From Collaborative Scenario Recording to Smart Room Assistance Models

Gregor Buchholz(✉) and Peter Forbrig

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

Abstract. There is still much to be done in implementing assistance systems for Intelligent Environments. Different approaches exist that aim at providing the user with useful and pleasant functionality. One group of methods uses behavioral models to derive supportive actions from the observation by sensors. This is a promising approach but creating such models is a laborious and error-prone task. Examples of the behavior of persons in intelligent environments and their interactions with the devices are a starting point for the (partial) generation of such models. In this paper we present an approach to record user behavior without the need of real users performing in the real environment. As a special thematic priority we will focus on the preparation phase of collaborative scenario recording and the used notation. Additionally, the paper will explain the generation of models from the recorded traces.

Keywords: Intelligent environments · Ubiquitous computing · Task models

1 Introduction

Numerous engineering techniques for interactive systems use the application of models as their basis. Among those, task model based approaches focus on the tasks users want to accomplish while interacting with the system. One main characteristic of such approaches is the application of models not only in the requirements analysis and development phases but also during run time. Ambient Intelligent Systems like Smart Environments can profit from these models, too. Such systems need to be aware of the users' movements, positions, their interaction with devices, the states of those devices and so on. With the help of models, the system tries to anticipate appropriate actions to support the users.

The interplay of different things around us like wearable computers and stationary devices is known as Ubiquitous Computing and considered to be part of our near future [10]. Environmental monitoring through sensors of all kinds enables them to gather information about many aspects including, beside their own current state (e.g. "position"), the amount, states and history of neighboring devices. In addition, the

The work of the first author is supported by DFG graduate school 1424 *Multimodal Smart Appliance Ensembles for Mobile Applications* (MuSAMA) at the University of Rostock, Germany

DOI: 10.1007/978-3-319-20804-6_24

availability of information about the context (e.g. "temperature" and "brightness") and persons acting therein (e.g. "number", "position", movement data) opens up perspectives for new software systems. The vision tied to networked devices of diverse types is that of an invisible but omnipresent computer that assists the users (or at least offers them support) in their activities based on its knowledge about the context and the current situation. The goal is to provide the best assistance possible with the currently connected devices.

In this paper we focus on scenarios in closed rooms with different kinds of sensors (e.g. real-time location systems like UbiSense[1], cameras, radio tomography), stationary actuators (lights, projectors, screens) and personal devices like users' laptops that join and leave the ensemble dynamically [9] ("Smart Meeting Rooms"). Depending on the number and type of devices that are available at a specific time the system can offer more or less functionality as each single device as well as device combinations contribute some functions.

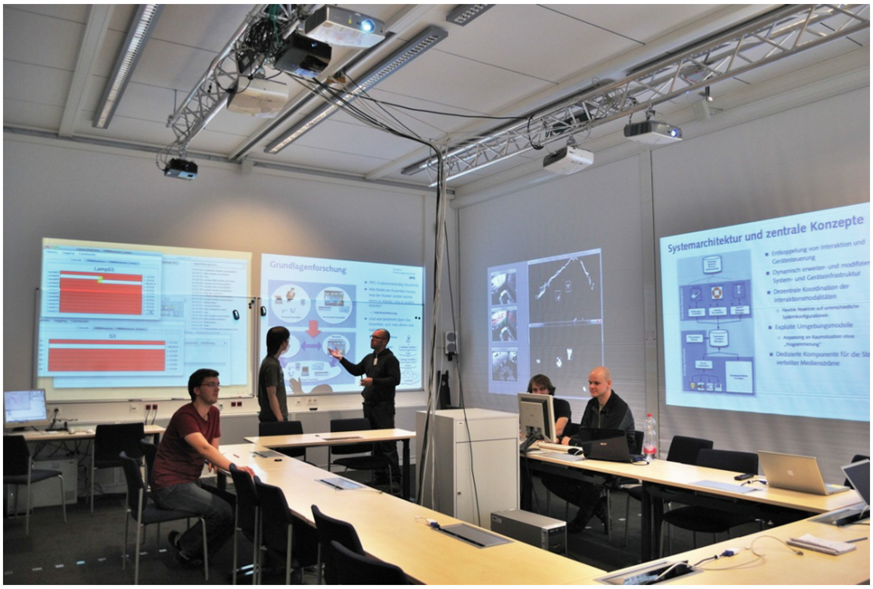

Fig. 1. Smart appliance lab

Figure 1 shows an example application of a Smart Appliance Lab: The room, equipped with presentation devices and sensors, distributes slides and other information over the available screens in a way that every person's needs, interests and preferences are matched. Persons acting as a presenter play a special role in this use case: Depending on their position within the presentation zones the content is rearranged. Other use cases beyond the depicted presentation scenario, e.g. working in two or more

[1] UbiSense real-time location systems: http://www.ubisense.net/.

groups, discussions or collaborative activities (parts of the content are contributed by different users) have to be supported, too.

1.1 Providing Assistance in Intelligent Rooms

Several competing methodological approaches are dedicated to provide assistance in Smart Environments. One group of methods uses concepts of artificial intelligence in order to adapt to changing behavior of users, to infer intentions from the observed activities and to provide assistance in reaching a predicted target state, defined as a set of variable assignments describing aspects of the environment [1, 6]. The increasing diversity of user activities and growing number of users, however, are accompanied with a higher demand for training data in order to ensure the recognition of user activities with sufficient accuracy. Moreover, during run time the calculations required to identify possible successor states and their probabilities place even greater demands on processing power and memory which, while reducible by certain state space compression methods, nevertheless are substantial.

A second group of approaches attaches much more importance to the mainly manual design of behavior models. Intertwined activities of different users fulfilling different roles can be described in separate models. Interdependencies between the models are described by special notations [14]. The application of such models significantly reduces the search space for possible pro-active assistance, lowering the resource requirements and promoting and fostering more targeted user assistance.

The models' adequacy and appropriateness is of vital importance for the system's utility value and thus the acceptance level among users. Model based development methods take account of this core requirement by giving much weight to the role of models during all development phases from requirements analysis and system design through to usability testing. The creation and discussion of models of varying degree of formality are a main concern during the discourse between user and analyst. Nevertheless, with increasing level of detail and decreasing level of abstraction (which is the most laborious part of such hierarchical models as the task tree progressively broadens downwards) more and more use is made of recorded scenario protocols [12] whose obtaining is an elaborating and expensively task too.

1.2 Model Construction from Scenarios

As shown in [4], the construction of task models as formalization of requirements analysis results can be combined with a partly automatic generation of model fragments. Two activities directed towards each other are combined: The expert-driven top-down-modeling forms the basic structure of the hierarchical model and a bottom up-generation completes it and derives temporal relations from the recorded scenario traces. Thus, the labor-intensive construction of widely branched model trees is considerably simplified and has experienced a noticeable acceleration. Figure 2 gives an overview of the entire process.

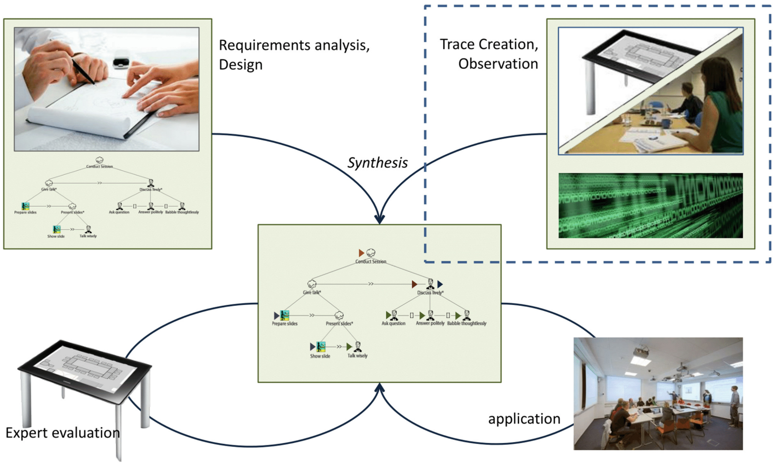

Fig. 2. Model creation process overview

However, the recording of scenario traces in the physical Smart Meeting Room is quite resource intensive. That motivates the development of a more efficient method to generate scenario traces in a virtualized environment. This paper describes an approach to the creation of behavioral models in the context of Smart Meeting Rooms combining the manual modeling of user activities on more abstract levels with the semi-automated generating of detailed model fragments. It especially highlights the recording of example traces in a simulated, virtual Smart Environment and the planning of those scenarios.

2 Related Work

Several approaches address the scenario-based formalization of human behavior, many of which fall into the field of process mining. A comprehensive collection of such algorithms is implemented in the ProM framework[2] [13]. Some of the numerous applications beyond the "basic" mining of processes are the identification of bottlenecks, verification of business rules, and creation of social network graphs. Importing from a number of different sources is possible and many plug-ins for import formats and functionality are available. In common with the majority of process mining approaches the focus lies in the extraction, visualization and optimization (in terms of resource usage) of processes. Here, it is taken less concern for requirements like human readability and understandability, support for a strong orientation towards a hierarchy reflecting different levels of abstractions from the viewpoint of a user and the suitability of the resulting models for discussions between modeling experts and untrained persons.

[2] http://www.promtools.org/prom6.

Reference [11] presents an ontology-based approach to define and recognize event patterns in sensor streams of Smart Environments using an EL^{++} description logic, see [2]. Their results regarding recognition rate, run-time performance and "ease of design" seem to be promising, although the system does not integrate direct interaction between users and the system's underlying models.

In [5] a method for automatic work flow induction is presented that is based on First-Order Logic representations and learns behavior models in Smart Environments from scratch by increasingly refining the work flow model (starting with an empty model). It uses two predicates (*activity* to assign tasks to steps and *next* to describe a following relation between two steps) to describe cases and two predicates to describe work flows. During learning, pre- and post-conditions are generated, too. Currently, it does not include distinct consideration of multi-user scenarios. The presented approach by itself in fact does allow for more than one person's "sensor trace" to be considered, but parallel and interlaced activities are not addressed explicitly.

An example of a system generating task models is *ActionStream*, introduced in [8], that records user activities for a long period of time while all interactions are interpreted as terminals of a grammar. By continuously adapting the grammar's production rules, *ActionStream* learns a formal model of the user's behavior. Such approaches are likely to produce quite precise models successfully covering the learned scenarios, lacking however a semantic meaning of the non-terminals. The resulting models, as "correct" as they may be, are of limited use in the communication between stakeholders during the development phase of a system.

3 Creation of Scenario Traces

Deriving task models from the traces is the purpose of capturing example scenarios. User behavior in this context covers all aspects of movements and interactions that are detectable by the environments sensors: walking, standing, sitting, talking, pointing, bringing devices into the ensemble, connecting devices (e.g. a laptop and a projector), using devices, and more. The traces describe a sequence of such events as captured and extracted from the sensor data streams. Recording a scenario demands for a thorough planning in order to create a set of examples that cover main use cases as well as exceptions and variations. In the following section we describe how to prepare efficient play-throughs.

3.1 Planning the Scenario Recording

For each use case a set of scenarios is developed that describes possible variations of actions sequences. In preparation for each cooperative recording of a scenario a graphical description of the planned interaction sequence is created. The purpose of these paper plans is to serve as handling instruction during the recording of scenarios. Such a plan should represent all important aspects of the envisioned users' behavior in the Smart Environment during the considered situation. To describe such a scenario we use the *ActionSketch* notation with the extended adaption for interactive environments [3].

The basic set of *ActionSketch* notation elements is designed for WIMP and Multi-touch interaction. The principle remains the same: Each frame depicts the initial state (black), user actions (green), and system actions (orange). While the first frame should show the complete initial state of the system all of the following frames only show the changes between frames. These rules (i.e. color coding and the notation of changes) also apply for the sketching of behavior in Interactive Environments. Here, the set of interaction icons is further extended by some symbols: a circle inside an ellipse to represent a person's position and representations from architectural sketches for common objects like tables and walls. In order to distinguish between different persons in the room, we changed the notation of persons to "a letter inside an ellipse". Additionally, we use a simple notation of timestamps: Each frame is labeled with the time difference in respect to the first frame with the initial state.

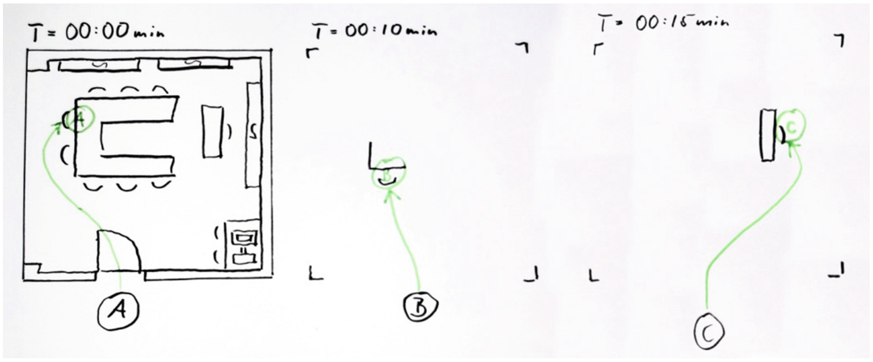

Fig. 3. ActionSketch example sequence

In Fig. 3 a simple example for the beginning of a frame sequence describing a multi user scenario is given. The first frame (timestamp 00:00) shows the room with the projector screens ("S"), the tables, chairs, the door, and a workstation in the lower right corner. At the beginning, Person "A" is outside the room. Beginning at the timestamp 00:00 that person moves to a chair. This arrow is drawn in green to mark it as user action, as movements in a Smart Room with sensors can be interpreted as interaction. The user action does not cause any system actions, so there is nothing to be drawn in orange. The second frame (timestamp 00:10) only shows the differences to the last state of the actions in the preceding frame: Now, person "B" moves from outside the room to another chair. That specific chair and a little part of the table are repeated again for easier orientation. In the third frame (timestamp 00:15) person "C" enters the room and moves to the presenter table. Again, only some little parts of the room are drawn for orientation purposes. More frames are added for a complete description of a scenario which can be used as guidance during the simulation.

In practical use, we make certain simplifications and modifications to reduce the number of frames and increase legibility. Two exemplary adaptations are described in the following:

- If the actions of two consecutive frames can be drawn without overlapping, we combine them to one picture. The action of the original first frame is annotated with a roman 1 (“I”), the second frame’s action with “II” and the timestamps are labeled as “T_I” and “T_{II}”.
- In order to distinguish more clearly between actions of different users we use colors (besides green and orange) that mark each person in a unique manner. If so, the person “B” is referred to as person “blue”, for example.

Interactions between users and a device in the room can be noted in two ways: One possibility is to draw the device interface “inline”, i.e. the UI is drawn in a dotted rectangle within the room frame. That option is exclusively suitable for very simple interactions like pressing a button. The second option is to switch from the room view to the device’s view. In this case, the first device interaction frame would show the complete interface (like the very first frame of a *ActionSketch* sequence) with only the differences and actions in the following frames. Leaving the device view, the whole room would be shown in the first frame showing the room again.

Please note that our application of *ActionSketch* does not allow the usage of the “OR” construct that can be found in the original specification. The usage of the notation as a description of a scenario as described here is meant to define one specific sequence of actions and not a number of possibilities. The latter would be seen as a use case within the context of our work. On the other hand, constraints like timestamps cannot be seen as absolute certainty, as a simulation by hand is inherently tainted with a certain degree of inaccuracy. So the minimum requirement of a scenario definition is to describe the actions that are performed by the users and their temporal order.

In the following sections we describe the system to simulate the scenarios defined by such frame sequences. The next section discusses the infrastructural setting followed by a characterization of the used hardware and the recording software.

3.2 Protocols from Observations in the Physical Environment

During the usage of the Smart Environment (as depicted in Fig. 1) the middleware Helferlein[3] provides the different application modules with the event streams produced by the sensors. The modules can feed their own events into the common event storage. Every module can use the publish/subscribe-mechanism to get notified as soon as a new event of the specified type is detected.

For the purpose of scenario recording additional modules have been developed. They collect and store the events produced by the table computer while moving the tagged objects over the room plan (See paragraph “software” in Sect. 3.3 for more details). The extraction of semantically meaningful events from the sensor streams can

[3] The middleware Helferlein: https://code.google.com/p/helferlein/.

be achieved in different ways. One way is to use sliding window techniques [7]. Such an algorithm is implemented as a module on the middleware. The occurrences of previously defined event patterns in the incoming data streams from different sensors are detected that are characteristic for certain user activities. The definition of these patterns has to be adjusted for different room setups, e.g. for changing sets of sensors.

The output of that module is a new event stream: the “basic action stream” that contains user-related events abstracted from sensor data. Such higher level events can be used as execution trigger for the models during run time. To enable an efficient binding between events and the models, the events are further aggregated in two ways.

1. Spatial Aggregation - As not every single change of position of every user in the room has a direct influence on the model execution progress we introduce a more abstract event category. Therefore, previously defined zones in the room with distinct meanings serve as references, e.g. presentation zones in front of the displays, the entrance area around the door, the seating accommodations and so on. The basic movement events are no longer the only possible triggers for the model execution but events raised when users enter or leave such a zone can be used, too.
2. Temporal Aggregation - Movement information inside and outside the defined zones are not always interesting to focus on in the highest level of detail. Therefore, key frames are calculated that represent users’ movements as a discrete set of data points.

Both forms of aggregation produce new events that are made accessible by other modules through the middleware’s publish/subscribe-mechanism.

The recording of scenarios in the Smart Environment always entails a lot of work. In the following, a method is described to significantly reduce the costs of producing such protocols.

3.3 Generation of Protocols in the Virtualized Environment

Hardware. For the implementation of a system for producing protocols exemplifying use cases a Samsung SUR40 has been selected. The SUR40 is a computer in the form of a table running *Microsoft PixelSense 4* (formerly known as *Microsoft Surface*). Its 40” display features multi-touch interaction and object recognition by attaching Identity Tags to things. Figure 4 shows (a) a SUR40 table computer and (b) two cubes with

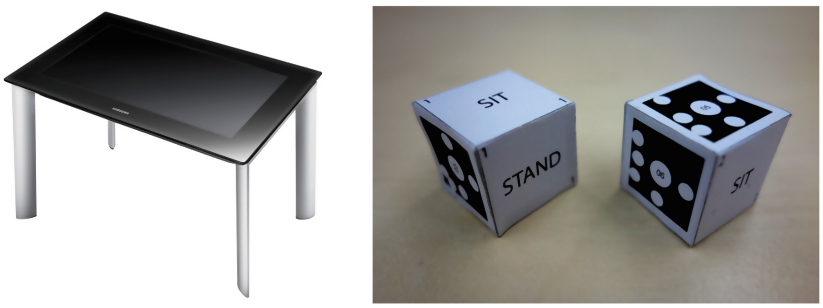

Fig. 4. (a) Samsung SUR40 table computer, (b) Cubes with identity tags

three Identity Tags each. During the recording of a scenario each cube represents the position of one person within the room. Three sides of each cube are labeled as SIT, STAND, and INTERACT, respectively. The opposite sides have a corresponding Identity Tag printed on. The recognition of a tag is interpreted as the label on the top side of the cube. Of course, each cube has its own set of tags. A position change in the state STAND is interpreted as normal walking and via INTERACT the interaction with a device nearby can be initiated.

Software. The middleware of the Smart Environment (*Helferlein*) is complemented by modules encapsulating the room's devices and sensors. A common interface enables other modules to request and use the public properties and methods of each object. For a large part of the devices emulations have been implemented providing an idealized (i.e. fault free) virtual device functionality via the same interface as the corresponding real device. Each module encapsulating a device or a sensor can be requested to deliver a GUI specification with property visualizations and method activators.

The recording of scenario protocols as described here uses these modules to emulate the devices in the Smart Environment and Surface2TUIO and TUIO4j[4] for the processing of multi-touch and object recognition events.

A schematic view of the room's setup is the core element of the GUI visualizing the furnishings and devices. The defined zones are highlighted in different colors. Personal devices can be integrated as separate objects with their own Identity Tag or as attached to a user.

The GUI features four main elements:

1. Room Plan - The interaction space with a schematic room view is the biggest part of the interface. Visual feedback shows the recognized position of tagged objects. Interaction elements for devices are displayed as soon as the INTERACT function of a person is selected.
2. Property Panel - Here the properties and methods of the currently selected object are visible. In the case of sensors (e.g. brightness sensor or thermometer) the behavior of the emulation object can be controlled. Thus, environmental conditions like strong incidence of light can be integrated into the scenario.
3. Status Panel with Timeline Display - This area shows meta data of the current recording, the timeline and its controls. To successively record parallel activities the user can jump to an earlier point in time and then record a second activity etc.
4. Protocol View - This view replaces the Room Plan and shows the recorded protocols. Here, protocol events can be assigned to tasks in the behavioral model as needed for further processing during model generation. Multiple protocols can be assigned to a use case in preparation of the succeeding steps.

Figure 5 shows a situation during a scenario recording: The defined zones are highlighted in red (A, entrance zone), green (B, presentation zone), and blue (C, seat regions. The current positions of the two simulated persons are circled in yellow. On the right-hand side the Property and Status Panel are visible. Thus, scenario traces for different use cases can be recorded.

[4] TUIO2j: https://code.google.com/p/tuio4j/.

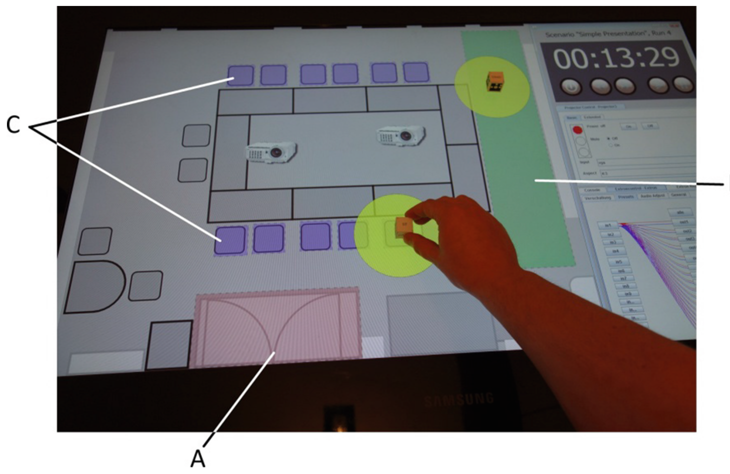

Fig. 5. Recording a Scenario on the touch table

Classification. The system described in this paper aims at the design of user assistance in smart environments at an early development stage. It is a significant advantage that no physical hardware in the room has to be installed. In fact, the specific configuration and layout of the room's equipment may be even unclear or may be subject to change and it is not the objective of this system to explore and compare different room designs, although that would be possible to a certain degree, too. It is for this reason that we decided to use a quite simple 2D layout at a certain abstraction level as a room representation. Of course it would be possible to use a more detailed graphical (e.g. 3D) design but that would disguise the development status of the system and anticipate decisions that have not been taken yet. However, our tool can be complimented by a 3D tool in the later development phases.

3.4 Limits of the Protocol Creation

Not all aspects of usage observable in the real environment can be included in the recording on a table computer. This is especially true for unexpected behavior and events like the opening of a window, leaving the room for a short time, interruptions through external factors and some more. Such situations have to be considered during the planning of a scenario recording and during the later steps of model generation and evaluation.

4 Conclusion and Future Work

In this paper we presented an approach to plan and produce scenario traces as examples for the usage of Intelligent Environments. The application of the *ActionSketch* notation allows an efficient and intuitive way to prepare and describe the sequence of actions that form a scenario. Multiple scenarios demonstrate different possible variations of use cases. They can be used to support the modeling of task trees in such a way, that the essential structure of the models is created manually based on the collected domain knowledge and the scenario traces are used to extract more detailed information about the process. In this context, the use of devices with multi-touch and object recognition capabilities allows an intuitive scenario recording by experts and users.

Multiple users can interact in the scenario and assume roles dynamically. The application of the proposed system can be summarized in the following steps:

1. Identify requirements with conventional methods (interviews, questionnaires etc.).
2. Formalize knowledge from step 1 as task model as far as possible.
3. Define scenarios by developing stories using the *ActionSketch* notation.
4. Use the recording software to produce scenario traces for a specific use case.
5. Combine the result of step 2 with generated model fragments using results of step 4 (semi-automatic model completion).
6. Check generated model by simulation and revise, if necessary.

The resulting models can then be used as the basic for the environment's supportive actions. So far, the tool chain from scenario recording (implemented in the Helferlein framework) and the model construction implementation are only loosely coupled. In order to support a fluent transition between recording, modeling and generating, the different parts should be integrated into the framework. Beyond this, expert evaluation of the task models in the virtual environment will be possible by integrating the run time environment for the instantiating and processing of task models into the system. Thus, several phases of the assistance development process for Smart Environments will be able to benefit from the virtualization of the room including devices, sensors, and the users' behavior.

References

1. Armentano, M.G., Amandi, A.A.: Recognition of user intentions for interface agents with variable order Markov models. In: Houben, G.-J., McCalla, G., Pianesi, F., Zancanaro, M. (eds.) UMAP 2009. LNCS, vol. 5535, pp. 173–184. Springer, Heidelberg (2009)
2. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proceedings of the IJCAI 2005, pp. 364–369. Morgan Kaufmann, San Francisco (2005)
3. Barros, G.: Extending ActionSketch for new interaction styles: gestural interfaces and interactive environments. In: Marcus, A. (ed.) DUXU 2014, Part II. LNCS, vol. 8518, pp. 509–520. Springer, Heidelberg (2014)
4. Buchholz, G., Forbrig, P.: Combining design of models for smart environments with pattern-based extraction. In: Kurosu, M. (ed.) HCI 2014, Part I. LNCS, vol. 8510, pp. 285–294. Springer, Heidelberg (2014)
5. Ferilli, S., De Carolis, B., Redavid, D.: Logic-based incremental process mining in smart environments. In: Ali, M., Bosse, T., Hindriks, K.V., Hoogendoorn, M., Jonker, C.M., Treur, J. (eds.) IEA/AIE 2013. LNCS, vol. 7906, pp. 392–401. Springer, Heidelberg (2013)
6. Kiefer, P.: Mobile intention recognition. In: Kiefer, P. (ed.) Mobile intention recognition, pp. 11–53. Springer, New York (2012)

7. Krämer, J., Seeger, B.: Semantics and implementation of continuous sliding window queries over data streams. ACM Trans. Database Syst. **34**, 4:1–4:49 (2009)
8. Maulsby, D.: Inductive task modeling for user interface customization. In: Proceedings of the IUI 1997, pp. 233–236. ACM, New York (1997)
9. Ramos, C., Marreiros, G., Santos, R., Freitas, C.F.: Smart offices and intelligent decision rooms. In: Nakashima, H., Aghajan, H., Augusto, J.C. (eds.) Handbook of Ambient Intelligence and Smart Environments, pp. 851–880. Springer, New York (2010)
10. Robles, R.J., Kim, T.-h.: Context aware systems, methods and trends in smart home technology. In: Kim, T.-h., Stoica, A., Chang, R.-S. (eds.) Security-Enriched Urban Computing and Smart Grid. Communications in Computer and Information Science, pp. 149–158. Springer, Heidelberg (2010)
11. Scalmato, A., Sgorbissa, A., Zaccaria, R.: Describing and recognizing patterns of events in smart environments with description logic. IEEE Trans. Cybern. **43**, 1882–1897 (2013). IEEE
12. Seyff, N., Maiden, N., Karlsen, K., Lockerbie, J., Grünbacher, P., Graf, F., Ncube, C.: Exploring how to use scenarios to discover requirements. Requirements Eng. **14**(2), 91–111 (2009). Springer, Heidelberg
13. van der Aalst, W.M.: Process mining in the large: a tutorial. In: Zimányi, E. (ed.) eBISS 2013. LNBIP, vol. 172, pp. 33–76. Springer, Heidelberg (2014)
14. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and task modeling for collaborative environments. J. Univ. Comput. Sci. **14**(19), 3188–3201 (2008). (Special Issue on Human-Computer Interaction)