Albayrak, S. Bridging models and systems at runtime to build adaptive user interfaces. In EICS '10: Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, ACM (2010).
3. Clerckx, T., Luyten, K., and Coninx, K. Dynamo-aid: A design process and a runtime architecture for dynamic model-based user interface development. In EHCI/DS-VIS (2004), 77–95.
4. Dingli, A., Wilks, Y., Catizone, R., and Cheng, W. The companions: Hybrid-world approach. In International Joint Conference on Artificial Intelligence (IJCAI)(Pasadena, CA, 2009).
5. Duarte, C. Design and Evaluation of Adaptative Multimodal Systems. PhD thesis, Department of Informatics, University of Lisbon, March 2008. DI/FCUL TR-08-9.
6. García Frey, A., Calvary, G., and Dupuy-Chesa, S. Xplain: an editor for building self-explanatory user interfaces by model-driven engineering. In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems, EICS '10, ACM (New York, NY, USA, 2010), 41–46.
7. Kang, H., Plaisant, C., and Shneiderman, B. New approaches to help users get started with visual interfaces: multi-layered interfaces and integrated initial guidance. In dg.o '03: Proceedings of the 2003 annual national conference on Digital government research, Digital Government Society of North America (2003), 1–6.
8. Kern, M., Trollmann, F., Blumendorf, M., and Albayrak, S. Adaptive user interface assistance in smart environments. In Proceedings of the Workshop on Meaning and Matching (AISB2010), De Montfort University Leicester, SSAISB (2010).
9. Myers, B. A., Weitzman, D. A., Ko, A. J., and Chau, D. H. Answering why and why not questions in user interfaces. In CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems, ACM (New York, NY, USA, 2006), 397–406.
10. Quade, M., Blumendorf, M., and Albayrak, S. Towards model-based runtime evaluation and adaptation of user interfaces. In Proceedings of International Workshop on User Modeling and Adaptation for Daily Routines (UMADR2010): Providing Assistance to People with Special and Specific Needs (2010).
11. Rich, C., and Sidner, C. DiamondHelp: A generic collaborative task guidance system. AI Magazine 28, 2 (2007).
12. Roberts, R. Help: a question answering system. In AFIPS '70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference, ACM (New York, NY, USA, 1970), 547–554.
13. Stede, M., and Schlangen, D. Information-seeking chat: Dialogue management by topic structure. In Proceedings of the 8th Workshop on Semantics and Pragmatics of Dialogue, CATALOG 04, Barcelona, 2004 (2004).Autonomous Systems (ICAS 2008)

Supportive User Interfaces in Adaptation

Víctor López-Jaquero
Laboratory on User Interaction & Software Engineering (LoUISE)
University of Castilla-La Mancha
02071 Albacete, Spain
+34 967 59 92 00
victor@dsi.uclm.es

Francisco Montero
Laboratory on User Interaction & Software Engineering (LoUISE)
University of Castilla-La Mancha
02071 Albacete, Spain
+34 967 59 92 00
fmontero@dsi.uclm.es

ABSTRACT

In this paper a discussion of how supportive user interfaces can be used in user interfaces with adaptation capabilities is provided. This discussion in made using as reference ISATINE adaptation framework, where the stages for a proper adaptation process and the tasks the user can get involved during the adaptation process are clearly described. Moreover, some open questions are enunciated to help in the identification of open issues in supportive user interfaces field.

Keywords

Supportive user interfaces, adaptation, ISATINE adaptation framework

INTRODUCTION

The growing complexity of the applications being currently developed to match complex functionality requirements, and the myriad of situations where the users want to interact with those applications has provoked the creation of complex user interfaces. Nevertheless, the complexity of the user interfaces produced together with the great number of features available in the user interface can easily lead to the misuse or underuse of the applications, by-passing important features that could increase user performance.

Furthermore, another issue found in complex applications covering a wide range of requirements is that each group of users takes advantage of a small part of the functionalities, but all the extra unused features still remain in the user interface, occupying screen space and conveying extra cognitive load to the user that is not required to perform the tasks.

These issues go beyond regular desktop applications, and get even worse for those applications designed for mobile devices, since the space available to present the user interface is greatly reduced. Thus, the functionalities found more common during the design process are usually the fastest to be carried out though the user interfaces. But, what if other users encounter problems finding some other functionality considered to be unimportant during the design? Should not the application support rearranging the user interface to support these unforeseen needs?

So far, the problems identified concern the complexity of the user interface and the heterogeneity of contexts of use. Nevertheless, another issue comes to play: understanding the user interface (one of the factors usability is considered to be composed of). Even for user interfaces with a reduced set of functionalities, the user can find it hard to understand how to carry out a task because the designers failed to match user’s mental model.

In all these situations, supportive user interfaces (SUIs) [4] can prove useful. We find that this kind of user interfaces are also closely related to adaptation, as considered in ISATINE framework [2], because SUIs are required to help in performing several stages of the adaptation process proposed in this framework.

SUPPORTIVE USER INTERFACES IN ADAPTATION

Adaptation can range from adaptability, where the user is in charge of performing the adaptation process, to adaptivity, where is the system the entity in charge of performing the adaptation process. Nevertheless, many intermediate configurations are possible, where different entities are responsible for the several stages required to carry out user interface adaptation.

Next, ISATINE framework is briefly discussed to illustrate how adaptation, either adaptability or adaptivity, or any other combination to reach adaptation, should be enriched with SUIs throughout the adaptation process stages.

ISATINE adaptation framework

ISATINE framework [2] is a specialization of Norman’s theory of action for adaptation, aiming at covering the whole adaptation cycle, going beyond most adaptation frameworks, mostly focused on the actual execution of the adaptation. Three entities are considered in this framework: the user (U), the interactive system (S), or any third party (T). Find below a brief explanation of the stages found in this adaptation process:

- Goals for user interface adaptation: any entity (U, S, or T) may be responsible for establishing and maintaining up-to-date a series of goals to ensure user interface adaptation.

- Initiative for adaptation: this stage is further refined into formulation for an adaptation request, detection of an adaptation need, and notification for an adaptation request, depending on their location.
- Specification of adaptation: this stage is further refined in specification by demonstration, by computation, or by definition, depending on their origin: respectively, U, S, or T.
- Application of adaptation: this stage specifies which entity will apply the adaptation specified in the previous stage. Since this adaptation is always applied on the UI, this UI should always provide some mechanism to support it.
- Transition with adaptation: this stage specifies which entity will ensure a smooth transition between the UI before and after adaptation. For instance, if S is responsible for this stage, it could provide some visualization techniques, which will visualize the steps, executed for the transition.
- INterpretation of adaptation: this stage specifies which entity will produce meaningful information in order to facilitate the understanding of the adaptation by other entities. Typically, when S performs some adaptation without explanation, U does not necessarily understand why this type of adaptation has been performed.
- Evaluation of adaptation: this stage specifies the entity responsible for evaluating the quality of the adaptation performed so that it will be possible to check whether or not the goals initially specified are met.

SUI in ISATINE

Supportive user interfaces could be thought for almost every stage in ISATINE framework. In this section some examples are provided to show how they could be used to help in the adaptation process in several stages. Some specific examples of SUI supporting ISATINE framework can be found in [3].

Specification of adaptation

In this stage there are two tasks in which the user could be supported. The first one is specifying the adaptation that the user would like to apply. This is already very common for adaptable or customizable user interfaces, where the user is supported in specifying what to change. It is also a common task in end-user programming for user interface adaptation. In this kind of task the user should be presented with a user interface to support the specification of the adaptations. Notice how this supportive user interface could be either part of the regular user interface or not.

The second task in this stage where the user can be supported is the selection of what adaptation to apply among a set of plausible adaptations. A user interface should be provided by the system to do this task. Very simple SUIs could be used to support the user, i.e. a simple selection list. However, much more complex SUIs could be imagined, i.e. providing previews for each adaptation selectable.

Application of adaptation

In this stage the adaptation selected should be applied to the user interface. If is the user the entity in charge, then a user interface must be provided to carry out this task. For instance, if the adaptation to be applied is for changing user interface elements layout, the user could be supported by providing a user interface where the user can move around the user interface.

Evaluation of adaptation

In this stage the system should assess how good an adaptation has been. If it is the user the entity in charge of performing this stage, then a user interface should be presented for the user to express his opinion. For instance, in [1] they present to the user a simple UI with different *smilies*, which represent how happy the user feels about the last adaptation.

Next, a discussion of SUIs in adaptation is included.

DISCUSSION AND OPEN QUESTIONS

The first thing to clarify is what we mean with supportive user interface. For us a SUI is a UI that exploits UI meta-model information to convey/receive information about the UI to/from the user, or provides a means to modify the structure, behavior or contents of the UI. Regarding the definition of SUI one question arises: are SUI a complement or an evolution of Mega-UI [5]?

SUI can be either part of the regular UI or not. Nevertheless, they should not escape general UI design principles and guidelines, although some extra ones should appear because of their supportive nature. We have plenty of design guidelines, interaction patterns, heuristics, design principles and standards, but how can be integrate all this plethora of knowledge in the design process, and more concretely in the design in the design process of SUI.

The design of SUIs for adaptation should pursue especially consistency, for the user to gain a common mental model for user interface adaptation tasks, as the user already has for the general task in a user interface.

Another open question is what the relation is between SUI and Intelligent User Interfaces (IUI). Therefore, we have to consider supportive vs. intelligent user interfaces. Will the *S* in SUI finally become “Semantic” to achieve Semantic User Interfaces. Has the evolution in the Web gone further beyond to reach the desktop to foster cooperative, semantic and ubiquitous desktop user interfaces?

SUIs require also the user of proper metaphors to prevent the user from becoming puzzled because of the usual overwhelming complexity of the underlying UI meta-model that SUI should manage.

Yet another open question is the evaluation of this kind of user interfaces. What criteria and metrics should be considered during the evaluation of SUIs? Is usability enough?

Still much understanding and general principles for SUI design are to be discovered. Adaptation capabilities are clearly a good domain to test this understanding and principles for SUI design, since as discussed in ISATINE framework, it requires of SUI for many of the adaptation stages to carry out a proper adaptation process.

To sum up, should we go one step further, and even coin the term Supportive User Interfaces Engineering (SUIE)? What is the relation SUIE has with Usability Engineering, Model-Based Development of User Interfaces or Model-Driven Development?

In this sense, we do believe ISATINE framework can provide a guide for the consideration of the specification, design, deployment and evaluation of SUIs.

ISATINE framework can help in providing SUI designers with a guide of what aspects should address the designer to create a SUI that: (i) effectively manipulates the user interface (therefore the specification of what is manipulated in the user interface should be carried out: Specification state in ISATINE), (ii) actually makes the required changes to the user interface (Execution stage in ISATINE), (iii) makes sure that the transition to the new version of the user interface produced by the SUI from the original one is smooth enough so the user does not get confused (Transition stage in ISATINE) or (iv) explains the user what changes were made (INterpretation stage in ISATINE). In our opinion, SUI designers could benefit from ISATINE guidelines for adaptation, but it should be probably refined to reflect the peculiarities of SUI.

ACKNOWLEDGMENTS

This work has been partially supported by grant PEII09-0054-9581 from the Junta de Comunidades de Castilla-La Mancha and also by the grant TIN2008-06596-C02-01 from the Spanish Government.

REFERENCES

1. Arhippainen, L., Rantakokko, T. and Tähti, M.: 2005, Navigation with an Adaptive Mobile Map-Application: User Experiences of Gesture- and Context-Sensitiveness. In: Proceedings of 2nd International Symposium on Ubiquitous Computing Systems, UCS’2005, Tokyo, November 8-9, 2004. Vol. 3598 of Lecture Notes in Computer Science. pp. 62–73, Springer, Berlin.

2. López Jaquero, V., Vanderdonckt, J., Montero, F., González, P. Towards an Extended Model of User Interface Adaptation: the ISATINE framework, Proc. of Engineering Interactive Systems 2007, EIS’2007 (Salamanca, 22-24 March 2007), M.B. Harning, J. Gulliksen (eds.), Springer-Verlag, Berlin, 2007. ISSN: 0302-9743

3. López-Jaquero, V., Montero, F. and Gonzalez, P. AB-HCI: an interface multi-agent system to support human-centred computing , IET Softw. 3, 14 (2009), DOI:10.1049/iet-sen:20070108.

4. Demeure, A., Lehmann, G., Petit, M. Calvary, G. Supportive User Interface description. http://www.supportiveui.org/cfp.html

5. Sottet, J-S., Calvary, G., Favre, J-M. and Coutaz, J. Megamodeling and Metamodel-Driven Engineering for Plastic User Interfaces: MEGA-UI. In Human-Centered Software Engineering, pp 173-200, Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (eds.), Springer Human-Computer Interaction Series. 2009.

A Supportive User Interface for Customization of Graphical-to-Vocal Adaptation

Fabio Paternò
CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
Fabio.Paterno@isti.cnr.it

Christian Sisti
CNR-ISTI, HIIS Laboratory
Via Moruzzi 1, 56124 Pisa, Italy
Christian.Sisti@isti.cnr.it

ABSTRACT
In this paper, we describe an approach to adapting graphical Web pages into vocal ones, and show how the approach is supported by a tool that allows the user to drive the adaptation results by customizing the adaptation parameters. The adaptation process exploits model-based user interface descriptions.

Keywords
Vocal Interfaces, Model-Based, Adaptation, Supportive User Interfaces, Accessibility.

INTRODUCTION

Vocal interfaces are important in a number of different contexts, such as for vision-impaired users or when the visual channel is busy (e.g, car driving) [7]. Design techniques in developing Vocal Interfaces has been widely studied [1] but little attention has been paid on how to adapt web pages for vocal browsing. Moreover, recognition of natural language is improving [2] and in future it will be possible to develop vocal interfaces able to recognize any user input.

We found that adaptation of graphical Web pages into vocal ones needs to be supplemented through Supportive User Interfaces (SUI), that enable the users to customize the adaptation. Indeed, a completely automatic transformation cannot provide good results in many case.

The adaptation process is based on the exploitation of MARIA [5], a recent model-based language, which allows designers to specify abstract and concrete user interface languages according to the CAMELEON Reference framework [3]. The customization tool has a Web interface allowing the user to drive the Vocal Interfaces generation.

In this workshop paper we firstly present the overall Model-Based Language Architecture, secondly we introduce the adaptation approach and lastly we show an example of application of the supportive interface for graphical-to-vocal adaptations, also showing how a parameter change can lead to different results in the final user interface.

MODEL-BASED INTERFACES in MULTI-DEVICE ENVIRONMENTS

MARIA is a model-based language, which allows designers to specify abstract and concrete user interface languages. Abstract User Interfaces (AUIs) are independent on the interaction modalities, while Concrete User Interfaces (CUIs) are dependent on the interaction resources of the target platforms but are independent of the implementation languages.

An AUI is composed by a number of *presentations*, a *data model* and a set of *external functions*. Moreover each presentation contains a number of user interface elements, called *interactors*, and a number of, so called, *interactor compositions.* Examples of interactor compositions are *grouping* and *relations* to group/relate different interactors. The interactors can be classified in terms of *editing*, *selection*, *output* and *control* and may have associated a number of *events handler*.

As already mentioned, the CUIs are dependent on the interaction resources of the target platform so, while in Desktop modality a presentation can be defined as a set of user interface elements perceivable at a given time, in the case of Vocal modality a presentation is defined as a set of dialogues between user and platform that can be identified as a logical unit (e.g. the communication necessary for a vocal form filling).

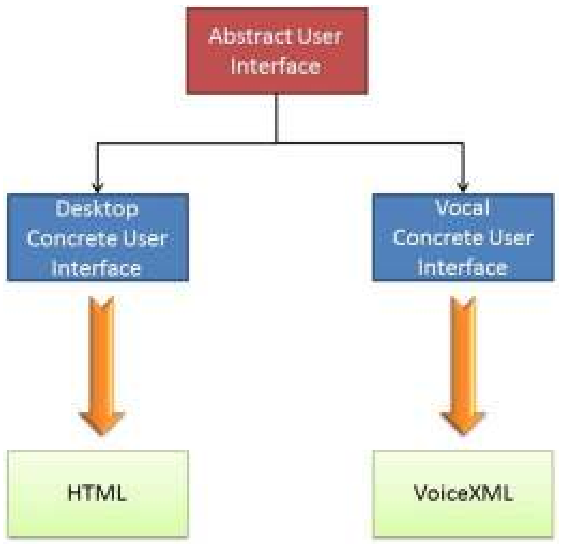

Abstract User Interface
Desktop Concrete User Interface
Vocal Concrete User Interface
HTML
VoiceXML

Figure 1. Some Possible Abstraction Levels

Figure 1 shows the relationship between AUI and CUIs limited to Desktop and Vocal target platform (some other target platforms available are Mobile, Multi-Touch and Multi-Modal). Figure 1 also represents some possible transformations that can be performed, such as the HTML generation from Desktop Logical

Descriptions (an instance of a Desktop CUI) and the VoiceXML generation from Vocal Logical Descriptions.

The aim of our work is to develop an adaptation process that take as input HTML pages, and generates corresponding VoiceXML (opportunely adapted for voice modality) documents. This is not a simple task and raises a large number of adaptation issues (such as the retrieving of the menu items for vocal interaction and the adaptation of images). In this context Supportive User Interfaces can provide useful support, in particular in the customization of the adaptation rules.

APPROACH

Our solution is based on an adaptation server that consists of three modules (see Figure 2):

- **Reverser:** parses the Web pages and builds up an equivalent Desktop Concrete Logical Description.
- **Adapter:** transforms the Desktop Concrete Logical Description into an adapted Vocal Concrete Logical Description.
- **Generator:** generates the VoiceXML taking in input the Vocal Concrete Logical Description.

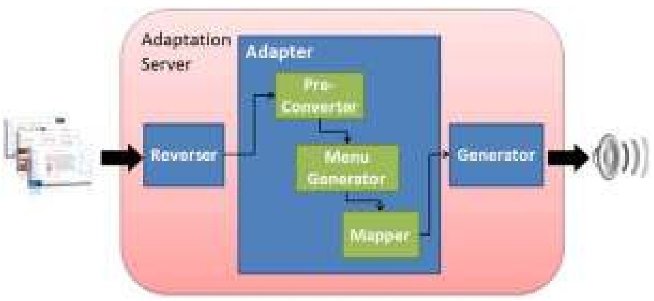


Figure 2. The Adaptation Server Architecture.

The reverser, taking into account the associated page style-sheet, transforms the HTML tag patterns into opportune Desktop CUI elements. This process enables the possibility to obtain a more semantic description. The adapter is subdivided into three sub-modules that are executed in pipeline:

1. **Pre-Converter:** removes the elements that cannot be rendered vocally (e.g., images without ALT tag) but also corrects possible inconsistences due to the reverse process (e.g., grouping containing only one interactor due to formatting purposes).
2. **Menu-Generator:** generally the vocal interfaces are navigated through lists of menus. This step aims to convert a Desktop Logical Description into a new one structured into a set of of menus/sub-menus hierarchically structured.
3. **Graphical-to-Vocal Mapper:** with this step each elements of the Desktop CUI is mapped into a (semantically equivalent) element of a new Vocal CUI.

The final implementation language is VoiceXML 2.0 [8], a standard language, supported by W3C, for the specification of Vocal Interfaces. The VoiceXML code generated by the transformation has been tested with the Voxeo Voice Browser [9] (suggested by W3C), and has passed the validation test integrated in it. More detail on the VoiceXML generation is provided in [4].

THE CUSTOMIZATION SUPPORT

The adaptation process is complex and the results depend on a number of factors, such as the structure of the Web pages in input and their conformance to the accessibility guidelines. In order to obtain better results we have designed a Supportive User Interface, which allows the user to customize the adaptation results.
The adaptation process can be driven setting a number of parameters. Such parameters can influence different states of the transformation process.
To adjust the *pre-conversion* step the following parameters are available:

- **Remove Whitespaces:** if enabled it removes the grouping that contains only whitespaces from the computation. This can happen due to graphical formatting purposes (e.g., list of "*&nbsp;*").
- **Min Image Width/Height:** images under these size limits (that not contains ALT attribute) are removed.
- **Min Grouping Threshold:** in the specification provided by the reverse engineering removing grouping operators when they contain little text (below the threshold) to synthesize.

To customize the *menu generator* step it is possible to set the following parameters:

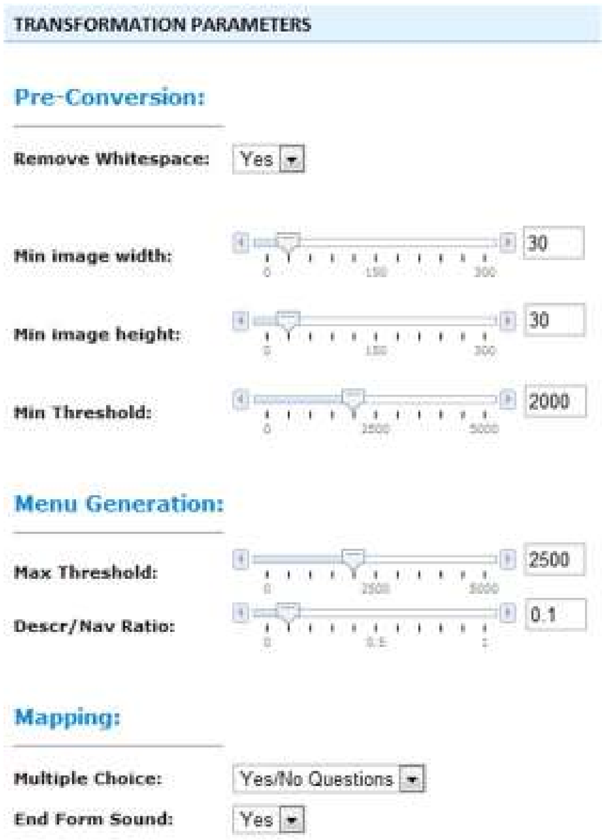
- **Max Grouping Threshold:** if the textual grouping content length is above the max threshold, then new menu items are created by splitting the original grouping.
- **Descr/Nav ratio:** to set the ratio between the description and navigator interactors in order to identify the groupings that contain a navigator bar.

Finally, to customize the *mapper* step, the parameters are:

- **Multiple Choice**: to set how the final vocal interface will perform the multiple choice. There are two solutions: *Yes/No Questions*, for every possible choice the platform will ask a Yes/No confirmation to the user; *Grammar Based*: the user can select more than one possible choice with one single sentence (listing the choices in sequence).

- **End Form Sound:** to decide if each vocal dialogue should terminate with a short sound.

Figure 3 and 4 show our Supportive User Interface that allows such parameterization. The left panel (shown in Figure 3) contains some modifiable parameters and their default's values.



TRANSFORMATION PARAMETERS

Pre-Conversion:

Remove Whitespace:	Yes
Min image width:	30
Min image height:	30
Min Threshold:	2000

Menu Generation:

Max Threshold:	2500
Descr/Nav Ratio:	0.1

Mapping:

Multiple Choice:	Yes/No Questions
End Form Sound:	Yes

Figure 3. Customization of the adapter.

The right panel (see figure below) shows the structure and the menu items of the generated vocal page. In this way the designer can decide whether to download the final vocal interface (as a zip file containing the VoiceXML documents) or change the transformation parameters in order to obtain a different structure.

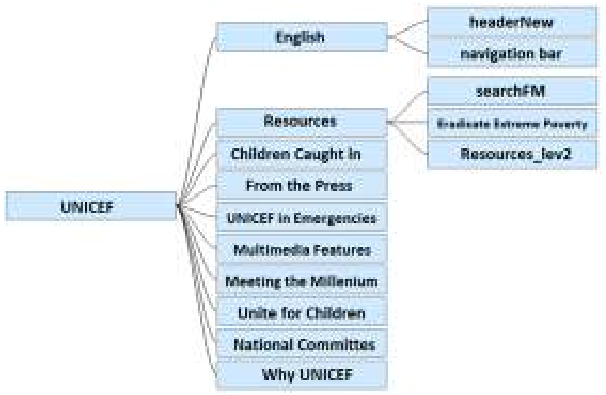


Figure 4. Application right panel: vocal menu structure.

EXAMPLE CONFIGURATION PARAMETER CHANGE

In this section we show an example of configuration parameter change, which affects the structure of the resulting user interface.

In particular, we consider Max_Threshold parameter, which defines the threshold in terms of text length to render vocally. If the length exceeds this limit the adaptation system splits the presentation content. If we set **max_threshold = 2500** then we obtain the structure in Figure 4.

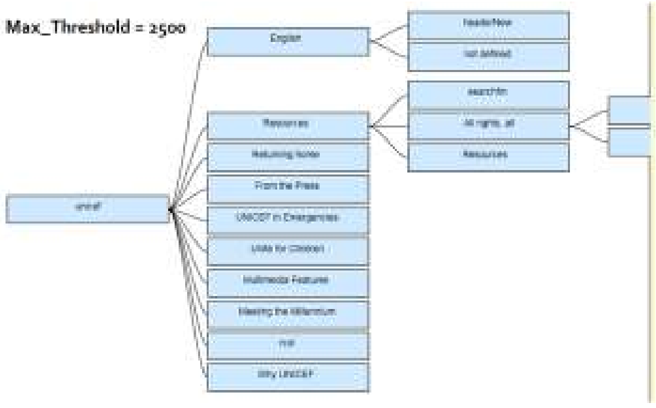
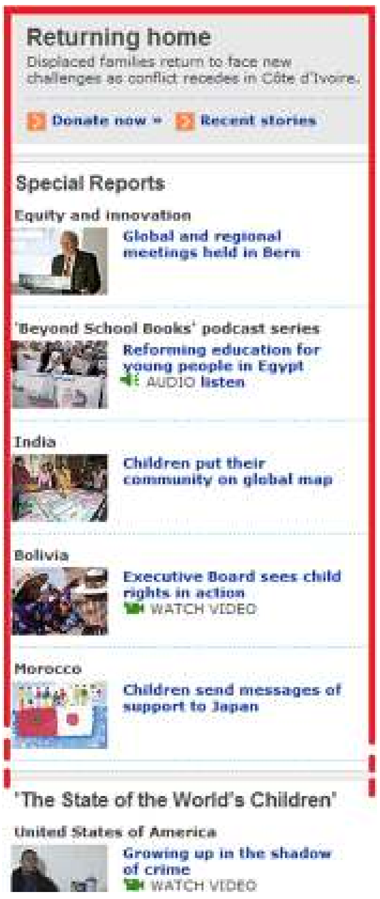


Figure 4. Initial parameter set.

Thus, the **Returning home** part (see Figure 5) will be rendered a single piece of information.



Returning home

Displaced families return to face new challenges as conflict recedes in Côte d'Ivoire.

Donate now » Recent stories

Special Reports

Equity and innovation

Global and regional meetings held in Bern

'Beyond School Books' podcast series

Reforming education for young people in Egypt

AUDIO listen

India

Children put their community on global map

Bolivia

Executive Board sees child rights in action

WATCH VIDEO

Morocco

Children send messages of support to Japan

'The State of the World's Children'

United States of America

Growing up in the shadow of crime

WATCH VIDEO

Figure 5. The considered content part.

If we change the parameter to **max_threshold = 700** we obtain the structure in Figure 6.

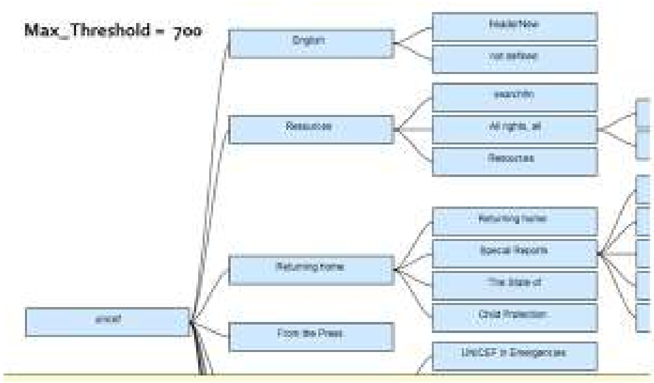


Figure 6. The resulting modified structure

We can note that the resulting structure has more sub-levels: the section **Returning home** is subdivided in multiple parts, highlighted by dashed lines in Figure 7, which can be further subdivided.

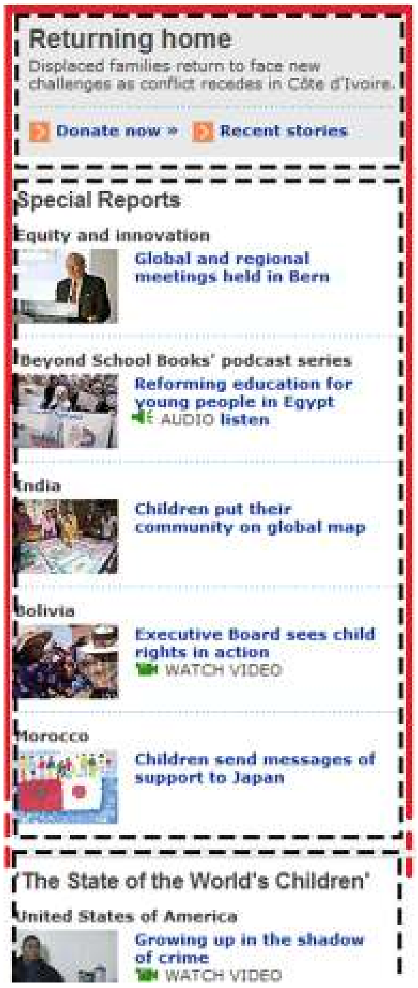

Returning home
Displaced families return to face new challenges as conflict recedes in Côte d'Ivoire.
Donate now » Recent stories
Special Reports
Equity and innovation
Global and regional meetings held in Bern
Beyond School Books' podcast series
Reforming education for young people in Egypt
AUDIO listen
India
Children put their community on global map
Bolivia
Executive Board sees child rights in action
WATCH VIDEO
Morocco
Children send messages of support to Japan
'The State of the World's Children'
United States of America
Growing up in the shadow of crime
WATCH VIDEO

Figure 7. How the content is further divided.

CONCLUSION

A Model-Based approach to supporting Graphical-to-Vocal Adaptation is introduced. A Supportive User Interface is then proposed (as Web Application) in order to help the user to manage the overall adaptation process.

We consider this tool as useful support to provide users with full control on the final results. Given the complexity of the existing Web content, we plan to add new features to both the adaptation rules and the customization interface, in order to have further flexible control on the adaptation results.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the Artemis EU SMARCOS and the ICT EU SERENOA projects.

REFERENCES

1. A., Edwards and I., Pitt.: Design of Speech-Based devices. Springer (2007).
2. A., Franz. and B., Milch.:Searching the web by Voice. In proceeding of the 19th international conference on Computational Linguistic - Volume 2, pp. 1-5, Stroudsburg, PA, USA. (2002).
3. Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, O., Marucci, L., Paternò, F.: The CAMELEON reference framework. CAMELEON project, Deliverable 1.1. (2002).
4. F., Paternò and C., Sisti.: Deriving Vocal Interfaces in Multi-device Authoring Environments. In Proceedings of the 10th International Conference on Web Engineering, pp. 204-217 (2010).
5. Paternò F., Santoro C., Spano L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact., 16(4). (2009).
6. UNICEF. http://www.unicef.org/.
7. Voice Browser Activity. http://www.w3.org/Voice/.
8. Voice extensible markup language (VoiceXML) version 2.0. http://www.w3.org/TR/2009/REC-voicexml20-20090303/7.
9. Voxeo Voice Browser. http://www.voxeo.com/.

Design and Implementation of Meta User Interfaces for Interaction in Smart Environments

Dirk Roscher, Grzegorz Lehmann, Marco Blumendorf, Sahin Albayrak
DAI-Labor, TU-Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
firstname.lastname@DAI-Labor.de

ABSTRACT

Interaction in smart environments encompasses multiple input and output devices, different modalities, and involves multiple applications. Each of these aspects is subject to changes and thus high adaptation requirements are posed on user interfaces in smart environments. One of the challenges in this context is the assuring of the usability of highly-adaptive user interfaces. In this paper, we describe the design and implementation of a Meta User Interface that enables the user to observe, understand, manage and control ubiquitous user interfaces. Our major contribution is a functional model and system architecture for Meta-User Interfaces for smart environments.

Keywords
Supportive UIs, meta-UI, smart environments

INTRODUCTION

Smart environments comprehend networks of (interaction) devices and sensors that influence the interaction between humans and computers. In contrast to the traditional usage of applications with one PC, the interaction in smart environments comprehends a dynamic set of multiple devices supporting different modalities and involves multiple applications and users. Based on an analysis of multimodal interaction in smart environments, the notion of ubiquitous user interfaces (UUIs) with five distinguished features has been defined in [1]:

1. Shapeability: Identifies the capability of a UI to provide multiple representations suitable for different contexts of use on a single interaction resource.
2. Distribution: Identifies the capability of a UI to present information simultaneously on multiple interaction resources, connected to different interaction devices.
3. Multimodality: Identifies the capability of the UI to support more than one modality.
4. Shareability: Denotes the capability of a UI to be used by more than one user (simultaneously or sequential) while sharing (partial) application data and (partial) interaction state.
5. Mergeability: Denotes the capability of a UI to be combined either partly or completely with another UI to create combined views and input possibilities.

These features enable UUIs to address the variable dimensions of smart environments (multiple devices, modalities, user, applications and situations). By addressing these challenges, UUIs become adaptive and can respond to dynamic alteration of one or more features at runtime. Such adaptations can be done either manually by the user or automatically by the runtime system. An important aspect in this sense is the transparency of system decisions and user control of the features. With respect to these needs, the term meta user interface (meta-UI) was established by Coutaz et al. [2] as a definition of "an interactive system whose set of functions is necessary and sufficient to control and evaluate the state of an interactive ambient space".

Meta-UIs have the potential to help the user in understanding and controlling the high variability within the interactive space. [3] presents a model-driven approach for developing self-explanatory UIs that make design decisions understandable to the user. In [4] a graphical representation of the system's state explains the interconnections between sensors and devices as well as their effects. These works show how the interaction in a highly adaptive interactive space can be improved when giving the user appropriate UI evaluation and control tools. However, there is yet no common understanding of the necessary features of meta-UIs for smart environments.

In the next section, we present an example UUI scenario, in which a meta-UI assists the user. In the section thereafter, based on the features of UUIs and the scenario, we describe necessary functionalities of a meta-UI for UUIs. Afterwards, we discuss the requirements for a runtime architecture for meta-UIs as well as for the actual applications. The section thereon illustrates our current implementation, addressing several of the identified challenges. Finally, we conclude the paper and denote some open research challenges.

INTERACTION IN A SMART ENVIRONMENT

The following scenario illustrates an example UUI and a possible usage of a meta-UI with the help of a calendar application utilized in a smart home environment. Thereby, we want to underline the necessity of control and evaluation capabilities that are required to analyze and configure the ubiquitous calendar application.

Dieter is living in a smart home, equipped with a broad range of networked devices and sensors. Every morning, when Dieter is in the kitchen, he asks his smart home to present him the calendar application with the appointments for today. (1) Dieter can control how the information is presented: if he utters the words “read out”, the appointments are presented via voice. Saying “show there” and pointing on the kitchen screen triggers the display of information on the screen. “Silence” disables all voice output. (2) When Dieter leaves the kitchen and walks around his smart home, the voice output follows him until all appointments are read out. Similarly, the displayed information also moves with him to the screens in his vicinity until he confirms to be done with his daily planning. (3) This behavior has been configured and trained by Dieter once after he installed his new calendar application. (4) Training took some effort though, and Dieter could continually monitor the system during the training process, while the system was giving valuable hints about why certain adaptations had been applied.

Sometimes Dieter needs to reschedule appointments to avoid conflicts. (5) To do so, he orders the system to change from voice or screen output to a presentation on the TV, synchronized with the display and controls of his smartphone. This allows him to interact and check details while keeping the overview on the big screen. Rescheduling appointments occasionally raises the need to contact colleagues and customers to agree on a different date or timeslot. (6) For this purpose, Dieter can configure the calendar application to set up video calls to the provided contact data while sharing the relevant calendar information with the called person. (7) Dieter can additionally select information from his notes application to share it. (8) He has the ability to store such a configuration and is able to reactivate the configuration whenever he wants.

EVALUATING AND CONTROLLING UUIs

The above scenario exemplifies UUIs with their five features (shapeability, distribution, multimodality, shareability and mergeability) and shows how the user influences each of these features at runtime. In the following, we describe the functionalities of a meta-UI in general and for all five features of UUIs in more detail.

General Features

According to the definition given in [2], a meta-UI provides evaluation and control features, which in our case allows to manage the adaptation of UIs in our example smart environment. The evaluation functionalities allow users to understand the behavior and current status of the interactive system, while the control features allow the user to influence and change the interactive system according to their needs.

Evaluation functionalities (e.g. (4) in our scenario) address the need of the user to always have access to information about the state of the system and enable the system to inform the user about any changes in the state of the interactive space. Changes do not only include automatic adaptations of the interactive system, but also cover manual adaptations where the user has to be informed as well especially when the manual adaptation does not provide the results expected by the user. Another very important information for the user in case of automatic adaptations is the reason why the adaptation happened. Information can thereby be conveyed implicitly by the look and feel of the UI [5] or be explicitly given to the user, which might be annoying in some cases though.

On the other hand, the control functionalities enable users to configure the interactive system according to their needs. That includes the possibility to configure the features independently on various levels of detail, the triggering of adaptations as well as the control of ongoing adaptations. For automatic adaptation, there is a need to configure the triggers that activate the adaptations, or to (de-)activate such adaptations at all.

The meta-UI has to support the user in the handling of the numerous situations and the possible configurations of the interactive system. Therefore the meta-UI has to provide capabilities to learn from the changes users’ made and to store configurations and reapply them when needed ((3) and (8) in scenario).

From our perspective, the meta-UI does not provide functionalities for end-user development as the user cannot create new functionality but “only” adapts and explores the interactive system based on existing functionality.

Shapeability

(5) shows how the user switches between the utilization of different devices and how this triggers the splitting of the UI to two devices. This requires the adaptation of the UI to the actual device features and the provisioning of different representations for the different utilized devices.

In terms of the evaluation of the shapeability feature, any adaptation of the graphical layout (e.g. rearrangements or reorientation of UI elements) should be made transparent for the user. For example, modern tablets and smartphones automatically change their screen orientation depending on how the user is holding them. Usually the orientation changes are animated so the user can follow and understand them. Another common shapeability feedback is a special beep tone indicating the currently configured volume for auditory UIs. Switching between different devices or device combinations, as in the scenario (5), requires even more advanced evaluation features. Users cannot follow the reshaping of the elements across devices and have to be aware of the changes between the different representations. This e.g. includes added or removed information because of more or less screen space.

One example for a more complex adaptation, which requires explicit access to information about the reason of the adaptation and means to control it, is the context-based GUI layouting functionality presented in [6]. The adaptation automatically resizes UI elements depending on the position of the user relative to the currently used display. Animations between different UI layouts are helpful, but not always sufficient to understand the adaptations. Thus, a meta-UI provides information about the position of the user currently detected by the system and the distance to the display. The user also has the possibility to turn the automatic adaptations off at any time.

Distribution
As shown in the scenario (2, 5), in a smart environment the user is able to use various interaction devices, between which the UI is distributed. Furthermore, the devices can also be changed dynamically by redistributing the UI. In terms of evaluation, the user has to be able to keep track of the distribution and may even want to explicitly inquire where a UI element has been distributed to. The user needs to know which devices are used for the output and also which devices can be used to enter data. In case of a redistribution of the UI the awareness of the changes can e.g. be transported by hints like "as you can see on the right display."

The control possibilities for the distribution of a UUI range from the application of distribution configurations preconfigured by the developer, to a very detailed shifting of single UI elements from one device (or even modality) to another performed by the user. Thereby it is also important for the user to know the devices available for a re-distribution and be informed about the potential effects; for example, if all tasks are still supported or if private information is visible to other people on a public display.

A more complex adaptation example for the distribution feature is the so called "follow me" mode illustrated in the scenario (2). Activation of the mode leads to an automatic redistribution of the UI to different devices based on changing situations. The interaction resources (IRs) available for the user are monitored and in case of changes (IRs becoming available or not) the UI elements are redistributed to a new calculated IR combination. Thereby, it is especially important to provide feedback to the user.

Multimodality
In the scenario the use case (1) illustrates how the user utilizes several modalities to interact with the application and seamlessly switches between them.

The user needs to be aware of the currently possible input modalities and ideally also the commands that are provided in each modality (e.g. currently active voice commands, which might be more than actually visible on the screen). A possible solution for implicitly transporting the usable input modalities in the graphical user interface is described in [5]. Control possibilities should at least include the turning on and off for certain modalities. Considering the numerous situations, it should also be possible to define certain situations with certain modality combinations.

Shareability
The capability to share parts of the UI or information with other users is illustrated in (7) within the scenario. This is also a basis for collaboration. While collaborating with other users, the user should be able to view and control which UI parts are shared with whom and with what rights (similar to e.g. social networking sites where it is possible for a user to view how others see the user's profile). Security and privacy thereby play a very important role for shareability. A meta-UI should make the user aware of (and in some cases even warn about) the risks of sharing security- or privacy-relevant UI parts.

Mergeability
Use case (6) shows how the user can merge different applications. This can include the transfer of information from one application to another as well as the combination of functionalities from different applications. The evaluation functionalities comprehend at least information about the current status of merged applications.

To control the merge of different applications, users need to know which applications or part of the applications can be combined with each other. Furthermore, the effects of the merge (e.g. enhanced functionality) also have to be made available for the user.

Based on the scenario analysis carried out in this section, in the next section, we derive requirements for the runtime infrastructure providing a meta-UI.

ARCHITECTURAL REQUIREMENTS
Besides some general requirements, the evaluation and control functionalities described in the previous section pose requirements on the UUIs and the runtime infrastructure in which the UUIs are deployed.

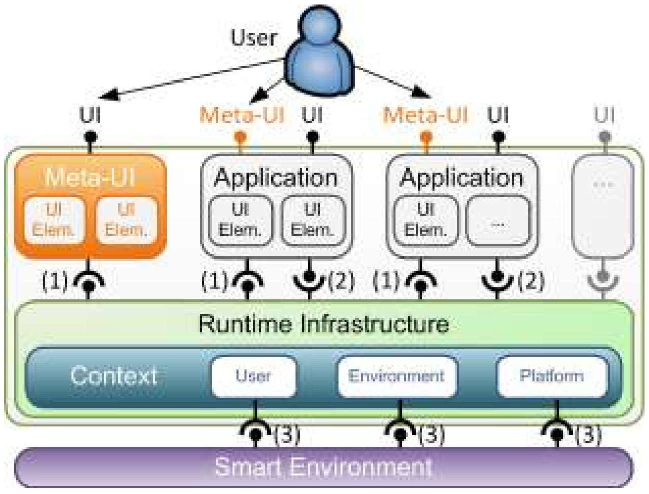


Figure 1: Meta-UI functionalities can be implemented either in a separate meta-UI application (orange box) or be part of applications. Control and evaluation interfaces of the runtime infrastructure (1), the applications (2) and the smart environment (3) are required to implement meta-UIs.

In general, a meta-UI for UUIs must be easily accessible and provide clear functionalities for evaluation and control of the UUIs in the environment. The meta-UI must hide the complexity of the interactive space (in terms of many devices, many modalities, many users, many applications, many and complex situations), while making it perceivable for the user.

As visualized in Figure 1, meta-UI functionalities can be realized twofold – either as a separate meta-UI application, or as part of the applications. In both cases, communication interfaces between the applications, the runtime infrastructure and the environment are needed.

To implement evaluation and control of each UUI feature, a meta-UI must be able to refer to every UI element affected by the respective feature. Thus, each application must provide information about its UI elements, their interaction capabilities and state ((2) in Figure 1). This information must be made accessible for the part of the meta-UI deployed within the runtime infrastructure ((1) in Figure 1). Similarly, meta-UIs require information about the environment, its users and the available platforms. The context information must be gathered at runtime from sensors and devices in the environment ((3) in Figure 1) and made accessible for the meta-UIs ((1) in Figure 1). By interpreting the information about the state of the applications and the context, meta-UIs can explain the current state of the interactive space.

As shown at various stages of the calendar application scenario (1, 5, 6, 7), meta-UI control functionalities require a detailed UUI configuration management. Through a meta-UI the UUI behavior can be configured manually (8) or automatically, e.g. by learning the user's preferences (3). Both pose a challenge for the runtime infrastructure handling different configurations and matching them with the current context situation.

A META-UI FOR SMART ENVIRONMENTS

Figure 1 shows a screenshot of our current implementation of a meta-UI. On the top in the center the user sees the modalities currently utilized for the application. At the bottom four menus enable the configuration of different UI features.

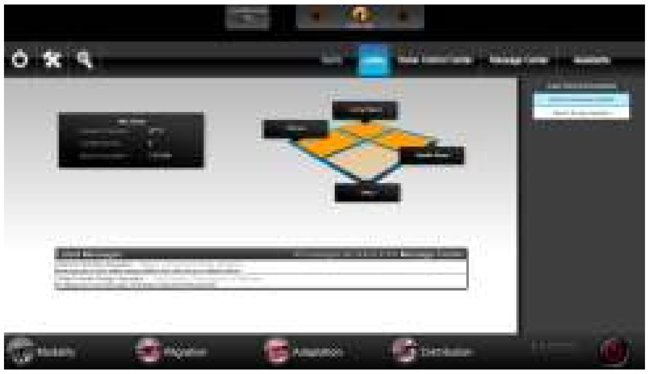


Figure 2: The Meta-UI surrounding the actual UI on the top and on the bottom.

The Migration menu provides possibilities to redistribute a UUI from one interaction resource to another, e.g. transfer the graphical UI to a screen better viewable from the users' current position. Through the Distribution menu the user can control the distribution on more fine grained levels by distributing selected parts of the UI among the available IRs. The user can also specify if the selected parts should be cloned or moved to the target IR. The selection of relevant UI elements can be done through an overlay display when activating the configuration possibility. The Modality configuration menu provides possibilities to configure the utilized modalities within the interaction. This allows users to e.g. switch off audio output if it is currently disturbing the user. Through the Adaptation menu the user controls more complex automatic adaptation functions (e.g. (de-)activates the follow me mode explained above).

In the future we plan to add the possibility to store and retrieve configurations. We also intend to implement the evaluation and control of mergeability and shareability.

CONLUSION

Meta-UIs are one of the available instruments for handling the variability of smart environments from the user's perspective. We have given an overview of general features Meta-UIs should include as well as of possible evaluation and control functionalities for UUIs. But to realize a well-established Meta-UI for UUIs like the traditional desktop metaphor for single PCs requires to solve many open challenges.

One open issue is to determine the concrete set of needed evaluation and configuration possibilities. Extensive user studies need to be done to solve this. Thereby question like the clustering and grouping of Meta-UI functionality has to be answered including possible different versions of Meta-UIs for e.g. users acting in a known or unknown environment (this e.g. poses additional requirements on the identification of interaction devices).

There are also several challenges for the configuration of the features by the user. One example are automatic adaptations that uses artificial intelligence. In cases of inappropriate behavior, the user should also influence and configure such algorithms. Another issue is the determination of the reason why a user reconfigures the system (context selection). Furthermore, the meta-UI is also a user interface the user is interacting with. So the same requirements for evaluation and configuration holds true for itself.

REFERENCES

1. Blumendorf, M. Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces. Dissertation, Technische Universität Berlin, 2009.
2. Coutaz, J. Meta-user interfaces for ambient spaces. Proceedings of TAMODIA'06, 2006, Springer, 1-15.
3. García Frey, A., Calvary, G. and Dupuy-Chesa, S. Xplain: an editor for building self-explanatory user interfaces by model-driven engineering. Proceedings of EICS '10, 2010, ACM, 42-46.
4. Vermeulen, J., Slenders, J., Luyten, K., and Coninx, K. I bet you look good on the wall: Making the invisible computer visible. Proceedings of AmI '09, Springer.
5. Weingarten, F., Blumendorf, M. and Albayrak, S. Conveying multimodal interaction possibilities through the use of appearances. 2010.
6. Schwartze, V. Adaptive user interfaces for smart environments. Proceedings of ICPS'10 Doctoral Colloquium, 2010.

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

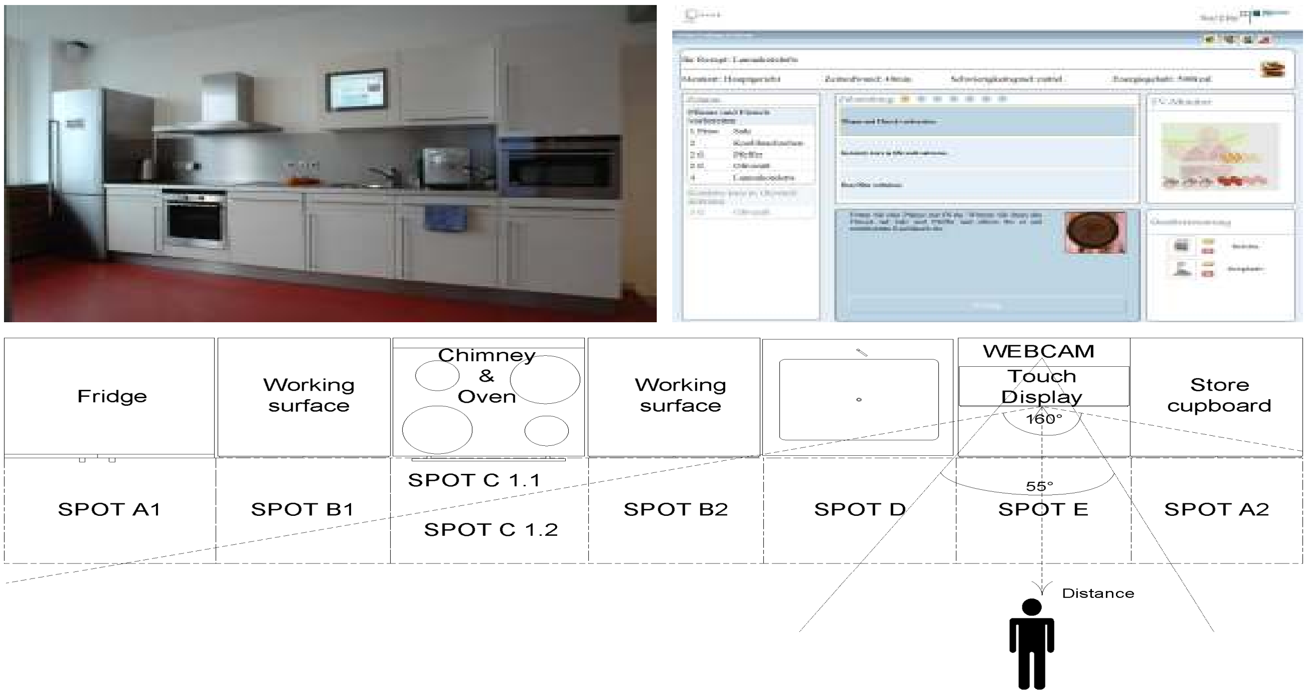


Figure 1: The kitchen with the cooking assistant running on a touch screen (top-left), the main screen of the cooking assistant (top-right), and the location spots defined by the context model (bottom).

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.

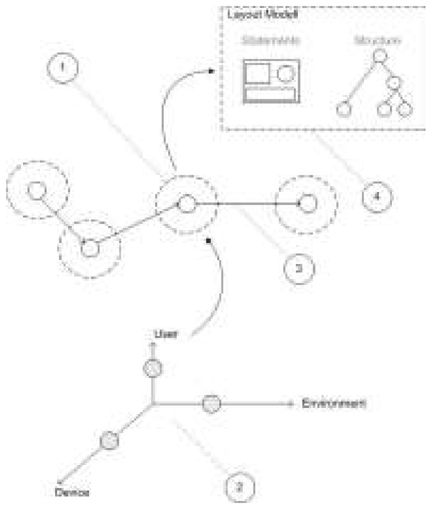


Figure 2: Example graph of layout model adaptations

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.

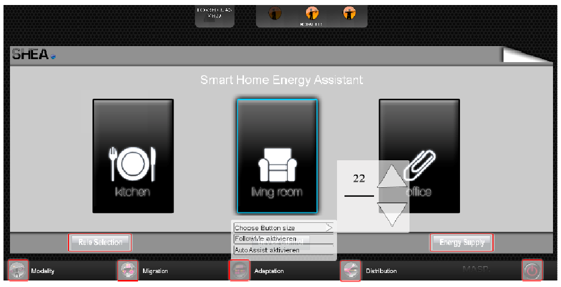


Figure 3: Supportive user interface screenshot

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.

A classification for Supportive User Interfaces derived from Collaborative User Interfaces

Carsten Wirth
TU-Berlin, DAI-Labor
Sekretariat TEL 14
Ernst-Reuter-Platz 7
10587 Berlin
carsten.wirth@dai-labor.de

Sahin Albayrak
TU-Berlin, DAI-Labor
Sekretariat TEL 14
Ernst-Reuter-Platz 7
10587 Berlin
sahin.albayrak@dai-labor.de

ABSTRACT

In this paper, we describe that the concept of supportive user interface is overlapping with aspects, which can be found in collaborative user interfaces and how this can help to classify and design supportive user interfaces accordingly.

Keywords
support, collaboration, classification

INTRODUCTION

The growing complexity of today's and future ubiquitous systems which is driven by innovative enabling technologies, new interaction techniques and concepts as well as context-of-use dynamics is raising new challenges regarding end user support. The User Interface (UI) has to be well designed by hiding complexity from the user but still providing easy access to all functions. It has to provide customization regards to user’s personal needs but also has to adapt automatically to the context of use for reducing user disturbance while performing her tasks. These requirements are partly conflicting so that the resulting system behavior can lead to user confusion. To solve this problem, the system has to enable the user to understand what is happening and how the application behavior can be controlled as desired.

A promising approach towards extended user support is seen in equipping the UI with corresponding supporting functionality, which is developed and/or provided simultaneously with the primarily functions. These Supportive User Interfaces (SUI) can come in manifold ways which makes comparisons and discussions difficult because there is missing a classification as well as a clear definition of SUI by now. In this paper we propose a classification which is derived from Collaborative User Interfaces (COUI) since as we will show many parallels between SUI and COUI can be drawn.

In the following section the concepts of SUI and COUI are presented along with examples of their manifestations. This enables to elaborate several parallels of the both UI types in the section thereafter. As a result a classification for SUI is proposed and implications on design aspects for SUI are described afterwards. The paper will finish with a conclusion and outlook.

RELATED WORK

In this section an overview of the UI types SUI and COUI and their manifold manifestations is given. So that the parallels between SUI and COUI can be elaborated on the common understanding in the next section.

Supportive User Interfaces

The concept of SUI is to provide the user with support within complex systems such as ubiquitous systems by means of making the user able to understand what is happening in the system and how the system can be controlled as desired with the numerous interaction possibilities provided. The SUI can come in manifold ways like self-explanatory user interfaces [5], process driven user-guidance environments [10], extended device control support [9], guidance for different modalities [7], support by utilizing contextual awareness [1], and Meta-UI, which can control and evaluate the states of the underlying system [2] and therefore can enable supportive functionality, and assistance with visualization of system behavior [13] amongst others.

Collaborative User Interfaces

Collaborative User Interfaces are part of collaborative environments and applications and are establishing a human to human collaboration regarding the three aspects communication, coordination and cooperation, which is also known as 3C-Model [4]. COUI can be found in diverse application functional classes [4]; e.g. Message Systems, Multi-User Editors, Group Decision Support Systems, Electronic Meeting Rooms, Computer Conferencing, Intelligent Agents, Workflow Management Systems and more. Depending on the purpose COUI are supporting each of the 3C differently [12].

PARALLELS OF SUI AND COUI

At first glance SUI and COUI seem to have few similarities based on their purpose. The purpose of the SUI is to help the user to understand what is happening and give a better control while the COUI focuses on supporting the user while performing shared tasks with other users.. To show the similarities of both a few examples are described subsequently. Thereafter a conclusion for the presented examples will be drawn in the following section.

Adaptation of a Workspace

A shared workspace is a common tool in Collaborative Environments (CE) [8][6] but team members normally have different preferences, different experiences, and often different training thus making adaptations necessary [11]. If a team member is changing the workspace layout in a way which is affecting all of the team (e.g., removing an important tool) the change has to be communicated and explained to be excepted by the team or at least the team has to be made aware of the change if a more hierarchical role concept is used. Likewise in a self adapting system (SAS) which is controlling a workspace environment adaptations of the workspace (removing a tool because of resolution changes) have to be communicated and explained to the user.

Simultaneous changes by the user and the system

Typically the work in CE takes place on some kind of shared business objects [11] which demands coordination of activities for conflict prevention or concurrency control to resolve conflicts between participants simultaneous operations [4]. In a SAS the user is sharing interface objects with the system. In example if the system decides to optimize the content of a toolbar at the same time the user is customizing it, this leads to a conflicting state. Either the user can get her privileges to change the toolbar revoked on short-term by the system to prevent conflicts or the system has to resolve emerging conflicts with a suitable solution. A simple one could be to overrule the users changes. Both the SUI and the COUI have to provide the appropriate coordination and concurrency control mechanism to minimize user confusion and disturbance along with suitable application control.

Application Tutoring

In CE colleagues may serve as tutors for inexperienced colleagues by guiding the first steps with tools provided by the environment (e.g., mouse traces can be followed, questions can be asked and are answered by others via chat etc.) This cooperation towards a goal with a common interest (in this case the same skill level for optimal working results) can be transferred to SAS or SUI respectively. The system and the user are sharing the common interest that the user can operate the application at best and therefore has to provide a SUI which enables cooperation towards this goal between the user and the system. To achieve this goal the SUI should be able to act as tutor for the user.

IMPLICATIONS FOR SUI FROM COUI

As shown by the examples in the section above system behavior triggered by an agent (whether that agent is automation or another human) establishes the same requirements upon the user support. Furthermore the aspects of communication, coordination and cooperation (3C), which are used to characterize collaborative applications can be found in the concept of SUI, with the difference that for SUI the user is collaborating with the system instead of a human.

For collaboration environments different classifications exist. In the context of SUI the 3C Model proposed by Teufel et al. [12] can be utilized to classify SUI respectively by weighting the support of each of the 3C within the system separately. The system can be classified by placing it in a triangle where each corner represents one of these properties as shown in Figure 1 (exemplary illustrated for [9][2][5] and a fictive Automation Level Configurator which allows the user to adjust the automation level of adaptations with guidance to find the optimal personal configuration).

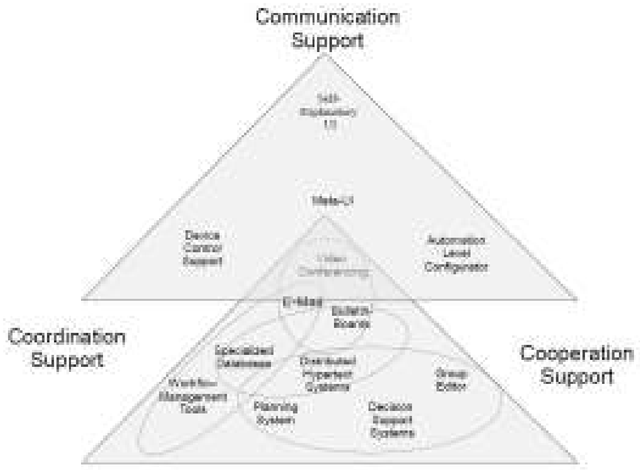


Figure 1: exemplary SUI 3C Classification based on[12]

The advantage of this classification is that both COUI and SUI become comparable. Furthermore, this can help to find design issues in SUIs. In a smart home for example the steering of activities of multiple users which may depend on shared device resources can be supported by SUI functionality with the goal to optimize daily routines and to avoid resource conflicts. This SUI with the focus on Device Control Support can inherit aspects and mechanisms of Workflow Management Systems because resource allocation and scheduling are fundamental issues of them [3].

Another benefit from considering collaborative aspects in SUI while designing interfaces is that parts of the UIs supportive functionality can be replaced later on by real collaborative functions if desired. Humans still tend to trust humans more then machines especially when life or money is involved. The configuration interface of an automated heating regulation system in a smart home for example can be either explained by the system itself or the user seeks the guidance of a human supervisor by switching to the collaborative mode. A fundamental issue of SUI amongst others therefore should be to support the user to get support, whether this support can be realized by the system itself, another system or other users.

CONCLUSION & OUTLOOK

In this paper the parallels between SUI and COUI have been shown; both share the aspects of communication, coordination and cooperation and are establishing the same requirements on the user support. Furthermore SUI can be classified with the help of the 3C Model likewise COUI. This classification can help to identify and to focus on design issues for SUI by considering related COUI implementations.

One can assume that a quality level of SUI could be how close the system is behaving in comparison to a real user within a similar collaborative environment. The specification of quality levels has to follow the clear specification of SUI and is therefore a interesting topic for future research.

REFERENCES

1. Bahr, G.; Balaban, C.; Milanova, M. & Choe, H. Stephanidis, C. (Ed.) Nonverbally Smart User Interfaces: Postural and Facial Expression Data in Human Computer Interaction *Universal Access in Human-Computer Interaction. Ambient Interaction*, Springer Berlin / Heidelberg, 2007, 4555, 740-749
2. Coutaz, J. Meta-user interfaces for ambient spaces. *Proceedings of TAMODIA'06*, 2006, Springer, 1-15.
3. Delias, P.; Doulamis, A.; Doulamis, N.; Matsatsinis, N.; , Optimizing Resource Conflicts in Workflow Management Systems, *Knowledge and Data Engineering, IEEE Transactions on* , vol.23, no.3, pp.417-432, March 2011
4. Ellis, C. A.; Gibbs, S. J. & Rein, G. Groupware: some issues and experiences *Commun. ACM*, ACM, 1991, 34, 39-58
5. Garc'ia Frey, A.; Calvary, G. & Dupuy-Chesa, S. Xplain: an editor for building self-explanatory user interfaces by model-driven engineering *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems*, ACM, 2010, 41-46
6. Greenberg, S. & Marwood, D. Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface, ACM Press, 1994, 207-217
7. Komatani, K.; Ueno, S.; Kawahara, T. & Okuno, H. G. Flexible guidance generation using user model in spoken dialogue systems *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, Association for Computational Linguistics, 2003, 256-263*
8. Lauwers, J. C. & Lantz, K. A. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 1990, 303-311
9. Seifried, T.; Haller, M.; Scott, S. D.; Perteneder, F.; Rendl, C.; Sakamoto, D. & Inami, M. CRISTAL: a collaborative home media and device controller based on a multi-touch display *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM, 2009, 33-40
10. Sliski, T. J.; Billmers, M. P.; Clarke, L. A. & Osterweil, L. J. An architecture for flexible, evolvable process-driven user-guidance environments *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, ACM, 2001, 33-43
11. Teege, G. Users as Composers: Parts and Features as a Basis for Tailorability in CSCW Systems *Computer Supported Cooperative Work (CSCW)*, Springer Netherlands, 2000, 9, 101-122
12. Teufel, S.; Sauter, C. & Mühlherr, T. Computerunterstützung für die Gruppenarbeit Addison-Wesley, 1995
13. Wachsmuth, S.; Wrede, S. & Hanheide, M. Coordinating interactive vision behaviors for cognitive assistance *Computer Vision and Image Understanding*, 2007, 108, 135 – 149

1st International Workshop on Supportive User Interfaces (SUI 2011)

3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems

Pisa, Italy - June 13, 2011 - http://www.supportiveui.org/

Organizers: Alexandre Demeure, Grzegorz Lehmann, Mathieu Petit, Gaëlle Calvary

Workshop Motivation and Goals

Enabling technologies make it possible to create more and more complex systems in terms of functional core, new interaction techniques and context-of-use dynamics. The users require a better understanding and control of their applications. This workshop focuses on human-computer interaction and more specifically on the engineering of user interfaces to foster intelligibility and control. In a broader context this workshop aims to identify and classify the supportive UIs that may enhance the interaction (e.g., by rendering the workflow in e-government applications or making it possible to the end-user to see the available platforms in the surrounding and redistribute the UIs him/herself).

SUI Goals from the User's Perspective [2]

- Customization and Personalization
- Appropriation
- End-user Empowerment
- Education
- Privacy and Auditability
- Comprehensive Behavior and Trust

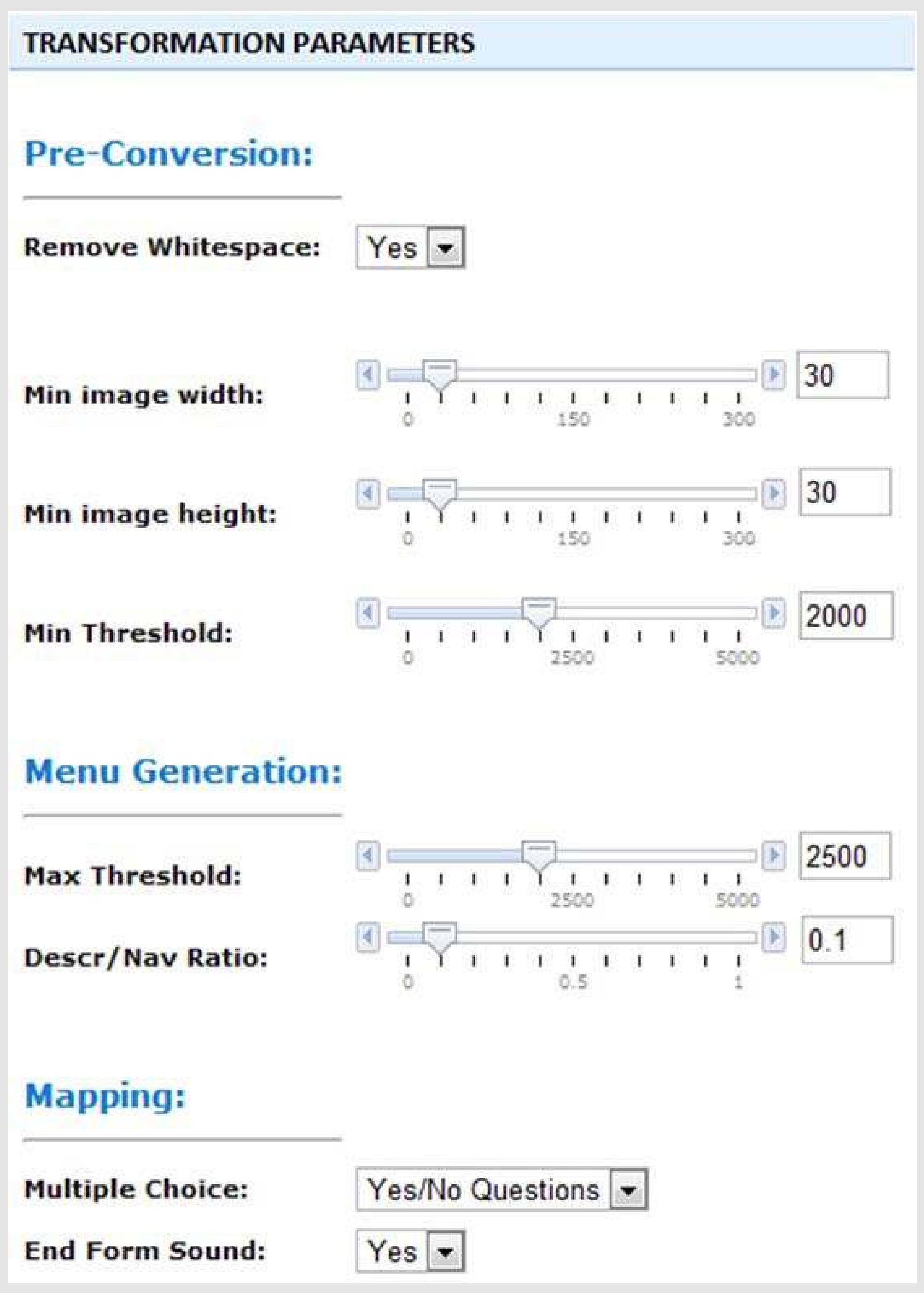
SUI Approaches

- Focused on the user [1, 2]
- Feature centric [6]
- Adaptation centric [4, 5, 7]

SUI Taxonomies

Two classifications based on:
- Collaborative UIs [8]
- Self-Explanatory UIs [3]

Some Examples Presented



TRANSFORMATION PARAMETERS

Pre-Conversion:

Remove Whitespace:	Yes
Min image width:	30
Min image height:	30
Min Threshold:	2000

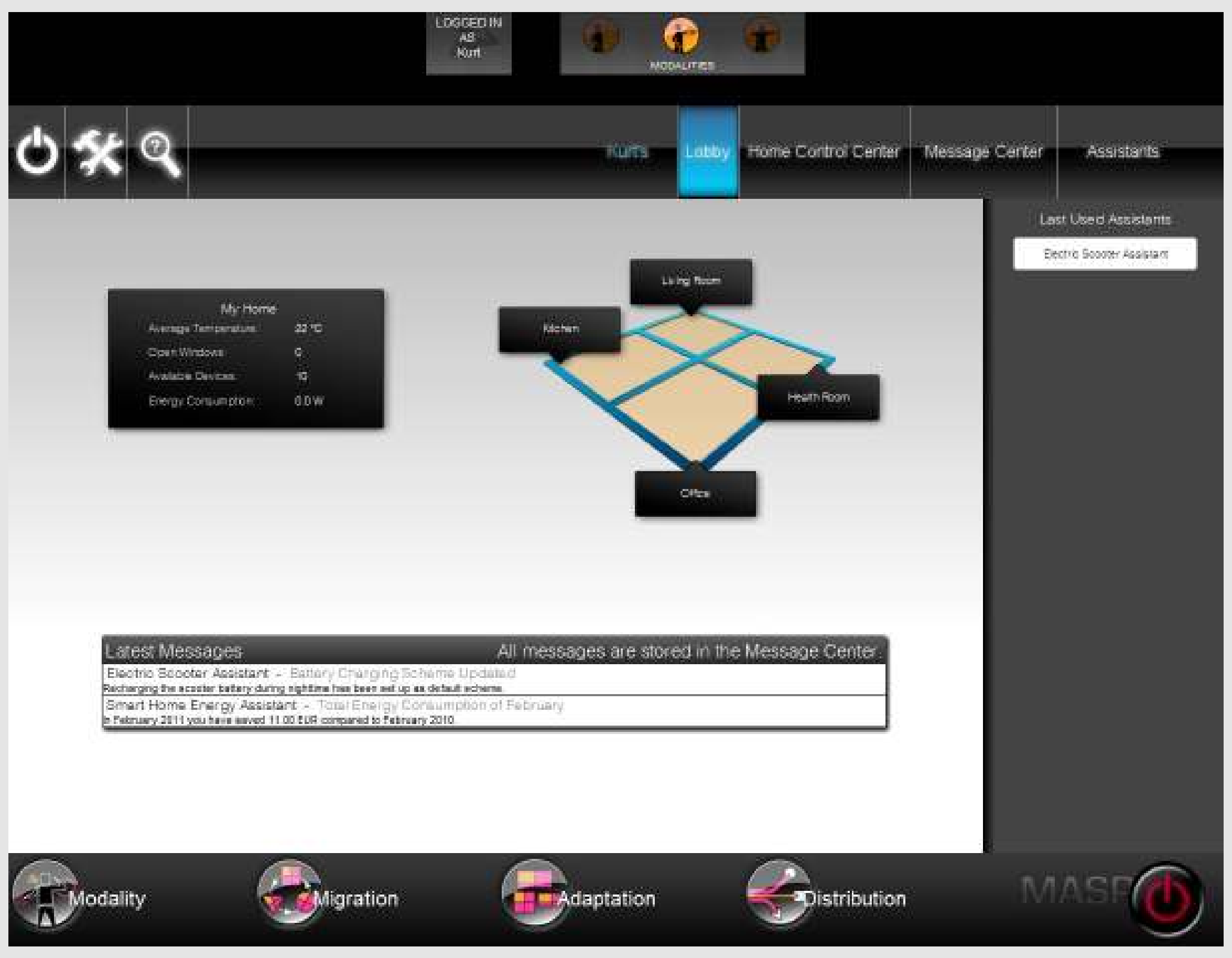
Menu Generation:

Max Threshold:	2500
Descr/Nav Ratio:	0.1

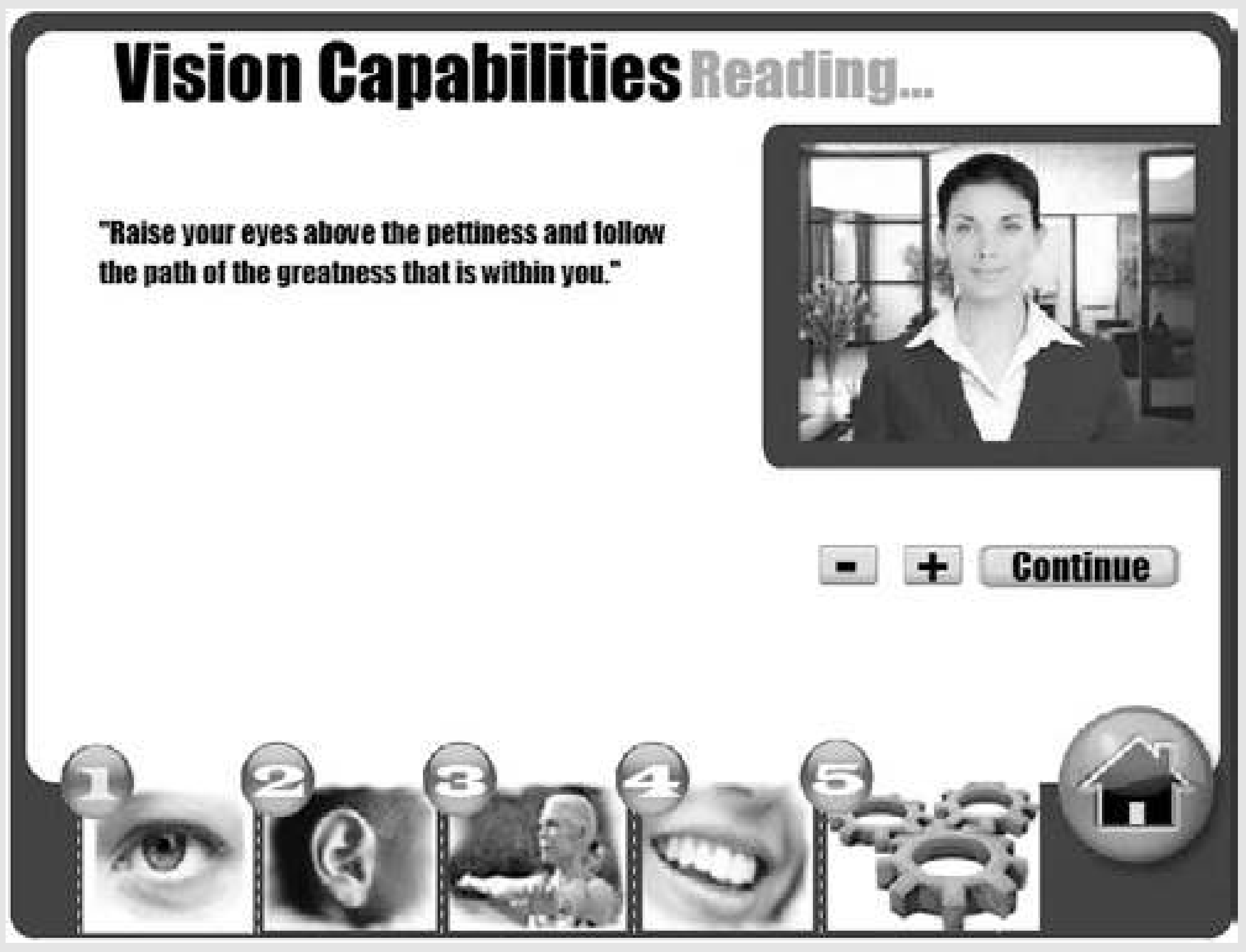
Mapping:

Multiple Choice:	Yes/No Questions
End Form Sound:	Yes

[5]

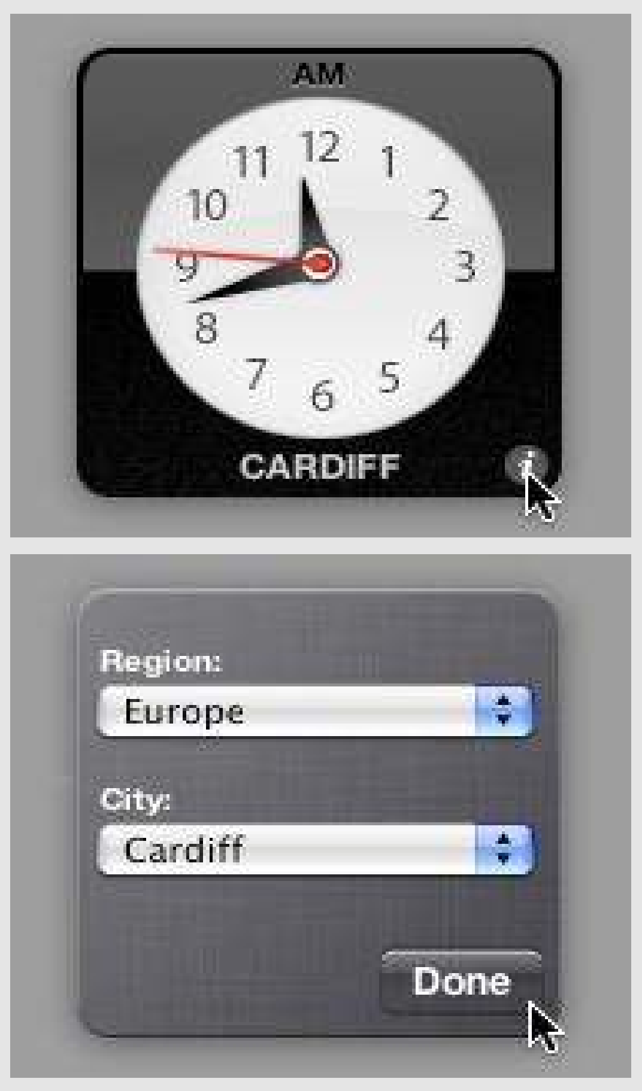


[6]

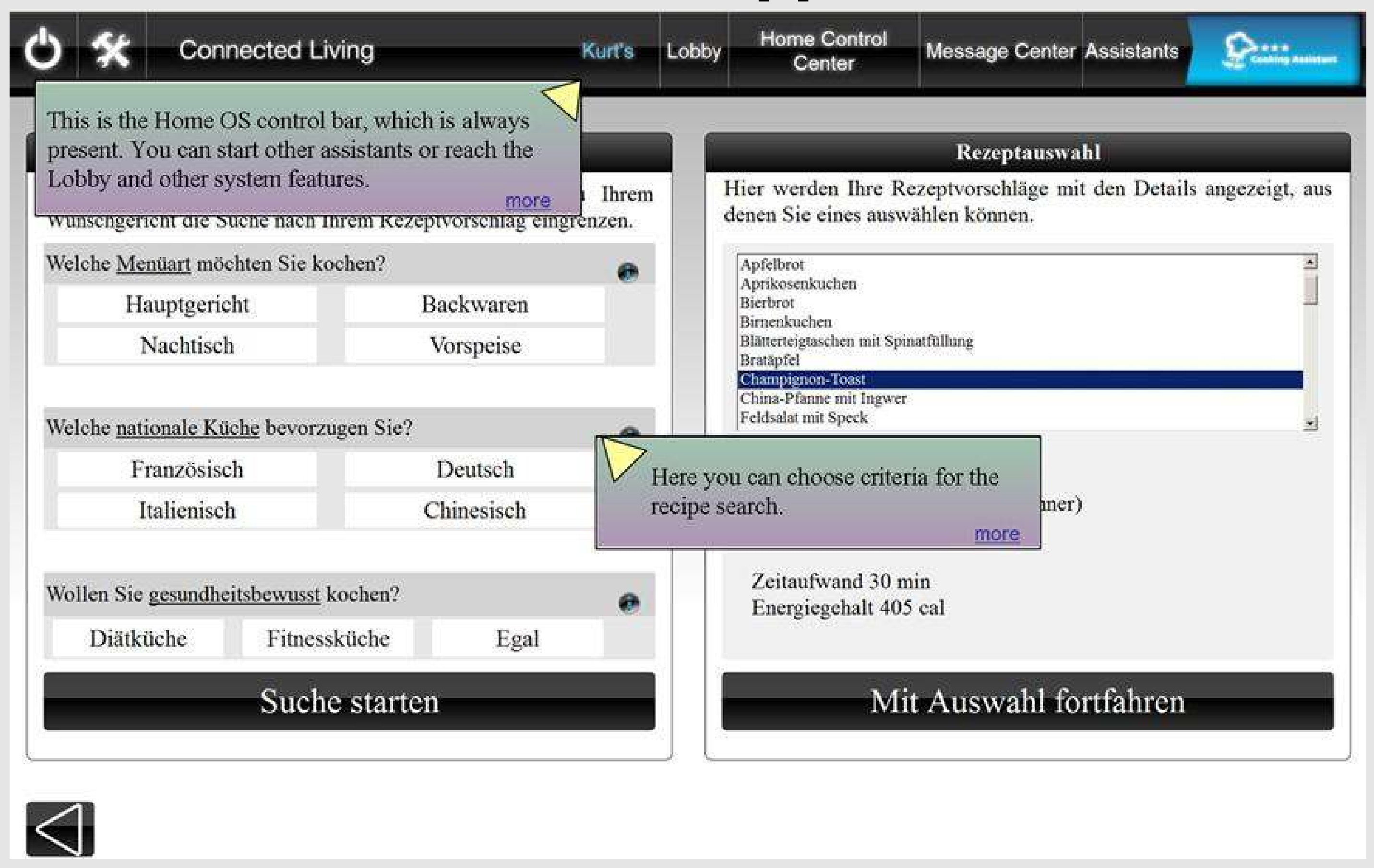

Vision Capabilities Reading...
"Raise your eyes above the pettiness and follow the path of the greatness that is within you."
-
+
Continue
1
2
3
4
5

[1]

[2]



[2]



Connected Living | Kurt's | Lobby | Home Control Center | Message Center | Assistants

This is the Home OS control bar, which is always present. You can start other assistants or reach the Lobby and other system features. more

Ihrem Wunschgericht die Suche nach Ihrem Rezeptvorschlag eingrenzen.

Welche Menüart möchten Sie kochen?

Hauptgericht	Backwaren
Nachtisch	Vorspeise

Welche nationale Küche bevorzugen Sie?

Französisch	Deutsch
Italienisch	Chinesisch

Here you can choose criteria for the recipe search. more

Wollen Sie gesundheitsbewusst kochen?

Diätküche	Fitnessküche	Egal

Suche starten

Rezeptauswahl

Hier werden Ihre Rezeptvorschläge mit den Details angezeigt, aus denen Sie eines auswählen können.

Apfelbrot
Aprikosenkuchen
Bierbrot
Birnenkuchen
Blätterteigtaschen mit Spinatfüllung
Bratäpfel
Champignon-Toast
China-Pfanne mit Ingwer
Feldsalat mit Speck

ner)

Zeitaufwand 30 min
Energiegehalt 405 cal

Mit Auswahl fortfahren

[3]

Agreed Definition of Supportive UIs

A supportive user interface (SUI) exchanges information about an interactive system with the user, and/or enables its modification, with the goal of improving the effectiveness and quality of the user's interaction with that system.

Research Agenda

- Elicit the dimensions of supportive UIs through a taxonomy that would cover both the abstraction and presentation of supportive UIs
- Discuss the properties supportive UIs should convey
- Explore how to integrate SUIs into development processes
- …

Presented Papers

1. *Building Supportive Multimodal User Interfaces* - José Coelho, Carlos Duarte
2. *Opening the Box - Meta-level Interfaces Needs and Solutions* - Alan Dix
3. *A Classification of Self-Explanatory User Interfaces* - Maximilian Kern, Marco Blumendorf, Sahin Albayrak
4. *Supportive User Interfaces in Adaptation* - Víctor López-Jaquero, Francisco Montero
5. *A Supportive User Interface for Customization of Graphical-to-Vocal Adaptation* - Fabio Paternò, Christian Sisti
6. *Design and Implementation of Meta User Interfaces for Interaction in Smart Environments* - Dirk Roscher, Grzegorz Lehmann, Marco Blumendorf, Sahin Albayrak
7. *The end-user vs. adaptive user interfaces* - Veit Schwartze, Frank Trollmann, Sahin Albayrak
8. *A classification for Supportive User Interfaces derived from Collaborative User Interfaces* - Carsten Wirth, Sahin Albayrak