Interactions in Smart Environments and the Importance of Modelling

Peter Forbrig
University of Rostock, Department of Computer Science
Albert-Einstein-Str. 21
D-18051 Rostock, Germany
+49 381 498 7620
peter.forbrig@uni-rostock.de

ABSTRACT

One challenge in software engineering is the development of smart environments that help users to intuitively accomplish their tasks. The ideal smart environment dynamically manages a diverse collection of devices, is accessible by multiple users and effectively supports the users' tasks. The design of smart environments relies on detailed models of devices, users and their tasks.

In this paper, we present our modelling language "CTML" specifically developed for smart environments. We demonstrate how the language was designed and how it was used for usability evaluations in a virtual smart environment. We then discuss the importance of "task migrateability" – a usability principle often neglected by contemporary smart environments. We argue that the proper implementation of this usability criterion can improve the usability of smart environments. Finally, we investigate how tangible user interfaces are related to smart environments and how this interaction technique can be used to support task migrateability.

Keywords
Supportive user interface, task migratability, smart environment

INTRODUCTION

One of the starting points for the development of new kinds of systems assisting users was Marc Weiser's vision of ubiquitous computing [22]. According to this vision supporting devices are weaving themselves automatically into everyday life in such a way that allows people to concentrate on their tasks.

Such an environment is considered to be smart. It tries to analyze the user's behaviour and to provide appropriate assistance.

In recent years, such systems made their way from research to industrial applications. Let us consider the domain of meeting rooms where the performance of workshops has to be supported. Within this context a presenter should be able to concentrate on his talk, while the smart environment (SE) intervenes by adjusting the projector, loading the necessary files and capturing audiovisual data for meeting documentation if needed. In the best case no direct interaction is necessary.

Figure 1 gives a visual impression of our laboratory where experiments are performed.

Figure 1: Smart Meeting Room in Rostock

Implicit interaction like going to the presentation area is enough to present the slides of the speaker. Experiences show [13] that the quality of support can be increased if some information is given to the system. Most important are the tasks the users want to perform within the environment.

Task models are an appropriate starting point for interactive processes development [7,9]. In [3] it is suggested to use dynamic task models in order to build adaptive user interfaces. The application of task models for smart environments is discussed in [11] and [16]. We will shortly discuss the collaborative task modelling language CTML. More details of the language can be found in [23]. Afterwards we will discuss the aspect of usability evaluation for smart environments. The virtual environment ViSE will be used to demonstrate possible tool support for usability evaluation. Finally we will discuss aspects of interactions in smart environments and the role of task migrateability. It will be also discussed whether tangible user interfaces can be considered as supportive user interfaces. At the end we conclude our ideas and summarize the main contributions presented in the paper.

The Collaborative Task Modelling Language CTML

The collaborative task modelling language (CTML) was developed in conjunction with modelling efforts in smart environments. It is specified in detail in the thesis of Maik Wurdel [23] and supports the idea of stakeholder-driven process management and has the potential to be used outside the context of smart environments [10].

Fundamental Assumptions

Four fundamental assumptions were the basis of the design of CTML:

1. Role-based Modelling.

   In limited and well-defined domains the behaviour of an actor can be approximated through her role.

2. Hierarchal Decomposition and Temporal Ordering.

   The behaviour of each role can be adequately expressed by an associated collaborative task expression.

3. Causal Modelling.

   The execution of tasks may depend on the current state of the environment (defined as the accumulation of the states of all available objects) and in turn may lead to a state modification.

4. Individual and Team Modelling.

   The execution of tasks of individual users may contribute to a higher level team task.

Collaborative Model

Based on these assumptions a collaborative model is specified in a two-folded manner:

a. Cooperation Model.

   Specifies the structural and behavioural properties of the model like roles and task models

b. Configuration(s).

   Specifies runtime information (e.g. actors assigned to roles).

For each cooperation model several configurations may exist in order to describe different situations in which the model is used. These configurations specify the instances of elements of the cooperation model and their behaviour.

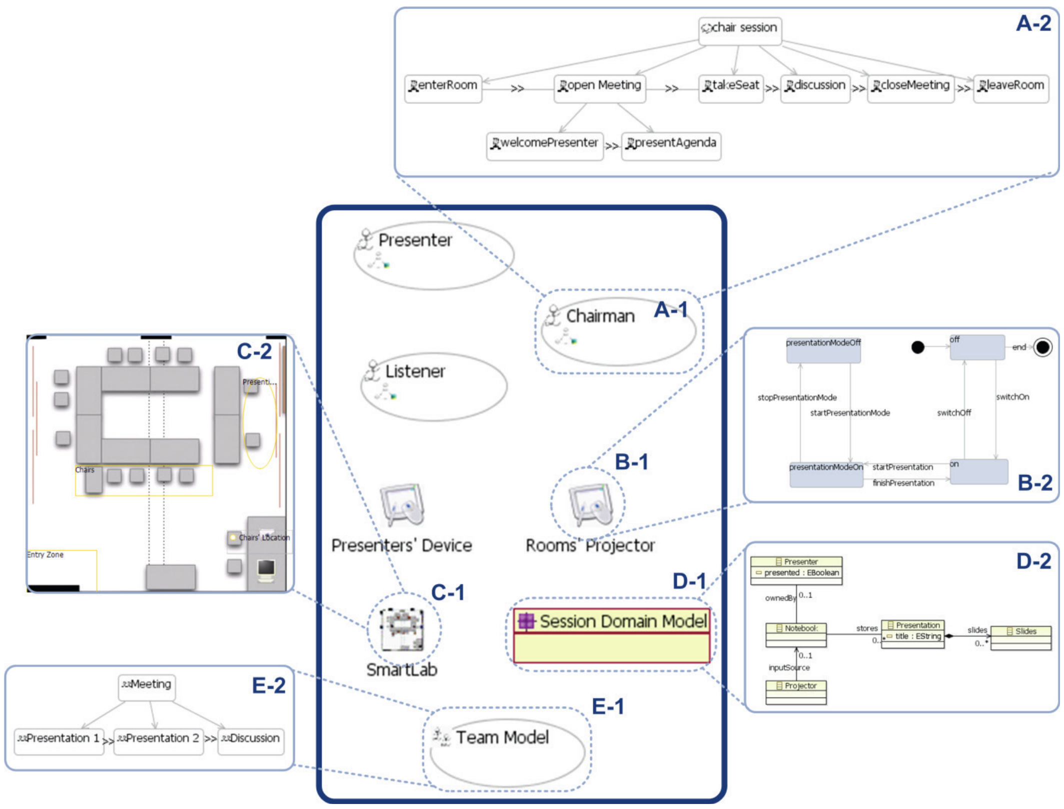


Figure 2: Schematic Cooperation Model for Meeting Scenario

Fig. 2 presents a schematic sketch of a cooperation model. Elements in the inner circle represent modelling entities (post fixed with "-1") whereas diagrams outside of the inner circle show detailed specifications of the corresponding entities (post fixed width "-2").The cooperation model specifies the relevant entities on an abstract level. Usually roles (e.g., A-1), devices (e.g., B-1), a location model (C-1), a domain model (D-1) and a team model (E-1) are necessary and can be specified.

The potential actions a user is able to perform are determined by his role(s). More precisely a role is associated with a collaborative task model (A-2 in Fig.4), which is visually represented by a task tree in a CTT-like notation [18]. Tasks are arranged hierarchically defining a tree structure. Atomic tasks, non refined tasks, are referred to as "actions". In addition, tasks on the same level of abstraction can be connected via temporal operators defining the temporal order of task execution.

Role modelling is a common concept in software engineering ([6; 10]). For use cases this concept is known under the term "actor". In the following, we will use the term actor as an instance of a role, or a person that acts according to the role. Roles are in this way abstractions of actors sharing the same characteristics. They categorize users of the same kind in terms of capability, responsibility, experience and limitations according to the domain.

In [10] it is stated that a user is not limited to one role at a time and role switching is often taking place. In CTML the role concept is employed to define the pool of actions of a user by means of task expressions. In task analysis and modelling, this approach is quite common but is usually restricted to a one-to-many relation of role and user [15; 16]. However this is a rather rigorous constraint. In the domain of smart environments it is frequently the case that an actor changes his role at runtime and that one role is being performed by several actors simultaneously. This might be the case in our modern business world as well. The role concept implemented in CTML incorporates this case.

In the example of Fig. 2 the roles are "Presenter", "Listener" and "Chairman". They represent the different types of stereotypical behaviour in the meeting scenario. . At a given time t, several persons are acting as listeners while there is normally only one presenter and one chairman.

Besides the cooperation model a CTML specification also contains one or more configurations providing essential runtime information for the cooperation model. A configuration represents necessary information for a concrete situation. Different settings can follow the same cooperation model. Obviously, different persons can execute a meeting in a similar way. This is e.g. true for defending a thesis.

As the cooperation model relies on a role-based specification actors operating in the environment need to be defined in accordance with a corresponding actor-role mapping. More precisely an actor may fulfil more than one role concurrently and a role may be assigned to different actors simultaneously. Moreover, not only concurrent roles fulfilling is allowed but also all other temporal operators defined in CTML are possible.

None of the currently existing task modelling languages supports this assumption even though this is a frequently encountered case while working cooperatively. Considering the example of the "Conference Session" one can imagine the case of an actor presenting a paper in front of the audience but also listening to other presentations afterward. Therefore, the simultaneous (or more precisely ordered) performance of more than one role is an important feature of the language as it also allows separating a role from another since they are assembled at runtime. Thus modularization and separation of concerns are achieved. Additionally some properties of actors are defined (e.g., initial position in the environment).

An example of a configuration for the schematic Cooperation Model of Fig. 2 is presented in Fig. 3.

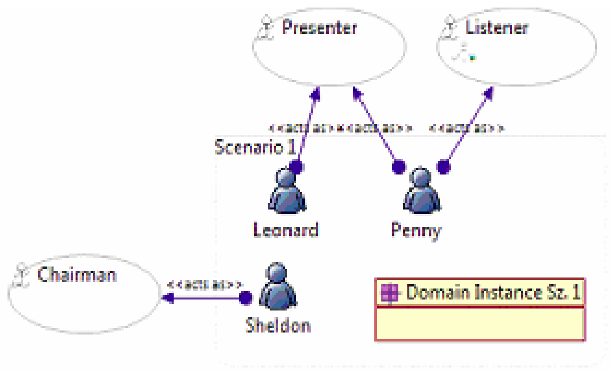
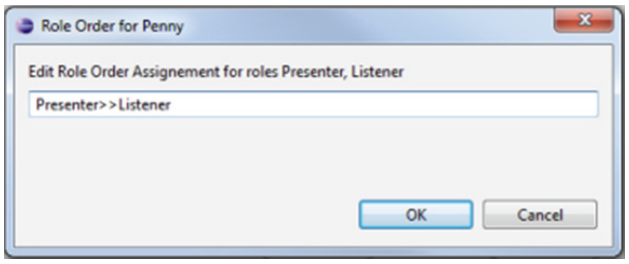


Figure 3: Configuration for the Cooperation Model

Not all before mentioned information have visual counterparts however, the actor-role mapping is represented by arrows. Penny fulfils the role Presenter and Listener. Sheldon acts as a Chairman and Leonard as a Presenter. The precise assignment of temporal operators for an actor fulfilling more than one role is performed in the dialog depicted in Fig. 4.


Role Order for Penny

Edit Role Order Assignement for roles Presenter, Listener

Presenter>>Listener

Figure 4: Assignment of Actors to Roles

In this example Penny first acts as "Presenter" and then as "Listener".

A configuration specifies the conditions under which the cooperation model is used. Instances of model elements are specified and behavioural information maybe additionally provided.

Beside the temporal relations between tasks CTML allows more specific relations in a similar way like OCL. These constrains can be specified between tasks but also between other model elements. CTML allows in this way the specification of preconditions and effects of tasks. With respect to the task expressions of the role chairman and the role presenter the following preconditions can be defined.

a) A presenter is only allowed to start his presentation if the talk had been announced by a Chairman.

```
Chairman.oneInstance.AnnounceTalk
```

b) A presenter can only respond to questions can if a Chairman has opened the discussion.

```
Chairman.oneInstance.OpenDiscussion
```

c) The precondition of the task of the chairman states that a discussion can only be announced if all presenters have finished their presentation beforehand.

```
Presenter.allInstances.EndPresentation
```

Effects of tasks can be specified by OCL-like expressions as well

a) After a chairman announced the discussion all notebooks in the room are switched off.

```
Notebook.allInstances.switchOff
```

b) After a presenter started his presentation a projector is switched on.

```
Projector.oneInstances.switchON
```

c) After a presenter finished his presentation this is recognised in the corresponding model. It results in setting the attribute presented of the current presenter to true.

```
self.presented=true
```

Preconditions defined on this level of abstraction integrate very well into the CTML approach of role based descriptions. Quantifiers can be used to specify how many actors fulfilling the role are addressed (one or all) by the corresponding condition.

Additionally effects can be specified to characterize the consequences of performing a task. Such effects are especially important during requirements analysis. They allow to precisely explore the domain and to support the animation of the models.

Tranformations

It was already mentioned that our 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 try to infer next possible actions of the users and based on that information convenient assistance is to be provided.

Bayesian Networks have to be specified and have to be trained. Additionally, the number of nodes in the network can be really huge. Therefore, the specification of such networks is time consuming.

It is possible to slightly extend task models by priorities and take such models as input for a transformation that generates a Bayesian network with initial values for probabilities of transitions. The idea was published in [12] and will be shortly described here again.

Let us assume that a workshop has to be organized in the smart meeting room. There is the following plan:

A. Presenter Paul gives a talk
B. Presenter Sheron gives a talk
C. Presenter Leonard gives a talk
D. Discussion of all presentations

The idea is that Paul starts the workshop with his presentation. Afterwards Sheron will give her talk. According to the plan Leonard is the third speaker. The workshop will be finished by a discussion. The tasks A, B, C and D can be considered as a schedule of the workshop. This schedule can be considered as a team model that all participants want to execute in a collaborative manner. This team model is specified as task model in our approach (see Fig. 1).

For the workshop schedule the corresponding task model is presented in Fig. 5. (We decided to use the choice "|=|" temporal operator instead of the enable ">>" one because variations of the schedule should be possible. This can also be reached if the temporal operator is defined a little bit more flexible.)

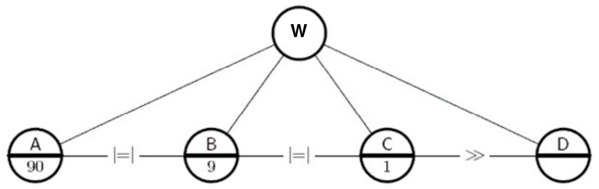


Figure 5: Task model with priorities

Task A, B and C can be executed in any order but A has the highest priority (90). B has the priority 9 and C the priority 1. This can be interpreted in such a way that C starts with 1%, B with 9% and A with 90%.

Based on this information an initial Bayesian Network can be generated.

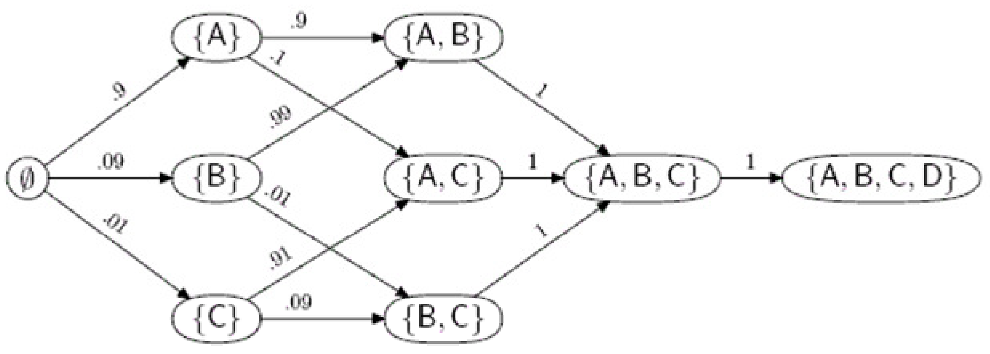


Figure 6: Bayesian Network

The nodes in the network represent the already performed tasks. Initially none task was performed. There is the probability of 90% (.9 from 1.0) that task A is performed first. After A was performed (Node {A}) there is the chance of 90 % that task B is performed and 10% that Task C is performed. After two talks were presented with 100% probability the third missing talk will be given. After all three talks were given there will definitely be a discussion.

For a given task model the corresponding initial Bayesian Network can be generated automatically. A plug-in for Eclipse was developed for that purpose.

The generated Network can be used immediately and can be further trained.

The Role of Models

Models play an important role in software engineering in general. We already discussed that they are important for the development of smart environments as well. The requirement process of such systems needs to incorporate several entities. We consider actors, stationary and personal devices with their interplay and additional constraints (e.g. location and domain information) as most relevant.

Thus, the result of the requirements analysis phase is a specification describing the interaction between actors and their devices. Based upon the previous sections and experimental modelling we found the next three phases to be of particular importance.

The first phase is about creating a scenario that is written in natural language as text and which describes the actors, their dependencies and the envisioned behaviour of the system under development on a high level of abstraction.

Next, use cases can be specified using the created scenario.

In the last phase task models are created based on use cases.

.

The presented specification language CTML can be used for different purposes.

Possible usages are:

1. Requirements Analysis

   Similar to use cases, task models allow developers to elaborate whether the main functionality of a system has been understood.

2. Runtime Support

   Interpreters are able to animate the models. In conjunction with sensors in the room the animation can be triggered and control the provided assistance.

3. Basis for further models

   Cooperative task models can be transformed to Bayesian networks that get initial values and are later further trained. Such networks can control the assistance provided by the room.

The discussed approach with its variations is always focusing on the tasks users have to perform.

Following a user-centred approach is always a crucial factor to successfully assist people while performing their tasks.

Technology driven prototypes have shown that user-centeredness is often disregarded. Satisfying user needs is the most important objective for smart environments. Therefore thoroughly performed requirements engineering is crucial to build those systems. In particular, the process is driven by use cases which are built upon scenarios in early stages.

USABILITY EVALUATION

To ensure an adequate integration of all software and hardware components, user studies have to be conducted.

One should start with test cases that are specified as screenplays and later allow people to freely interact. This requires a lot of resources since several people have to act in the environment. Therefore a certain level of quality should be reached beforehand.

For traditional interactive software "expert evaluation" is an accepted method to identify bugs before user testing. Unfortunately, expert evaluation is not feasible for a cooperative smart environment where several people have to act.

One idea is to use a virtual environment that allows an expert to control several virtual persons. This idea was mainly developed and evaluated by

Expert Tests

ViSE (Virtual Smart Environment) [16] is a prototypical implementation of such a system. It was mainly developed by Stefan Propp [19].

An expert can perform the following interactions with the virtual room.

He can:

- Change the location of a person
- Attach items to persons
- Change device states
- Establish connections between devices

Fig. 7 provides a first impression of the virtual environment by presenting parts of the user interface. Those parts of the user interface are taken by a screenshot while running an example.

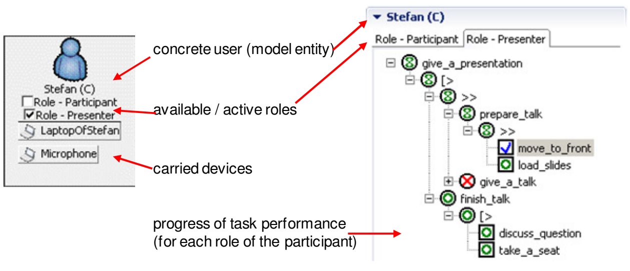


Figure 7: Specification of a user

On the left hand side of Fig. 7 one can see the interaction options for an expert for changing the state of an actor while on the right hand side the current state of one task model of this actor is visualized. This task model is related to the role an actor is performing.

Currently Stefan plays the role of a Presenter. This can be seen on both sides. The role presenter is clicked and the tab for the role presenter is visible. In the configuration model it must have been specified that Stefan plays the role of the "Participant" and the role of "Presenter" because both options are visualised in the graphical user interface. Additionally, a laptop and a microphone were identified to be attached to Stefan. This can be done interactively or can be the result of sensing.

One can see details of the task performance on the right hand side of Fig. 5. The state of the animated task models can be seen for each role. The roles are represented by different tabs.

In his role as Presenter Stefan moved to the front of the audience (move_to_front was executed). Additionally, it can be seen that with this action he started to prepare his talk (prepare_talk is running). At the moment it is still not possible to give his talk (give_a_talk is blocked) because the slides have to be loaded (load_slide task is enabled) to finish his preparation.

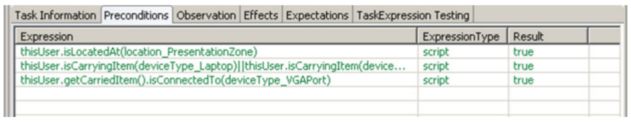
However, by taking a seat (take seat is enabled) Stefan could finish his talk without giving his presentation or discussing any questions. There is no problem if models are specified in this way. They are only used to specify the expected behaviour and to provide corresponding assistance. There is no need to specify the models in a more strict way.

In this view (right part of Fig. 7) one can see the temporal relations between tasks as well. They are presented in specific nodes meaning that all sub-nodes follow this temporal relation. The enabling operaror (>>) menas e.g. that move_to_front, has to be finished before load_slides can be executed.

This representation of the running task model is in some way a short reminder of the model specification the animation is base on.

It was already mentioned that tasks can have preconditions and effects. It is of course of great interest for the usability expert to see these specifications. Sometimes this might even be necessary to understand the current state of a task.

The virtual environment ViSE gives support in this respect as well. It provides a special view for constrains and their values during animation. Fig. 8 provides a view on the preconditions for the task "present".



Expression	ExpressionType	Result
thisUser.isLocatedAt(location_PresentationZone)	script	true
thisUser.isCarryingItem(deviceType_Laptop)||thisUser.isCarryingItem(device...	script	true
thisUser.getCarriedItem().isConnectedTo(deviceType_VGAPort)	script	true

Figure 8: Evaluated Constrains

For the presentation task there are three preconditions specified. Those preconditions can be interpreted in the following way. A Presenter has to be located in the "Presentation Zone" and has to carry a device of type laptop or PDA. Additionally this device has to be connected to a VGA port. All three pre conditions have to be fulfilled before the task give talk can be started.

Constrains can be edited textually or the state of the system has to be changed if constrains are not fulfilled.

Fig. 7 gives an impression of how the two dimensional virtual environment looks like.

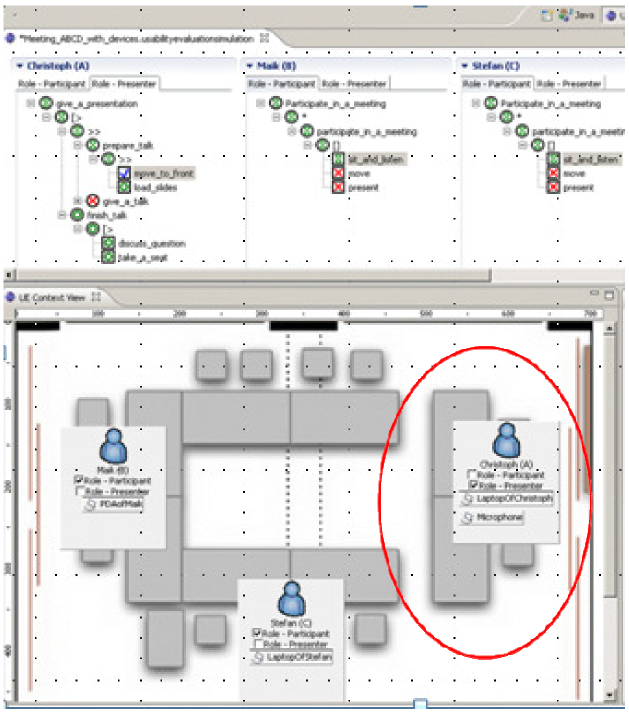


Figure 9: Location model in ViSE

In the environment different zones can be specified. In the screenshot of Fig. 9 only the presentation zone is visualised. An example with more zones (one entrance zone, three different listener zones and two presentation zones) is presented in the screenshot of Fig. 10.

In the example of Fig. 9 Stefan is visible in the middle of the bottom of the Figure. He is outside of the presentation zone and would not be able to give his talk. At the moment, Christoph is in this zone.

The usability expert can move Christoph out of the presentation zone and move Stefan into this zone.

Constrains and animated models will be updated accordingly. The expert can observe whether the models changed in a way that seems to be reasonable. If this is not the case he can refer to the problem and ask for changing the models. Often new constrains will be introduced or existing ones have to be changed.

The usability expert is really able to perform experiments employing the virtual environment. He can e.g. move two persons into the presentation zone and check what happens. Discussions can be inspired by precise examples and decisions can be made based on the specified models.

User Tests

After expert evaluation was performed and problems in the models were solved further tests are necessary. Real users have to perform tasks in the smart environment. They have first to act according to predefined scenarios and can act later more freely.

Having recorded theses evaluations by videos, an emerging problem is the participants' privacy. One can edit the video in such a way that people are not recognized anymore but this is very time consuming. It would be better if there is a system that is able to visualise the data that are captured by sensors in the environment.

ViSE can be used for this purpose as well. It can be used to present interactions in the meeting room in an anonymous way. A common language for the real and the virtual room were defined In this way both environments can communicate with each other.

ViSE was enhanced by a replay mode that allows replaying captured sensor data with different animation speeds.

Movements are visualised in a bird's eye view (see Fig. 10). In this example it is again the meeting room. Users and devices are represented by icons. Not only the location and the movement are visualised but also traces representing previous locations.

During replay further expert observations can be annotated. It might be possible that additional tasks were identified. A corresponding information can be stored.

Further views provide several logs and visualizations of sensor data on demand, for instance the progress of task performance as animated task model.

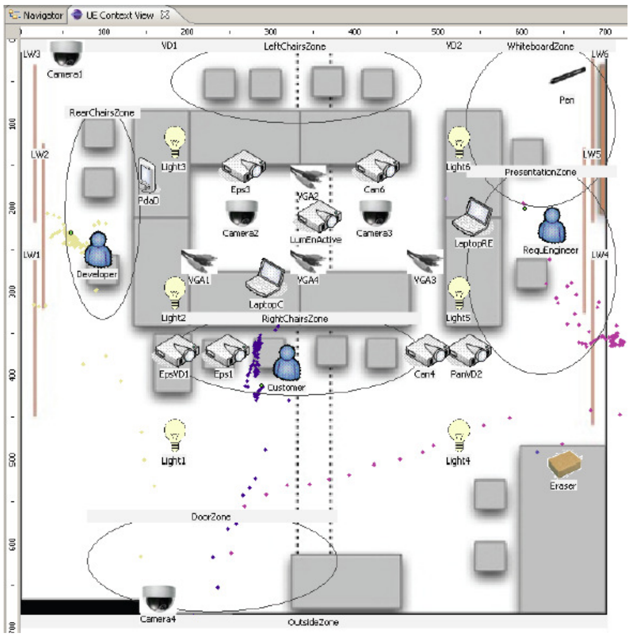


Figure 10: Replayed meeting ViSE

SUPPORTIVE USER INTERFACES IN SMART ENVIRONMENTS

The general term of "supportive user interfaces" fits to nearly all interactive applications as in some way every user interface has to be supportive. As a result of the Supportive User Interface workshop SUI 2011, [5]) participants agreed on the following more specific and precise definition:

"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." [6].

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 is based on the "Meta-User Interface" approach [4] that has been introduced to control and evaluate the state of interactive ambient spaces.

We will focus in our discussion on the role of supportive user interfaces in smart environments.

With task models and Bayesian networks we already discussed different approaches to control smart environments. Such models try to infer next possible actions of the users and based on that information convenient assistance is 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 those systems and to adapt them according to their needs." [16]

Meta-UIs are mentioned by the authors as a solution for this problem.

Roscher et al. [20] discuss a functional model and a system architecture for Meta-User Interfaces for smart environments. They focused on the development of a user interface that allows controlling devices in different ways.

"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."

For ubiquitous user interfaces the five features shapeability, distribution, multimodality, shareabilty and mergeability are specified and presented in [21]. 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 sequentially) 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 characterize the technical properties of user interfaces in a given ubiquitous environment. However, the usability of such user interfaces is not yet considered.

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. We will also discuss the role of tangible Meta-UIs.

TASK MIGRATABILITY

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. There are rarely systems where control between user and software system can be changed on different levels. Some systems like an autopilot of an airplane exist where the pilot can give control to the system or take it back.

The Importance of Task Migratability

An interesting study has been conducted by V. Hinze-Hore [14]. She looked at the publications of the ten most cited authors in HCI and counted the number of times that a particular HCI principle was proposed by one of these significant authors. Multiplying this number by a weighting factor derived from the author citation frequency allowed a ranking of HCI principles to be determined.

According to this procedure, task migratability is ranked the fifth among all usability principles.

A crucial pre-request for delivering an optimal assistance to a user is to recognize in which way the current task of the user is at best supported by an interactive system.

Sometimes it might be easier to perform a sub-task without computer support. Therefore "designers need to acquire a deeper understanding of what the tasks of the users might be in certain situations and how to support their achievement." [8] Depending on the encountered situation, adaptations should be possible.

Task Migratabilty in Smart Environments

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. There is no discussion how users can reduce provided support. The interface of the room is fixed and cannot be configured or adapted.

Sometimes it is possible to explicitly configure the environment via a user interface [20]. This can be one solution to that problem. However, to realize such an idea, devices with which users can interact are required.

The concept of a Meta-UI is not directly related to task migratability. Often user interfaces elements are only distributed to different devices 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

Tangible user interfaces are often used in conjunction with table top systems. A user interacts with digital information through physical objects. The goal is a seamless coupling between the two very different worlds of bits and atoms [6].

In our smart environment a pencil plays such a role of tangible user interface. In case the pencil is in the box, no white board is needed and the cinema screen can go down (Figure 11). If somebody takes the pencil out of the box the cinema screen has to move up, otherwise the user might get the idea to write on the screen. Therefore, the smart environment has to react accordingly (Figure 12).

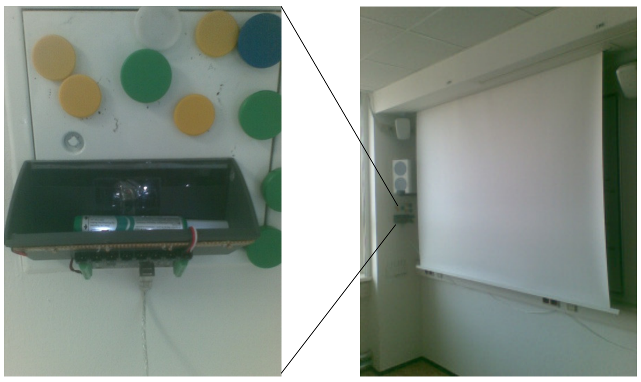


Figure 11 Pencil is in the box

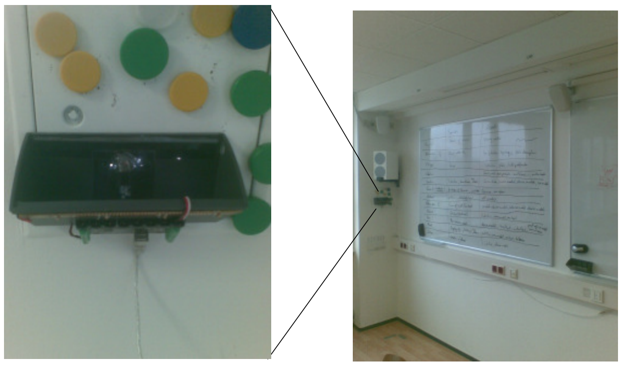


Figure 12 Pencil is in hand and not in the box

Sensors in the box help to achieve this behaviour. However, for the user the pencil seems to be the tangible user interface that allows him to interact with the environment. This pencil can be considered as a very simple supportive user interface. It allows configuring the room and the room itself can be considered as an interface to the people acting in it.

Tangible user interfaces seem to be an interesting option for supportive user interfaces. Physical objects can be used to configure the interface of the smart environment. 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.). Certain physical objects can be identified and selected to create a language for communication with the environment. The location of such objects and their special relation to each other can be considered as code. This code can be interpreted as a command by the environment.

Following agreements or definitions could be made. If the coffee pot is located on the table, this means that a business meeting is currently taking place. If the coffee pot is placed on the side board, a workshop is performed. A coffee pot on the window sill 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.

Here 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's level of support that is appreciated.

On the one hand, a vase standing on the big meeting table might express that all available support in the environment should be provided. On the other hand, if the vase 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 can be formulated in the following set of questions:

1. 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.

2. Should one object in different states/locations or several objects be used to specify the input to the environment?

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

3. Should existing metaphors in favour of new introduced metaphors be used?

   There is again the question of learn ability. 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 to all of those questions. Based on a thorough analysis of the application domain, design decisions have to be made like in classical interactive systems

TANGIBLE UI VERSUS GUI

Tangible user interfaces cannot express all necessary information in an appropriate way. They are very helpful if a specific state and which is an element of a limited state space has to be determined. This is possible for discrete levels of assistance or specific configurations of the room.

In the context of our application domain (smart meeting rooms), different presentation styles can be supported (presentation of the current slide with one projector, presentation of all slides in a sliding window mode, presentation of the outline with one projector and presentation of the current slide with another one, …). It will not be possible to support the loading process of a given presentation from the file system in a convenient way employing tangible interactions.

There might have been profiles specified, that allow selecting files based on predefined specifications in a tangible way. This would be easy to use. In the general case where an arbitrary file has to be uploaded, a graphical user interface is much more appropriate.

Again a broad variety of assistance can be envisioned. Everything is configured via a GUI like in [20] and [21], or everything is selected for predefined specifications by tangible objects. It is of course to follow a strategy that is something in between those extreme strategies. It depends on the context which strategy fits best.

For a very important meeting most facts might be known and can be configured beforehand. There might also be some time for a preparation beforehand.

During brainstorming sessions, ideas might come up to present something that has to be configured via a GUI or a tangible user interface.

Discussion

We introduced tangible user interfaces for smart environments as a subset of supportive user interfaces.

In fact tagged objects can play the role of a Meta-UI. They help to explicitly inform the environment about the intention of the users that have otherwise to be deduced implicitly from a lot of sensor data. This is especially helpful if no historical data are available that allow training the algorithms of the smart environment. In such cases the direct expression of intentions is very helpful.

Such a tangible user interface has the advantage of informing the environment explicitly. However, it delivers wrong results if objects are not manipulated in the correct way. In this case correct deduction might be overruled by the tracked objects that are not used in the expected way.

Tangible objects should therefore follow a metaphor that is easy to learn and to remember. Further studies have to show for each context the kind of metaphors and objects that are appropriate to be used for communicating with a smart environment.

CONCLUSIONS

In this paper we argued for using models in connection with smart environments, We insisted on the fact that they are helpful in the requirements analysis phase because they focus on the tasks users have to perform. It was discussed how the specification language CTML might be helpful in specifying and evaluating smart environments. The virtual smart environment ViSE was developed for this purpose. It was shown that expert evaluation is possible for cooperative tasks with such a virtual environment. Additionally, the replay of evaluations in the real environment can be provided in an anonymous way using ViSE.

Additionally we argued in this paper to consider task migratability as an important aspect for smart environments.

At the moment, this aspect plays only a minor role in the discussions. We believe that the acceptance of the concept of smart environments may increase if the user is able to influence the dynamic task allocation in a convenient way.

It has also to be shown by experiments that the combination of tangible and graphical user interfaces can improve the usability of smart environments. In both cases task migratability has to be taken more into account.

ACKNOWLEDGMENTS

The presented ideas were developed together with Anke Dittmar, Maik Wurdel, Stefan Propp and Michael Zaki during the last years. Some of the research was performed within the graduate school MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Applications) which is funded by the German National Research Foundation (DFG), Graduate School 1424.

REFERENCES

1. Blumendorf, M., Lehmann, G., Feuerstack, S., and Albayrak, S.: Executable Models for Human-Computer Interaction. In Interactive Systems. Design, Specification, and Verification, T. C. Graham and Philippe Palanque (Eds.). Lecture Notes In Computer Science, Vol. 5136. Springer-Verlag, Berlin, Heidelberg 238-251.
2. Blumendorf, M. Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces. PhD Thesis, Technical University of Berlin, 2009.
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 Proc. of The 9th IFIP Working Conference on Engineering for Human-Computer Interaction Jointly with The 11th International Workshop on Design, Specification and Verification of Interactive Systems, pp 11-13, 2004.
4. 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.
5. Demeure, A. Lehmann, G., Petit, M. and 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/.
6. Demeure, A. Lehmann, G., Petit, M. and Calvary, G. SUI 2011 Workshop Summary Poster, in [5].
7. Diaper, D., Stanton, N.: The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Assoc. Inc., 2003
8. Dittmar, A. and Forbrig, P., Selective modeling to support task migratability of interactive artifacts, Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction – Interact 2011, Lisbon, Portugal, Springer-Verlag, Volume Part III, ISBN 978-3-642-23764-5, p. 571-588.
9. Dix, A., Finlay, J.E., Abowd, G.D. and Beale, B.: Human-Computer Interaction, 3rd edn, Prentice-Hall, Englewood Cliffs (2003)
10. Forbrig, P., Dittmar, A., Brüning, J., and Wurdel, M.: Making Task Modeling Suitable for Stakeholder Driven Workflow specifications, Universal Access in Human-Computer Interaction. Design for All and eInclusion - 6th International Conference, UAHCI 2011, Held as Part of HCI International 2011, Orlando, FL, USA, July 9-14, 2011, Proceedings, Part I , pp 51-60.
11. Forbrig, P., Wurdel, M. and Zaki, M.: The roles of models and patterns in smart environments, EICS Workshop, Copenhagen, 2012.
12. Giersich, M., Forbrig, P., Fuchs, G., Kirste, T., Reichart, D., and Schumann, H:. Towards an Integrated Approach for Task Modeling and Human Behavior Recognition. In Proc. HCI International 2007: 12th International Conference on Human-Computer Interaction, volume 1 of LNCS, pages 1109–1118, Beijing, China, July 22-27 2007. Springer.
13. Giersich, M.: Concept of a Robust & Training-free Probabilistic System for Online Intention Analysis in Teams, PhD Thesis, University of Rostock, 2009.
14. Hinze-Hoare, V. Review and Analysis of Human Computer Interaction (HCI)Principles, July 2007, http://arxiv.org/ftp/arxiv/papers/0707/0707.3638.pdf
15. Ishii, H. and Ulmer, B., Tangible bits: towards seamless interfaces between people, bits, and atoms. In: Proceedings of the CHI'97 conference on human factors in computing systems, Atlanta, Georgia, March 1997, pp 234 – 241.
16. Luyten, K., Vandervelpen, C., Conix, K.: Task modelling for ambient intelligent environments – design support for situated tasks, In Proceedings of the 4th international workshop on Task models and diagrams (TAMODIA '05). ACM, New York, NY, USA, 87-94.Propp, S. and Forbrig, P.: ViSE – A Virtual Smart Environment for Usability Evaluation, in Proc. HCSE 2011, Reykjavik, 2010, Springer, pp 39-46.

17. Malý, I., Cuřin, J., Kleindienst, J., and Slavìk, P:. Creation and visualization of user behavior in ambient intelligent environment. In 12th International Conference on Information Visualisation (IV08) - International Symposium of Human-Computer Interaction (HCI), London, United Kingdom, July 2008. IEEE Computer Society, pp 497-502.
18. Malý, I., Slavik, P., and Kleindienst, J.: Combination of models and logs for visual analysis of data from usability evaluation. In Proceedings of the Third IASTED International Conference on Human Computer Interaction (HCI '08), Daniel Cunliffe (Ed.). ACTA Press, Anaheim, CA, USA, 279-284.
19. Propp, S. Usability-Evaluation in intelligenten Umgebungen, PhD Thesis, University of Rostock, 2011.
20. Roscher, G., Blumendorf, M. and Albayrak, S. Using Meta User Interfaces to Control Multimodal Interaction in Smart Environments, Proceedings of the IUI'09 Workshop on Model Driven Development of Advanced User Interfaces, http://ceur-ws.org/Vol-439/paper4.pd
21. Roscher, D., Lehmann, G., Blumendorf, M. and Albayrak, S. Design and Implementation of Meta User Interfaces for Interaction in Smart Environments, in [5]
22. Weiser, M.: The Computer for the 21st Century, Scientific American, 265, pp. 94-104, 1991
23. Wurdel, M.: An Integrated Formal Task Specification Method for Smart Environments, PhD Thesis, University of Rostock, 2011
24. Zaki, M., Wurdel, M., and Forbrig, P., Pattern Driven Task Model Refinement, International Symposium on Distributed Computing and Artificial Intelligence, DCAI 2011, Salamanca, Spain, 6-8 April 2011, Ed. Ajith Abraham and Juan M. Corchado and Sara Rodriguez-Gonzalez and Juan F. De Paz Santana, Advances in Soft Computing, Vol. 91, ISBN = 978-3-642-19933-2 , p. 249-256.