

Special Challenges for Models and Patterns in Smart Environments

Peter Forbrig, Christian Märtin, and Michael Zaki

University of Rostock, Department of Computer Science,
Albert Einstein Str. 21,
18055 Rostock, Germany

{peter.forbrig, christian.maertin, michael.zaki}@uni-rostock.de

Abstract. Smart environments aim at inferring the intention of the user and based on that information, they offer optimal assistance for the users while performing their tasks. This paper discusses the role of supportive user interfaces for explicitly interacting with the environment in such cases where implicit interactions of the users fail or the users want to get informed about the state of the environment. It will be shown by small examples how patterns help to specify the intended support with implicit and explicit interactions. A notation for presentation patterns will be introduced that allows users dynamically to change the presentation style. It will be discussed how extended task models can be combined with presentation patterns and how this information can be used in supportive user interfaces on mobile devices.

Keywords: Smart Environment, model-based design, pattern, supportive user interface, task migratability, task pattern, presentation patter.

1 Introduction

During the last few decades a lot of work has been accomplished by different research teams to study prototypes of environments of assisting users performing their daily life tasks. This research was often focused on elderly people but sometimes also focuses on children (e.g. [3] and [16]).

Our paper is based on research within our graduate school MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Applications). The experimental basis is a smart meeting room. The room is equipped with a lot of sensors, projectors and cinema screens (see Fig. 1.).

Bayesian algorithms are informed by sensors and try to infer next possible actions of the users. Based on that information, convenient assistance has to be provided.

“This creates complex and unpredictable interactive computing environments that are hard to understand. Users thus have difficulties to build up their mental model of such interactive systems. To address this issue users need possibilities to evaluate the state of these systems and to adapt them according to their needs.” [13]

Meta-UIs are mentioned by the authors of paper [13] as a solution for this problem. This means that users are able to configure user interfaces that can be visible (explicit interaction) and invisible (implicit interaction via sensors).



Fig. 1. Smart meeting room

As a result of the SUI 2011 workshop participants agreed on the following more specific and precise definition for this kind of user interfaces:

“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.“ [7].

The most important aspect of this definition is the fact that the user interface should be adaptable in order to give the user the opportunity to interact with the system in a more appropriate way according to the specific encountered context of use.

This idea of a “Meta-User Interface” approach for controlling and evaluating interactive ambient spaces was also suggested by [2].

We will focus our discussion in this paper on the role of supportive user interfaces in smart meeting rooms. The paper is structured in such a way that first, existing approaches for supportive user interfaces in ubiquitous environments are discussed. Afterwards, models are studied that help to develop supportive user interfaces. Additionally, it will be discussed how specific patterns might help to assemble models and are helpful during run-time.

2 Models and Supportive User Interfaces

Within our graduate school MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Applications) we have the opportunity to study research questions for supporting users while performing their tasks in a smart meeting room. We already mentioned that one approach for such environments is the usage of Bayesian networks. These networks describe possible activities of users. Algorithms are available that

infer next possible actions. Based on this information it is possible to provide convenient assistance to the user.

Unfortunately, it is not easy to create such a Bayesian network. Additionally, such networks have to be trained. In this way a lot of meetings with similar goals and participants have to be observed. Even when this was possible it is not easy to provide the user with information about the current state of the environment. Roscher et al. discuss in [13] a functional model and system architecture for Meta-User Interfaces. Such interfaces allow users to control devices in the smart environment in an explicit way. In this way it is possible to “manually overrule” the decision of the environment.

“The Migration menu provides possibilities to redistribute a UUI (ubiquitous user interface) 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.” [13].

For ubiquitous user interfaces the five features shapeability, distribution, multimodality, shareability and mergeability are specified and presented in [14]. These results are originally from [2].

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 results reflect in a wonderful way necessary technical properties of user interfaces in given ubiquitous environment. They also underline the necessity of having explicit interactions in smart environments.

In our discussion we will focus on two main aspects:

1. What kind of models can help to specify user interfaces for smart environments in detail?
2. What kind of patterns can support the modeling of smart environments?

3 Models for Smart Environments

In conjunction with modeling efforts for smart environments the collaborative task modeling language (CTML) was developed in our group. This language consists of models specifying the activities of stakeholders and the whole team by task models. Additionally there are models for devices and the room as well. Details can be found in the PhD thesis of Maik Wurdel [21].

In this paper we will concentrate on the models that help us to generate user interfaces for explicit interactions. Task models are most important for this aspect. CTML uses task tress in the notation of CTT [12] extended by constrains in an OCL-like style. This notation was extended in [22] by new task types that are recalled in Fig. 2.





Type	Symbol	Description
User Output task		The user is providing output (information) to other users in the environment, without interacting with the system.
User Input Task		The user is receiving input (information) from other users in the environment, without interacting with the system.
Display Application Task		This task is performed by the system, after receiving some internal information. This task results in an output to be displayed to the user.
Computational Application Task		This task symbolizes an internal computation performed by the system without providing any output to the user.

Fig. 2. New task types introduced in [22]

Additionally, a development process for assistive user interfaces was suggested. The model of this process is shown in Fig. 3. One can see that based on a severe analysis of the tasks that have to be performed and supported some kind of task model is designed. This model is very important for the further development. It is the basis of the further development of implicit and explicit interactions. Implicit interactions are specified within task models. Explicit interactions are designed by the combination of tasks and dialogs. The navigation between different dialogs is specified by a so called dialog graph. It allows the automatic generation of supportive user interfaces.

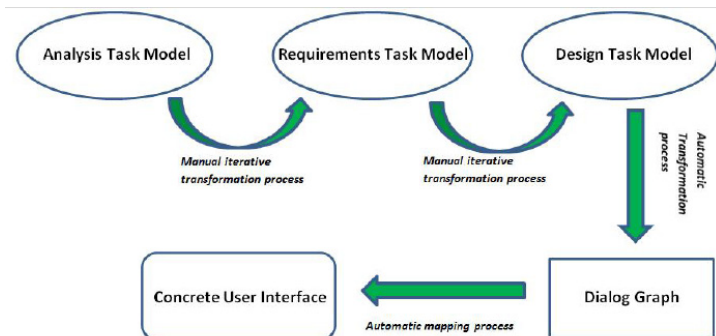


Fig. 3. Our application’s development flow from [22]

Dialog graphs were introduced some years ago for the development of interactive systems. It was extended in [22] to fulfill the special needs for supportive user interface design in smart environments by introducing implicit concurrent and implicit sequential transitions.

Initially, a dialog graph had only explicit concurrent and sequential transitions. They are activated by the user when interacting with the system and performing certain task. Sequential transitions result in a hiding of the old dialog and appearing the new dialog whilst concurrent transitions do not hide old dialog but give the activity to the new one.

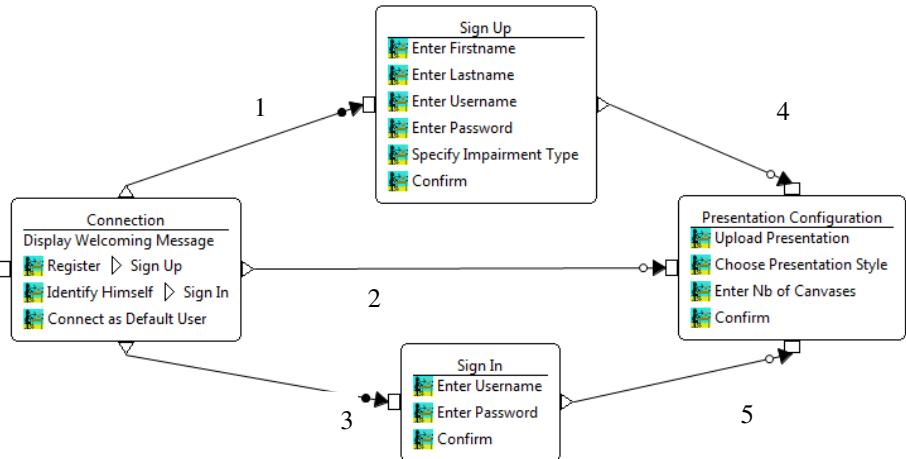


Fig. 4. Part of a dialog graph for a role presenter

The dialog graph of Fig. 4. consists of four dialogs with five transitions. Transition 1 and 3 are explicit transitions while all the others are implicit. They are related to activities that are part of the task model but are identified by sensors. In the example of the dialog graph of Fig. 4. there is a task “going to the front”. This task has to be performed before a presentation can be given. In case that some sensors signal that a person that is announced as “presenter” is in the presentation area, it can be concluded that the task “going to the front” was executed depending on the current dialog (“Sign in” or “Sign up”) an implicit transition (4 or 5) is executed. The user interface of the presenter is updated accordingly. He or she can load the presentation, choose the presentation style and specify the number of canvases. This input can be considered as the selection of a presentation pattern. This aspect will be discussed in more detail in the following paragraph.

4 Patterns in Smart Environment

Design patterns have proved [10] to be a good tool to represent knowledge in software design. They spread through computer science domain despite the fact that patterns were first discussed in architecture [1]. Additionally, many approaches take benefit of the usage of patterns in the HCI area [17]. Breedvelt-Schouten et al. [4] introduced task patterns that inspired our work. Sinnig [15] provided generic task patterns to be able to adapt a pattern to the context of use.

In a given smart environment numerous actors try to achieve a common goal that can be characterized as team goal. For the meeting room example, the ultimate goal is the efficient exchange of information among the actors in the room. Every task executed by an actor in its role is in a way a contribution to the team goal. It is a step towards this goal. Additionally, the task helps to reach the own individual goal (e.g. to make a good presentation).

A first step to develop patterns in the context of smart meeting rooms was to identify possible team goals (a certain state that the team wants to reach). First results were presented in [23] by providing six abstract team goals. These goals were (I) conference session performed, (II) lecture given, (III) work defended, (IV) topic discussed, (V) debate managed and (VI) video watched. Some further patterns were identified in the meantime. One of these patterns is presented in Fig. 5. This pattern was identified in an institute of climate research. It is a team pattern for discussing weather phenomena.

Usually during meetings at this institute there is first a general presentation. Later on participants split into two subgroups and discuss some pictures and data. At the end the combined results from both groups are presented to the whole plenum.

This kind of patterns can help to structure an application in an appropriate way. The pattern follows a user-centered approach. It can be considered as static. There will be no changes for the pattern instance at run-time.

However, there are other types of patterns that have to be instantiated during run-time and different instances are used. This is especially true for presentation patterns. Each presenter might have a different style for presenting his ideas. Some stakeholder might give a presentation in the classical way of presenting one slide after another. If more than one projector is available it might make sense to use one for the outline of a talk and the others for the slides.

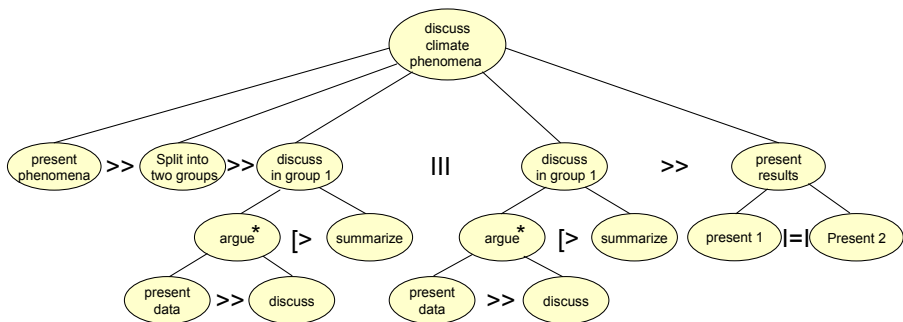


Fig. 5. Goal Pattern for discussing climate phenomena

This led us to the identification of information-distribution patterns. We will describe such patterns by concentrating on projectors but similar patterns are applicable to the general distribution of information to different devices. First, we tried to identify the generic parts of such a pattern. This is at least the name of the file where the information comes from, the number and the identification of the projectors. Finally,

the presentation style is necessary. This generic part can be specified by four parameters. This can be represented by the notation given in Fig. 6.

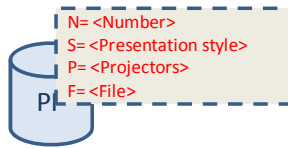


Fig. 6. Presentation Pattern

Instances of such a pattern are interpreted during run-time. All generic parameters missing an assigned value will be displayed by a supporting user interface for a mobile device. In this way the current presenter is asked to interactively provide the necessary information.

One can imagine that instances of patterns can be since the modeling stage. In this case values can already be assigned to parameters of these pattern instances. The file or the number of projectors might have been known during at the design phase. We attached several instances of the information distribution pattern to the team goal pattern of Fig. 4. The result is presented in Fig. 7.

First (left hand side of the model) only projector 1 is used to show the data from file X.ppt. Later in the discussion phase two projectors are used by each subgroup. The number of available projectors is two and the projectors are already explicitly assigned. Group 1 is sitting in one corner of the room using projector 1 and 2 and group 2 is sitting in another corner using projector 3 and 4. The presentation style is sliding window (sw), which means that always the current and the previous slide are presented. In case there were more projectors more slides are shown. The file that has to be shown is not known during design time.

At the end of the meeting only one projector is used and this is projector 1. The file name will be available during run time and is not known beforehand.

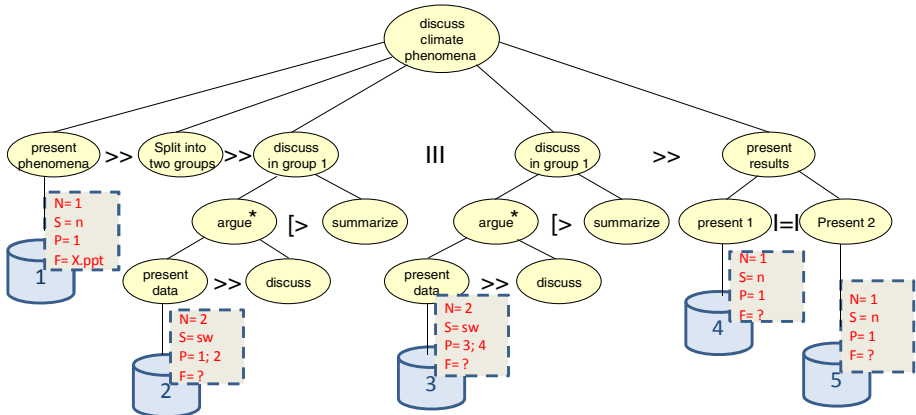


Fig. 7. Information-Distribution Pattern as part of a Goal Pattern

5 Task Migratability in Smart Environments

Task migratability is one of the usability criteria of interactive systems. It specifies the transfer of control for tasks execution between user and system. “It should be possible for the user or system to pass control of a task over to the other or promote the task from a completely internalized one to a shared and cooperative venture” [9].

Many interactive systems are static in this respect. The software designer decides often already during the development phase which task is to be allocated to which actor. In our discussion we will especially focus on the dynamic allocation of tasks (task migratability) and the possibility to influence this allocation by a supportive user interface.

Currently task migratability seems to be not a big issue in smart environments. In general the systems try to support users as much as possible. Sometimes it is possible to explicitly configure the environment via a user interface [13]. However, the concept of a Meta-UI is not directly related to task migratability. Often user interfaces are only distributed in a different way while the allocation of tasks remains the same. However, the concept of Meta-UIs can also be applied in such a way that a new configuration of a system results in a different task allocation. Consequently, Meta-UIs combined with supportive user interfaces can then be employed to make task migratability conceivable and possible in smart environments.

Tangible user interfaces seem to be an interesting option for supportive user interfaces. Tracked objects can help to identify the desired kind of support based on the inferred meeting type (brain storming session, workshop, business meeting, coffee break, etc.). The environment can be configured in such a way that a coffee pot on the table announces a business meeting. If the coffee pot is placed on the side board a workshop is performed. A coffee pot on the windowsill signals a brainstorming session and finally during a coffee break the pot has to be placed on a small table next to the big meeting table.

In this case, the coffee pot plays the role of the supportive user interface. Its location configures the provided support.

Additionally, objects can also be used to signal the environment level of support that is appreciated. On the one hand, a coffee pot standing on the big meeting table might express that all available support in the environment should be provided. On the other hand, if the pot stands on the window sill no assistance is needed. Users want to control everything in a manual way. Certain states in between these both extreme states can be specified as well.

However, the usage of tangible user interfaces in smart environments raises new challenges which are formulized in the following set of questions:

- a) Should already existing objects from the domain be used or specific new objects be introduced?

While existing objects might be more convenient, they might have the disadvantage that they are used for their original purpose and thus placed somewhere. New introduced objects like stones seem to be safer and less confusing because the manipulation of those objects will not be performed often since they do not play another role in the room.

- b) Should one object in different states/locations or several objects be used to specify the input to the environment?

There seems to be context dependent learnability problem. Is it easier to memorize the different states of one object or different objects?

- c) Should existing metaphors in favor of new introduced metaphors be used?

There is again the question of learnability. Does the metaphor fit to the mental models of the users? Is it convenient for the users to act according to the metaphor?

There seems to be no general answer for all of those questions. Based on a thorough analysis of the application domain, design decisions have to be made like in classical interactive systems

6 Summary and Outlook

In this paper we argued for a model-based approach for smart environments. We presented some details of our specification language CTML that allows specifying the tasks of different actors and the cooperation of a team. It is argued to split the specification into a cooperation model and a configuration model. The cooperation model specifies general knowledge of activities of a specific domain. This knowledge is long lasting. The configuration model has to be specified according to the current instance of a session. Who are the participants that take part and which roles do they play?

References

1. Alexander, C., Silverstien, M.: A Pattern Language. In: Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., King, I.F., Angel, S. (eds.) *Towns, Buildings, Construction*. Oxford University Press, New York (1977) ISBN 0195019199
2. Blumendorf, M.: *Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces*. Dissertation. Technische Universität Berlin (2009)
3. Bobick, A.F., Intille, S.S., Davis, J.W., Baird, F., Pinhanez, C.S., Campbell, L.W., Ivanov, Y.A., Schtte, A., Wilson, A.: The kidsroom: Perceptually based interactive and immersive story environment. In: *PRESENCE*, pp. 367–391 (1999)
4. Breedvelt-Schouten, I.M., Paterno, F., Severijns, C.: 00000. In: *Proceedings of DSV-IS*, pp. 225–239 (1997)
5. Coutaz, J.: Meta-User Interfaces for Ambient Spaces. In: Coninx, K., Luyten, K., Schneider, K.A. (eds.) *TAMODIA 2006*. LNCS, vol. 4385, pp. 1–15. Springer, Heidelberg (2007)
6. Demeure, A., Lehmann, G., Petit, M., Calvary, G. (eds.): *Proceedings of the 1st International Workshop on Supportive User Interfaces: SUI 2011*, Pisa, Italy, June 13 (2011), <http://ceur-ws.org/Vol-828/>
7. Demeure, A., Lehmann, G., Petit, M., Calvary, G.: SUI 2011 Workshop Summary Poster. In: [6]
8. Dittmar, A., Forbrig, P.: Selective modeling to support task migratability of interactive artifacts. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) *INTERACT 2011, Part III*. LNCS, vol. 6948, pp. 571–588. Springer, Heidelberg (2011)

9. Dix, A., Finlay, J.E., Abowd, G.D., Beale, B.: *Human-Computer Interaction*, 3rd edn. Prentice-Hall, Englewood Cliffs (2003)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley (1995)
11. Ishii, H., Ulmer, B.: Tangible bits: towards seamless interfaces between people, bits, and atoms. In: *Proceedings of the CHI 1997 Conference on Human Factors in Computing Systems*, Atlanta, Georgia, pp. 234–241 (March 1997)
12. Paterno, F., Meniconi, C., Meniconi, S.: ConcurTaskTrees: A diagrammatic Notation for Specifying Task Models. In: *INTERACT 1997, IFIP TC13*, pp. 362–369 (1997)
13. Roscher, G., Blumendorf, M., Albayrak, S.: Using Meta User Interfaces to Control Multimodal Interaction in Smart Environments. In: *Proceedings of the IUI 2009 Workshop on Model Driven Development of Advanced User Interfaces* (2009), <http://ceur-ws.org/Vol-439/paper4.pdf>
14. Roscher, D., Lehmann, G., Blumendorf, M., Albayrak, S.: Design and Implementation of Meta User Interfaces for Interaction in Smart Environments. In: [6]
15. Sinnig, D.: *The Complexity of Patterns and Model-based Development*, Computer Science. Concordia University, Montreal (Thesis (2004))
16. Srivastava, M., Muntz, R., Potkonjak, M.: Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments. In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, MobiCom 2001*, pp. 132–138. ACM, New York (2001)
17. Tidewell, J.: *Interaction Design Patterns: Twelve Theses*. In: *Proc. Conference on Pattern Languages of Programming, PLoP 1998*, Monticello, Illinois (1998)
18. Zaki, M., Wurdel, M., Forbrig, P.: Pattern Driven Task Model Refinement. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J.F. (eds.) *DCAI 2011. AISC*, vol. 91, pp. 249–256. Springer, Heidelberg (2011)
19. Molina, A.I., Redondo, M.A., Ortega, M., Hoppe, U.: CIAM: A Methodology for the Development of Groupware User Interfaces. *Journal of Universal Computer Science* 14, 1435–1446 (2008)
20. Mori, G., Paternò, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. *IEEE Trans. Software Eng.* 28(8), 797–813 (2002)
21. Wurdel, M.: *An Integrated Formal Task Specification Method for Smart Environments*. PhD Thesis, University of Rostock (2011)
22. Zaki, M., Forbrig, P.: Making task models and dialog graphs suitable for generating assistive and adaptable user interfaces for smart environments. In: *PECCS 2013*, Barcelona, Spain, February 19–21 (2013)
23. Zaki, M., Wurdel, M., Forbrig, P.: Pattern Driven Task Model Refinement. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J.F. (eds.) *DCAI 2011. AISC*, vol. 91, pp. 249–256. Springer, Heidelberg (2011)