Adaptive user interface assistance in smart environments

Maximilian Kern and **Frank Trollmann** and **Marco Blumendorf** and **Sahin Albayrak** [1]

Abstract. More than half of the products, which appear to be malfunctioning are often brought back to stores by the customers although they are in full working order. The reason for this is either the customers' lack of understanding of how to use such products or the unmanageable complexity of the products. This fact claims for intelligent user interfaces which adapt to the user at runtime. Such systems also require new kinds of help systems, which also need to be adaptive. In this paper, the work in progress of a user interface assistance approach in smart environments is presented. This approach aims at the utilization of model-based development in order to build self-explanatory systems which are able to guide and help the user dynamically. We take benefit of the self-describing nature of models and generate help for navigation within user interfaces. In the future, we will extend this approach in order to use the same semantics of the model which we consider for help generation in order to offer the user an alternative goal-driven interaction pattern.

1 INTRODUCTION

Software assistants provide information on the usage of a software system to the user. Their purpose is manifold. Software assistants support the user to orientate herself within a user interface and provide her explanations regarding available options, reasons for disabled options and instructions for applying certain actions.
The increasing complexity of today's devices requires intelligent user interfaces which adapt to the user at runtime. Such systems require new techniques for integrating help systems. Hence, the help systems also need to be adaptive. Adaptation can happen either on context information through changes made by the user or within dynamic environments where ad-hoc ensembles are formed. A premise for help systems to adapt is to have a description, more specifically, a model of the software system which should be guided and a description of the system state at runtime.

We developed a user interface assistance approach, the MASP Guide, which was integrated into the MASP (Multi Access Service Platform), a model-based framework for user interface generation. As we focussed on model definitions for UI generation, we now aim at the utilization of the same models for explaining the user how the system works. The approach is capable of generating explanations for the usage of the system, that are based on various models with additional annotations. Following features are currently covered:

- explain the user which tasks are available at one moment.
- tell the user how to fulfil an appropriate task.
- describe which interaction devices are available and how to apply them.

[1] Technische Universität Berlin, Germany, email: firstname.lastname@dai-labor.de

First an overview of the state of the art in assistant systems is given in the next section. After this, in section 3, we define the problem which our approach addresses. In section 4, the requirements for solving the problem are declared. Then, previous work is presented in section 5, on which our approach is based. Subsequently, the current state of our approach is described in section 6. Finally we conclude our approach and discuss future work.

2 STATE OF THE ART

The earliest approaches for software assistants emerged around 1966 with the HELP system developed under the Genie project [10][11]. 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 improvement enables recent assistants being capable of understanding, interpreting and speaking human language, capturing and considering context information of the user and learn from the user by observing her interaction.
Rich et al. [9] [8] introduced the DiamondHelp system which establishes a collaborative dialogue between the user and a system based on the system task model. The screen is split horizontally, where in the upper half the dialogue and in the lower area the user interface of the system is displayed. Thus, the user 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..
[4] introduced a hybrid approach, 'The Companions', which is able to incorporate knowledge retrieved from a social network or news site into a local rdf-based knowledge base. In order to give the user the impression of talking with 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. In [12] 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.
In [1] 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 according user interface elements are highlighted in order to provide user interface guidance.
By comparing these approaches to our approach introduced in this paper, many similarities can be recognized. One aspect, which is connecting these approaches among each other, is the challenge of utiliz-

ing knowledge contained in databases, ontologies or the Internet respectively by offering alternative ways of interaction and assistance, which pay more attention to the human nature. These approaches go beyond the WIMP-based[2] user interfaces, which use classical control elements as a enforcedly replacement for more intuitive user-centric interaction paradigms.

3 THE PROBLEM

In conventional user interface systems, each system state is well defined and can be already foreseen at design-time. Hence it is trivial to integrate static help for each system state. Intelligent user interfaces nowadays are highly adaptive and try to adjust themselves to the user upon context information which is gathered from sensors observing the environment, from extracted behaviour patterns and available interaction devices. The system needs to be able to offer help within any state that can be reached, although the concrete states are not predictable at design-time. Because of the resulting volatile dynamic of intelligent user interfaces, help functions and their integration within such environments became an increasingly challenging issue. The system needs to be capable of generating help which is at least as dynamic as its surrounding runtime environment. The requirements necessary to cope with this problem are discussed in the following section.

4 REQUIREMENTS

An essential premise for adaptive help is a comprehensive description of the system not only at design-time, but also at runtime. An application needs to be able to introspectively access and understand arbitrary system states at runtime, e.g. to explain required or optional tasks at the current user interface mask or explain the interaction with the displayed control elements.
Also, the adaption itself needs to be explained to the user. By introspection of the application's logic, reasons for inactivated control elements can be explained to the user and which prerequisites for their activation can be given. For instance, a search button will be activated by entering a key word or the user needs to change her location close to the device in order to activate the shown control elements for specific security-related devices such as a mixer.
Adhoc ensembles such as discovering new devices, which are available at the current location of the user, need to provide information about how to use them. As new devices are integrated into the user interface, this information needs to be dynamically weaved into the help which is available at the current step of the adapting user interface. For multimodal systems, available interaction devices should also be considered, e.g. if a microphone is available, the generated help should explain alternative interaction patterns such as using speech in order to control the user interface.
Conventional help documentations include clippings of the user interface to provide the user a link to the real user interface off the help documentation. For intelligent user interfaces it can be difficult for the user to identify those clippings after they are adapted, because their appearance is unpredictable. In order to provide meaningful help within adaptive systems, it is important to include help within the user interface itself at the current runtime state and offering direct manipulation of the user interface. Thus, by highlighting according areas of the user interface the user is able to apply given hints directly.

[2] Windows, Icons, Mouse, Pointer

By linking artefacts of the system to concepts of ontologies, multiple relations, which are not taken into account by the application designer, can be found and concluded to higher-level information. By applying ontology matching algorithms [5], the content presented to the user can be augmented by retrieving additional information from the internet or in case of a recipe search application, ingredients not available currently can be substituted by other appropriate ingredients. Furthermore, word analysis can be incorporated into adaptive help. Synonyms of the initiated vocabulary can be used in order to adjust the wording of the help description to the user's age or preference automatically. As the degree of adaptation is inexhaustible, the possibilities for help generation of such systems is following.

5 PRIMARILY WORK

Within the SerCHO (Service Centric HOme) project, the MASP was developed. The MASP is a model-based approach to generate user interfaces for several platforms. It incorporates well-known model based approaches such as CTT-task notation [7] and different abstraction layers such as task level, abstract user interface (AUI), concrete user interface (CUI) and final user interface (FUI) [3][6]. Each of the used models describes different aspects of the user interface, e.g. the context, the interface elements (with different abstractions), services and mappings between the models to mention only some of them. Furthermore, each model conforms to a meta-meta-model in order to separate model artefacts describing design-time- and run time-aspects and thus, allowing the models to execute themselves [2] and to adapt to any situation. Since those models describe themselves in a comprehensive way, we are convinced that they are a well-suited basis to generate meaningful support to the user. In this regard, the models can also be comprehended as an ontology which describes the system.
We evaluated the MASP by generating a virtual cook application, which offers functionalities for finding recipes with specified criteria, specifying the number of persons, generating a shopping list and prepare the chosen recipe. For each functionality a separate user interface mask is presented to the user, where she can specify desired options and run appropriate tasks. In order to improve usability and offer a more effective, intuitive and flexible interaction pattern, we extended the MASP by a user support function, which is providing help for navigation to the user. In the following section we describe this approach.

6 THE APPROACH

For the first version of our approach for generating help and guiding the user, we initially defined the requirements to be fulfilled. The MASP Guide should be capable of responding to help requests from the user in naturally spoken language. The user should be informed about available tasks at the current user interface mask. Furthermore, the user should receive detailed instructions on how to fulfil the available tasks, i.e. which information is needed, where she needs to specify them and through which modality she can enter these information. In order to attract the attention of the user to the area of the user interface which the system is currently mentioning, the appropriate area should be highlighted.
As a next step, we identified artefacts of the models, which are useful for fulfilling our requirements and information that needs to be defined additionally. In order to highlight elements as mentioned earlier, we gain layout information such as position and dimension of interface elements from the layout model. For identifying the used

interface elements (e.g. checkbox, radio button, button, etc.), the user interface model is considered. By checking the type of the available interaction task and its content, keywords which can be spoken to select a certain option are retrieved. Thus, we are able to determine which modality is available. The content (domain elements) of the current user interface is derived from the domain model. In order to

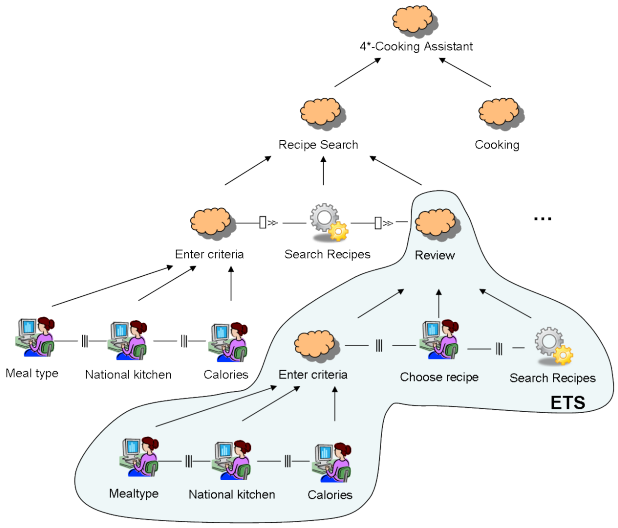

Figure 1. Excerpt of the task model of the virtual cook application.

explain a task, its task model is examined. An example for the task model is given in figure 1. Unfortunately, the information included within the task model is not sufficient for generating naturally spoken text. Therefore, we annotated descriptions to each interaction task and abstract task of the task model as depicted in table 1. Application

Task	Annotated description
Recipe Search	search for recipes
Enter criteria	define criteria for the recipe search
Meal type	specify the desired meal type
National kitchen	specify the desired national kitchen
Calories	define parameters for health awareness
Choose recipe	choose a recipe from the list of found recipes

Table 1. Annotated task descriptions.

tasks are executed by the system. At the current development phase, they are not yet considered. Since the task explanations are generated dynamically according to the enabled task set (ETS), the slices of the task description may theoretically be assembled in an arbitrary order and thus have to follow a distinct way of formulation. Additionally, the descriptions are used to identify the information requested by the user. We annotated global words for concatenating the slices, words for the beginning of the explanations, verbs for expressing the modality (e.g. speak and checking) as well as starting words for questions. Some of these definitions can be seen in table 2. Using these definitions and the task descriptions, as given in table 1, the virtual cooking assistant could formulate the sentence ”You can specify the desired meal or specify the desired national kitchen or define parameters for health awareness or choose a recipe from the list of found recipes.”, describing the users’ current options in the situation as depicted in figure 1.

The MASP guide application itself is a standalone application within the MASP with privileges for accessing any model artefact

Description	Words
Global question	What can I do here?
Detail question	How can I ...?
Explain global start	Here you can ...
Detailed explanation start	You can ...
Concatenation word	or

Table 2. Global word definitions.

of applications running within the same MASP instance. As a consequence, the MASP Guide has its own task model, which is presented in figure 2. Initially the MASP guide is starting with the application task ‘Collect description’ in order to collect the information of current active applications and their enabled tasks. After collecting the information mentioned above, an interaction task ‘Request help’, which is waiting for an input from the user, is started. As soon as suited input is received, the assembling application task is started, which is building up the explanation by collecting the first occurring annotated task description from the top task of the ETS. For instance, as for the review task the task description below the abstract task ‘Enter criteria’ is not yet considered. They are considered as soon as the user is requesting details for the task ‘Enter criteria’.

The abstract task ‘Assemble & Explain’ is iterative in order to man-

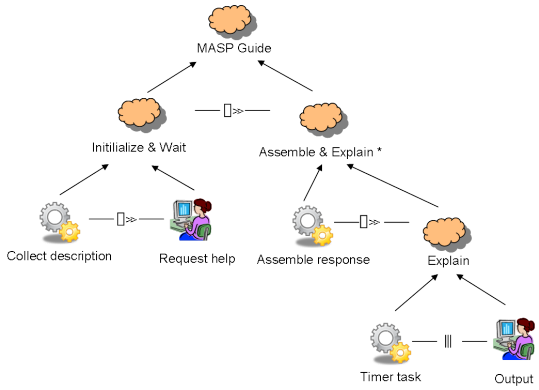

Figure 2. Task model of the MASP Guide

age the highlighting of the described area of the user interface. After building up the first/next portion of the response text, it is delegated to the explain task. The explain task is an abstract task with two child tasks, a timer task and an output task. The output task is an interaction task, which is responsible for the voice output of the previously assembled explanation and for highlighting the appropriate area of the user interface. Since there is no specification about how long an interaction task lasts, the timer task is waiting for the time span of the voice output. In the following, the pseudo code of the task ‘Assemble & Explain’ for the ETS highlighted in figure 1 is provided, assuming, the user asks “What can I do here?”:

1. Task ‘Assemble & Explain’ and its subtask ‘Assemble response’ are activated.
2. Retrieve ETS from active application. Collect next unvisited task, if there is no unvisited task, clear state and stop the task ‘Assemble & Explain’.
3. Retrieve box within the user interface, e.g. which contains control elements for entering criteria (linked to abstract task ‘Enter criteria’, see also left box within figure 3).
4. If first time executed, append word ‘Explain global start’ to the response, if not append the ‘Concatenation word’ to the response.

5. Retrieve description of the current task, e.g. ‘Enter criteria’, and append it to the response. Mark task as visited.
6. Delegate box and response to abstract task ‘Explain’.
7. Abstract task ‘Explain’ and its subtasks ‘Timer task’ and ‘Output’ are activated.
8. ‘Timer task’ is executed for estimated time of voice output of the response. Abstract task ‘Explain’ ends after time lapses.
9. In parallel, the task ‘Output’ is emitting the response via voice output and pulsates the corresponding box.
10. Start over from 1.

With all mentioned annotations and model artefacts, the MASP Guide is able to explain any MASP application. Within an arbitrary state of the virtual cook application, the user can ask “What can I do here?”. For instance, in the review task (see figure 3) the system replies “Here you can define the criteria for the recipe search or choose a recipe from the list of found recipes.”. For retrieving detailed information about an interaction task, the user can ask e.g. “How can I define criteria for the recipe search?” and the system accordingly responds “You can specify criteria by choosing the meal type, the national kitchen or the desired calories.”. To get information about how to specify e.g. criteria for meal type the user can ask “How can I choose the meal type?”, the system responds “You can choose a meal type by saying or checking main menu, dessert, appetizer or pastries.”

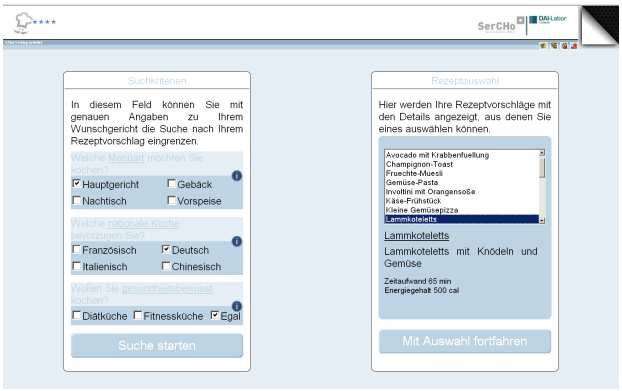

Figure 3. Recipe result mask

7 CONCLUSION AND OUTLOOK

Our approach is based on the models of the virtual cook application augmented with additional information. The MASP Guide is able to generate appropriate answers to a small set of questions. The approach is supposed to be extended by incorporating further models in future. Advanced assistants provide knowledge or call system functionalities, which are requested by the user through a dialog with the system. Assistants are a promising approach for providing the user a goal-oriented way of interaction. The user is enabled to focus on his goal rather than on single system functions. While humans are thinking on a conceptual level when interacting with a system, e.g. ‘Printing a document’, computers offer this abstract task as a sequence of basic operations ($open\ a\ document \rightarrow choose\ menu\ `file' \rightarrow choose\ `print...' \rightarrow set\ up\ parameters \rightarrow click\ button\ `print'$). Assistants work as a mediator between human and systems in order to find a link between the goal, which the user has in mind and system functionalities available within a system.

In the research area of model-based software engineering, task models help a software designer to focus on a conceptual level rather than on underlying low-level functionalities. Thus, task models can be adapted in order not only to generate help, but also understand semantics of interactions on a more abstract level. Due to the nature of the CTT notation with tasks and their temporal relations, an essential ascertainment is the chance to predict any possible interaction path the user can reach within the application as described in [1]. Furthermore, it can be deduced which domain objects, more specifically which user input, is necessary for satisfying a user interaction task and run the related application task.
Based on this gnosis, we intend to provide an auto-complete function as a new interaction pattern for the MASP. With this auto-complete function the user will be able to execute multiple tasks by specifying one single sentence. For instance, the user says “I want to cook a German meal for 2 persons!” and the MASP is running the according application tasks with specified parameters, leading the user to the recipe list of German meals and skip the user interface mask, where the amount is calculated upon the number of persons.
Finally, we want to analyze wordings (find synonyms, word relations) in order to give the user more freedom in her expression. This function may become a great support to the user and may also be seen as a first step towards a better understanding of what the user wants to accomplish.

REFERENCES

[1] Matthias Bezold, ‘Describing user interactions in adaptive interactive systems’, in *UMAP*, pp. 150–161, (2009).
[2] Marco Blumendorf, Grzegorz Lehmann, Sebastian Feuerstack, and Sahin Albayrak, ‘Executable models for human-computer interaction’, in *Interactive Systems. Design, Specification, and Verification: 15th International Workshop, DSV-IS 2008 Kingston, Canada, July 16-18, 2008 Revised Papers*, eds., T. C. Nicholas Graham and Philippe Palanque, pp. 238–251, Berlin, Heidelberg, (2008). Springer-Verlag.
[3] Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Laurent Bouillon, and Jean Vanderdonckt, ‘A unifying reference framework for multi-target user interfaces’, *Interacting with Computers*, **15**(3), 289–308, (2003).
[4] A. Dingli, Y. Wilks, R. Catizone, and W. Cheng, ‘The companions: Hybrid-world approach’, in *International Joint Conference on Artificial Intelligence (IJCAI)*, (2009).
[5] Jérôme Euzenat and Pavel Shvaiko, *Ontology Matching*, Springer-Verlag, Berlin-Heidelberg, 2007.
[6] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillonand Murielle Florins, and Daniela Trevisan, ‘Usixml: A user interface description language for context-sensitive user interfaces’, in *ACM AVI 2004 Workshop: UIXML*, pp. 55–62, (2004).
[7] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi, ‘Concurtasktrees: A diagrammatic notation for specifying task models’, in *Proceedings of Interact’97*, ed., Gitte Lindgaard Steve Howard, Judy Hammond, Part of the INTERACT: IFIP International Conference On Human-Computer Interaction conference series. Chapman and Hall, (1997).
[8] Charles Rich and Candace Sidner, ‘Diamondhelp: A generic collaborative task guidance system’, *AI Magazine*, **28**(2), (2007).
[9] Charles Rich, Candace Sidner, Neal Lesh, Andrew Garland, Shane Booth, and Markus Chimani, ‘Diamondhelp: a new interaction design for networked home appliances’, *Personal Ubiquitous Comput.*, **10**(2-3), 187–190, (2006).
[10] Roger Roberts, ‘C s carr help—an on line computer information system’, Technical Report P-4, Project Genie Document, (January 1966).
[11] Roger Roberts, ‘Help: a question answering system’, in *AFIPS ’70 (Fall): Proceedings of the November 17-19, 1970, fall joint computer conference*, pp. 547–554, New York, NY, USA, (1970). ACM.
[12] Manfred Stede and David Schlangen, ‘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).