Models as a Starting Point of Software Development for Smart Environments

Peter Forbrig, Michael Zaki, and Gregor Buchholz

University of Rostock, Department of Computer Science,
Albert Einstein Str. 21,
18055 Rostock, Germany
{peter.forbrig,gregor.buchholz,michael.zaki}@uni-rostock.de

Abstract. Creating a smart environment is a challenging task because of the excessive software development and adaptation required. Additionally, hardware in form of stationary as well as dynamic devices has to be installed. Similar to traditional software development, evaluating only the end product is often very costly in terms of time and effort needed. This is due to the fact that usually a lot of changes have to take place since the system fails to deliver the expected behaviour. Therefore, modelling is of great benefit. Models help to get a shared and thorough understanding of a specific domain. Making the animation of those models feasible allows getting a first impression of the system under development. Such prototypes of a system can be created on different levels of abstraction. The paper aims to demonstrate how modelling the human behaviour from the perspective of the activities performed in the environment can lead to first abstract prototypes. Those prototypes can be further extended and fostered by device models as well as models for the whole environment. In the paper, we also strive to discuss the costs and benefits of offering an abstract environmental model in 2D or 3D.

Keywords: Smart Environment, model-based design, evaluation, prototyping.

1 Introduction

Model-based development of interactive systems is already quite a long tradition (see e.g. [11]). It is based on the idea of analyzing the tasks users currently perform and designing a task model of their envisaged activities. Based on this designed task model the user interface of the system under development is created.

Additionally, usability evaluation of the developed system can be supported by task models. This can even be done remotely (see e.g. [14]). However, it is important to find ways to elaborate in a very early development stage whether the ideas for a new system really fit to the requirements of the users. Prototyping seems to be a good solution for that. This statement is supported by [15]. Regarding evaluation techniques, the authors claim: "Discount methods work well with low fidelity prototypes, which allows evaluations to take place during early development when there is no operational prototype for users to test in a real work setting. These discount methods require some means of understanding and representing the tasks …". The statement was made in conjunction with traditional systems from the domain of CSCW (Computer Supported Cooperative Work). It seems to be true for Smart Environments as well which are specific CSCW systems.

Different teams have accomplished research aiming to study prototypes of environments for assistive systems in smart environments. Often, elderly people or children are in the focus of such research (see e.g. [3], [23]). Like in our graduate school MuSAMA (Multimodal Smart Appliance Ensembles for Mobile Applications), meeting rooms are also sometimes the embracing environments for such applications. Occasionally the meeting scenario is also used for learning aspects.

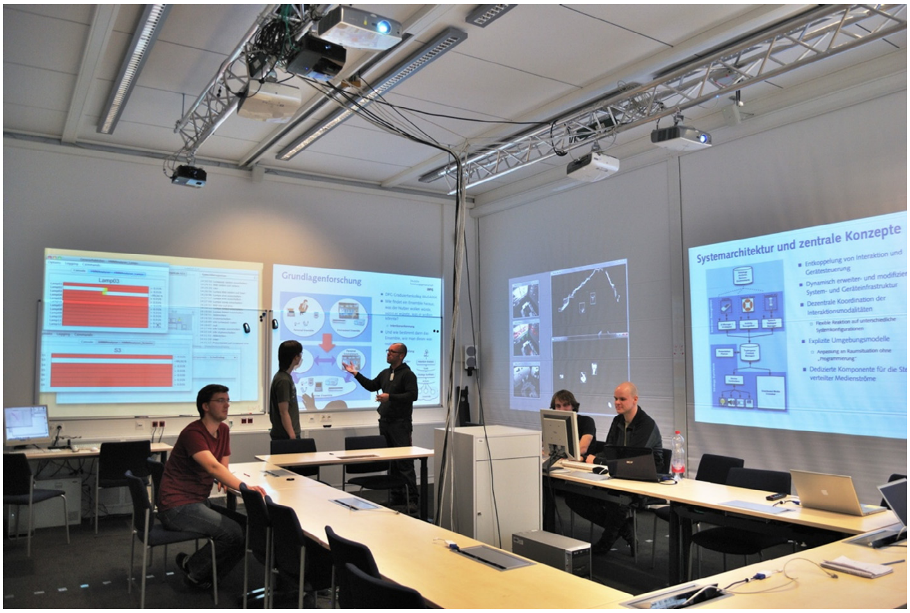


Fig. 1. Smart meeting room at our university

The paper is structured in such a way that we provide a discussion of models for smart environments.

2 Models for Smart Environments

There are sometimes controversial discussions about the roles of models in smart environments. Some approaches follow the idea of artificial intelligence, where certain basic ideas are specified as rules, neuronal nets or Bayesian Networks (see e.g. 3]). Support is provided based on training data or a combination of rules during runtime.

We follow a different approach that is founded on experiences in software engineering [7]. According to these experiences design of interactive systems is often helpful [6]. In this way we strive to build design models for smart environments as well [9]. We do not state that this approach is the only gateway for designing models which are compatible with smart environments. However, we believe that this approach is promising and should be followed. That was the reason for designing a language that allows to specify applications in smart meeting rooms. The language is called CTML (collaborative task modelling language). It consists of task models for all stakeholders, a task model for the team, a special model for the environment and models for devices. The following figure provides a visual representation of these models.

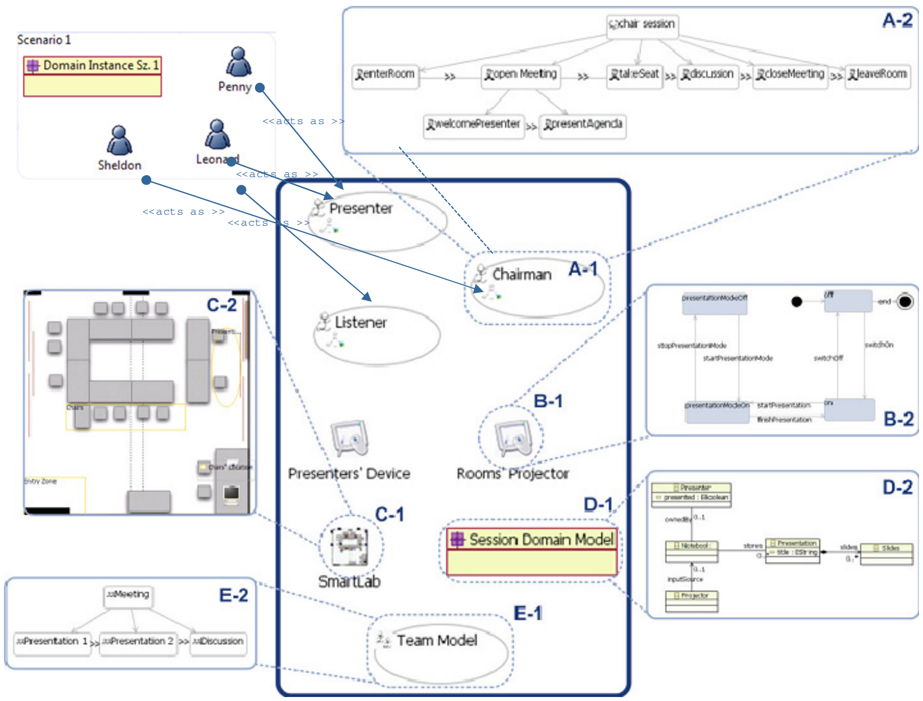


Fig. 2. Schematic representation of models for smart meeting rooms

Most important elements for providing support are task models. They specify tasks for different roles like *presenter*, *listener* or *chairman*. However, there is also a task model for the whole team. It describes the coordination of tasks among the different stakeholders. It is very much related to the sequence in which the meeting has to proceed. Additionally, there is a model for the smart lab that represents the special characteristics of the room and particular zones like the “presentation area”.

These modelling entities are shown in the inner circle of Fig. 2 and post fixed with “-1”. Models outside of the inner circle are specifications of the corresponding entities and post fixed width “-2”.

At the upper left corner of Fig. 2 one can see a scenario specification. Users are assigned to roles. A given role determines the actions a user can perform. In CTML task models are specified in a CTT [13]-like notation where a tree structure represents hierarchically arranged tasks. The leafs of the tree (non refined tasks) are called atomic tasks or actions. Tasks on the same level of abstraction are connected by temporal operators defining the temporal order of task execution.

In the lab different zones are specified. They allow the triggering of specific tasks. If a potential presenter enters the presentation area it can e.g. be assumed that he/she will now have the role of a presenter and the corresponding task model is assigned.

In addition to the general domain model in form of a class diagram there are state chart models for all devices.

The interconnection of the different models is specified by OCL-like expressions. Such expressions can check preconditions before activating a task or actively performing actions as post conditions (details can be found in [24]).

It can be stated that a precondition is needed for the task *OpenDiscussion* of a *chairman*. He can open a discussion only if all presenters have finished their presentations beforehand. This precondition can be expressed by `Presenter.allInstances.EndPresentation` (All people that play the role *presenter* performed the task *EndPresentation.*).

After a *chairman* has announced the end of the discussion all notebooks in the room have to be switched off. This can be specified by the expression `Notebook.allInstances.switchOff`.

More details about CTML can be found in [9] and [24].

3 Prototyping

It was already mentioned in the introduction that prototyping seems to play an important role in “low cost evaluation” of interactive systems. Because of the huge amount of work needed for installing tons of sensors and a lot of software this is especially true for smart environments. Following our modelling approach it is possible to distinguish between the following:

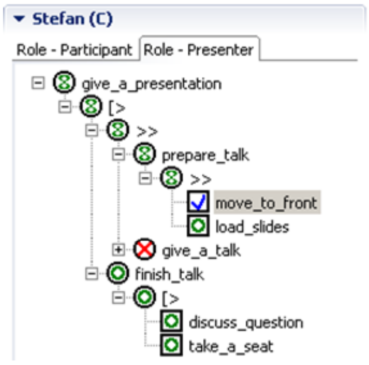
- Simple abstract prototyping
- Complex abstract prototyping
- Prototyping in a virtual environment

Details will be discussed within the following paragraphs.

3.1 Simple Abstract Prototyping

Prototyping in this very abstract sense consists of animating the models for roles, team and devices independently. This might be especially important for task models. Fig. 3 presents an animated task model for the role presenter. The corresponding person entered the presentation area (“move to the front” is already finished). At this state slides can be loaded, discussion can be performed or the presenter can take a seat without presenting anything.

During evaluation of the model it can be discussed whether it is really necessary to load slides before giving a talk or whether discussion is possible without a given talk. Additionally, there might be other activities of interest that were not specified.



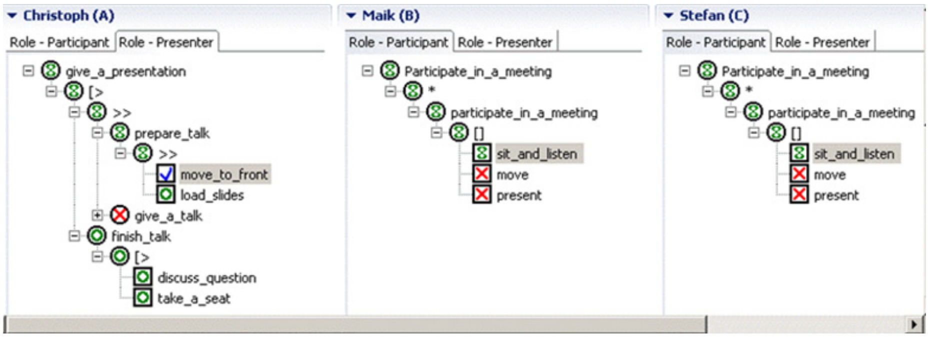
```
▾ Stefan (C)
Role - Participant | Role - Presenter
⊟ give_a_presentation
  ⊟ [>
    ⊟ >>
      ⊟ prepare_talk
        ⊟ >>
          ✓ move_to_front
            load_slides
      ⊞ give_a_talk
    ⊟ finish_talk
      ⊟ [>
          discuss_question
          take_a_seat
```

Fig. 3. Animated task model for role presenter

3.2 Complex Abstract Prototyping

Animating models separately gives a first impression of the behaviour of stakeholders and devices. However, dependencies between different models remain unconsidered. Therefore, animation of all models has to be done in parallel. Additionally, preconditions and post-conditions (in our case the OCL-like expressions) have to be executed. An appropriate interpreter is needed for that.

In the discussed way it can be checked whether a certain task can be executed if a specific device is in the state “off”. Additionally, the consequences of performing a task can be checked by looking at changes in other models. Fig. 4 gives an overview of different users’ respective task models which are animated at the same time.



```
▾ Christoph (A)
Role - Participant | Role - Presenter
⊟ give_a_presentation
  ⊟ [>
    ⊟ >>
      ⊟ prepare_talk
        ⊟ >>
          ✓ move_to_front
            load_slides
      ⊞ give_a_talk
    ⊟ finish_talk
      ⊟ [>
          discuss_question
          take_a_seat

▾ Maik (B)
Role - Participant | Role - Presenter
⊟ Participate_in_a_meeting
  ⊟ *
    ⊟ participate_in_a_meeting
      ⊟ []
          sit_and_listen
          move
          present

▾ Stefan (C)
Role - Participant | Role - Presenter
⊟ Participate_in_a_meeting
  ⊟ *
    ⊟ participate_in_a_meeting
      ⊟ []
          sit_and_listen
          move
          present
```

Fig. 4. Different animated models

One can see in Fig. 4 the animated task models for three persons. They all can play the role of a participant and a presenter. Currently Christoph is the presenter. He just moved to the front and the other persons are sitting and listening. At this stage it can e.g. be checked how the models interact and whether it is possible to have two or three presenters.

This kind of evaluation is still abstract. It would be good to have an impression of the situation in the environment. That was the reason for implementing a 2D virtual environment that is called ViSE.

3.3 Prototyping in a Virtual Environment

The already discussed kinds of abstract prototyping are very helpful. However, they require a direct interaction with the models. It is very obvious that having a virtual environment that allows to interact with actors and devices would significantly improve the evaluation process. In this case the later situation in the real world can already be simulated.

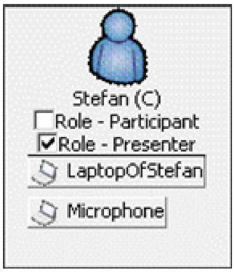


Fig. 5. Representation of persons in the virtual environment

Fig. 5 illustrates the way persons are represented in the virtual environment. It is possible to interactively assign roles to such persons. However, this can also be done automatically using post-conditions. Currently Stefan plays the role of a presenter. Additionally, it is possible to assign devices to such persons. In the given example, Stefan carries a laptop and has a microphone. The icon representing the person "Stefan" can be created and moved in the virtual space of the environment.

A specific situation for our meeting room can be seen in Fig. 6. It represents a case in the virtual environment that fits to the state of the task models that are presented in Fig. 4. Christoph is in the presentation area which is located in front of the whiteboard. The whiteboard is symbolised by the line to the right of Fig. 6. Maik and Stefan took a seat and play currently the role of a participant. While Maik is carrying a PDA Stefan has currently a laptop. Christoph has a laptop and a microphone.

While interacting with the virtual environment the animations of the corresponding models are updated. In this way it can be checked how the state of the environment changes while interacting with icons representing persons or devices. Additionally, the execution of tasks can be performed by directly interacting with the models. For moving to the front the corresponding icon has only to be moved to the respective position. However, to load slides the execution of the corresponding task has to be explicitly triggered.

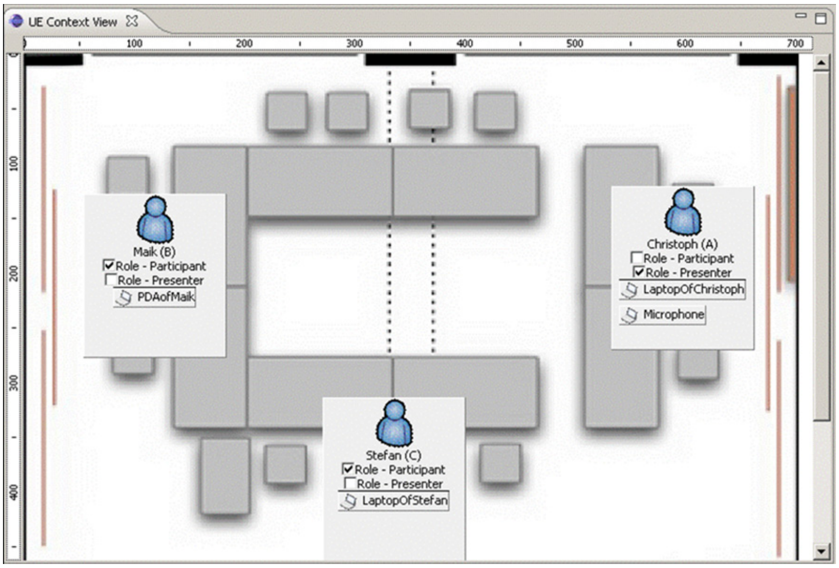


Fig. 6. Virtual meeting room with three persons

This kind of evaluation is not only helpful as long as the real system does not exist. Indeed it supports evaluation very well. For classical interactive systems it is common sense that expert evaluation should be performed before user tests are executed. However, how can expert evaluation be performed in a smart meeting room with three acting people? This requires involving three people who have to follow the existing scripts and play the exact role assigned. This is very costly and time consuming.

In its traditional sense, expert evaluation is only possible with virtual environments. In such environments an expert can control several persons and thus check the behaviour of a specified system. This is a significant advantage guaranteed by means of expert evaluation giving it preference over tests that have to be performed by several people in the real world environment.

4 Discussion and Related Work

In the previous paragraphs we argued for a model-based approach for smart environments based on examples for smart meeting rooms. From our point of view, a detailed analysis of tasks that are performed by potential users helps to understand the domain. Design models for activities in form of task models are beneficial. They are a good representation to think about supporting users. They allow explicit design for the support offered in smart environments.

Using the discussed models allows prototyping at different development stages. Singular models can be animated first. Later they and their dependencies can be evaluated by animation. Other authors argue in a similar way. Some use task models even as a control mechanism for runtime (see e.g. [2] or [19]).

More user friendly prototyping can be performed having a virtual environment available. Such an environment allows expert evaluation because experts are able to control several virtual persons at the same time.

Our experiments were based on an own virtual 2D environment. In conjunction with the APEX-project [20, 21] open simulator [12] is suggested as a 3D virtual environment for prototyping of smart environments. Models are specified as Petri-net specifications. The APEX-framework allows the synchronisation of animated Petri-nets with the virtual environment of open simulator.

Prototyping in 3D has the benefits of providing a more realistic environment. It also provides more realistic interaction techniques between agents in the environment and the smart environment. To certain extent, gestures can be supported in 3D whereas they have to be abstracted in 2D.

However, there are also drawbacks. Creating a virtual 3D environment for a specific application is much more complicated than to provide a 2D environment. That depends of course on the number of available 3D objects. After a while the construction of new environments might decrease a lot. Additionally, the 3D representation asks for more detailed models. This is due to the existence of post-conditions requiring more precise state information.

It seems to be interesting to have both options available. For some cases it might be enough to have a 2D representation available whereas for others prototyping in 3D might be the only way to provide the right experience. There seems to be a need for metrics that help to estimate the necessary effort for both approaches. Additionally, it would be helpful that environments provide features which allow testers to specify parallel activities of agents in a script form. It is even very hard to do this interactively.

For our virtual environment ViSE as well as the real environment, there exists a language for a special kind of middleware. All devices can be controlled via this feature. It is not only a communication channel between all devices but also between the real world and the virtual world. Real meetings can thus be visualised through meetings in the virtual environment. In contrast to videos, this kind of replay allows the analysis of meetings in an anonymous way. Additionally, events from the virtual environment and the real environment can be mixed. A costly new sensor can only be installed in the virtual environment but can give feedback to events from the real world environment. In this way it can be checked whether it makes sense to y the sensor. It will be a challenge for the future to combine a virtual environment with several real world environments. Two real meeting rooms (like two video conference rooms) could be supported by one virtual room.

Finally, more research has to be conducted in order to make it feasible to present the system state by animated models in a convenient human understandable way. A potential solution is actually providing information with different levels of detail that can be interactively changed.

5 Summary and Outlook

A model-based approach seems not only to be a good idea for general interactive systems but also for smart environments. It allows a precise requirements specification and an appropriate design. Models provide the opportunity for evaluations at early stages of software development with relatively low costs. Animations provide excellent chances for communicating ideas of envisaged behaviour for assistive systems with users. Users can already interact with the system under development in an abstract way. This can be done through animated models, virtual 2D environments or virtual 3D systems. Within the paper advantages and disadvantages related to usability evaluation were discussed. Some challenges were identified that raise questions seeking answers in the near future. These challenges are

- Providing metrics for costs of modelling 2D versus 3D.
- Presentation of the system state using animated models in a suitable human understandable way.
- Providing a feature for precise specification for the performance of parallel activities of various agents in the virtual environment.
- Combination of a virtual environment with several real environments.

References

1. APEX: `http://twiki.di.uminho.pt/twiki/bin/view/Research/APEX/WebHome` (last accessed January 3, 2014)
2. Blumendorf, M.: Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces. Dissertation, TU 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. 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)
5. 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/`
6. 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)
7. Forbrig, P., Dittmar, A., Brüning, J., Wurdel, M.: Making Task Modeling Suitable for Stakeholder Driven Workflow specifications. In: Stephanidis, C. (ed.) Universal Access in HCI, Part I, HCII 2011. LNCS, vol. 6765, pp. 51–60. Springer, Heidelberg (2011)
8. Forbrig, P., Wurdel, M., Zaki, M.: 2012: The roles of models and patterns in smart environments. In: EICS Workshop, Copenhagen (2012)
9. Forbrig, P.: 2012: Interactions in Smart Environments and the Importance of Modelling. Romanian Journal of Human - Computer Interaction 5, 1–12 (2012); Special issue: Human Computer Interaction (2012) ISSN 1843-4460, `http://rochi.utcluj.ro/rrioc/en/rochi2012.html`
10. 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)

11. Johnson, P., Wilson, S., Markopoulos, P., Pycock, J.: ADEPT: Advanced Design Environment for Prototyping with Task Models. In: Proceedings of the INTERACT 1993 and CHI 1993 Conference on Human Factors in Computing Systems (CHI 1993), p. 56. ACM, New York (1993)
12. OpenSimulator: `http://opensimulator.org/wiki/Main_Page` (last accessed January 3, 2014)
13. Paterno, F., Meniconi, C.: ConcurTaskTrees: A diagrammatic Notation for Specifying Task Models. In: INTERACT 1997, IFIP TC13, pp. 362–369 (1997)
14. Paterno, F., Ballardin, G.: Model-aided remote usability evaluation. In: Sasse, A., Johnson, C. (eds.) Proceedings of the IFIP TC13 Seventh International Conference on Human-Computer Interaction, pp. 434–442. IOS Press, Amsterdam (1999)
15. Pinelle, D., Gutwin, C., Greenberg, S.: Task Analysis for Groupware Usability Evaluation: Modeling Shared-Workspace Tasks with the Mechanics of Collaboration. ACM TOCHI 10(4) (2003)
16. Propp, S., Forbrig, P.: ViSE – A Virtual Smart Environment for Usability Evaluation. In: Bernhaupt, R., Forbrig, P., Gulliksen, J., Lárusdótti, M. (eds.) HCSE 2010. LNCS, vol. 6409, pp. 38–45. Springer, Heidelberg (2010)
17. Propp, S., Buchholz, G., Forbrig, P.: Integration of usability evaluation and model-based software development. Advances in Engineering Software 40(12), 1223–1230 (2009)
18. 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.pd`
19. Roscher, D., Lehmann, G., Blumendorf, M., Albayrak, S.: Design and Implementation of Meta User Interfaces for Interaction in Smart Environments. In: [5]
20. Silva, J.L., Campos, J., Harrison, M.: Formal analysis of ubiquitous computing environments through the APEX framework. In: Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2012), pp. 131–140 (2012)
21. Silva, J.L., Ribeiro, O., Fernandes, J.M., Campos, J.C., Harrison, M.D.: (2010)
22. Silva, J.L., Ribeiro, Ó.R., Fernandes, J.M., Campos, J.C., Harrison, M.D.: The APEX framework: prototyping of ubiquitous environments based on Petrinets. In: Bernhaupt, R., Forbrig, P., Gulliksen, J., Lárusdótti, M. (eds.) HCSE 2010. LNCS, vol. 6409, pp. 6–21. Springer, Heidelberg (2010)
23. 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)
24. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and Task Modeling for Collaborative Environments. Journal of Universal Computer Science 14 (2008); Special Issue on Human-Computer Interaction
25. 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, Feburary 19-21 (2013)
26. Zaki, M., Wurdel, M., Forbrig, P.: Pattern Driven Task Model Refinement. In: Abraham, A., Corchado, J.M., Rodriguez-Gonzalez, S., De Paz Santana, J.F. (eds.) International Symposium on Distributed Computing and Artificial Intelligence, DCAI 2011, Salamanca, Spain, April 6-8. Advances in Soft Computing, vol. 91, pp. 249–256 (2011) ISBN = 978-3-642-19933-2