Combining Design of Models for Smart Environments with Pattern-Based Extraction

Gregor Buchholz and Peter Forbrig

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

Abstract. There are two different types of approaches for smart environments. The first group provides an infrastructure that contains mechanisms from artificial intelligence that allow to adapt to certain behavior of users and to support them by performing their tasks. These approaches work fine if the conditions in the environment are not experiencing too many changes. However, when different types of activities have to be supported and participants change a lot there is the problem of getting enough training data to recognize the users' activities with sufficient reliability. In such cases, designing support by providing models for activities of participating users seems to be a solution. Thus, mechanisms from artificial intelligence can be supported by reducing the search space for possible actions.

Designing of activity models can be performed by employing the top-down approach through predefined generic patterns or alternatively the bottom-up mechanism by looking at traces of performed activities (scenarios). Again patterns play an important role as they allow the identification of important parts of traces that lead to parts of models. The identification of such trace sections can be done almost automatically. The mapping to parts of models however, has to be done in an interactive way. Human decisions are necessary to provide good models. Different strategies can be supported by tools in order to make decisions within the models ranging from abstract levels down to the most detailed level.

This paper will provide a discussion of the outlined approach.

Keywords: task models, smart environment, model generation.

1 Introduction

The application of models is a well-established principle of numerous engineering techniques for designing interactive systems. Task model based development approaches highlight on the tasks users want to accomplish while using the system. Thus, not only the requirements analysis and development process are having a strong focus on the users' tasks but also the running system can extensively profit from these models. This is particularly true for Ambient Intelligence Systems like Smart Environments since the utility value and thus the acceptance

M. Kurosu (Ed.): Human-Computer Interaction, Part I, HCII 2014, LNCS 8510, pp. 285–294, 2014.

level among users are dependent to a decisive extent on the usefulness and appropriateness of the system's support. For this purpose, the Smart Environment needs to be aware of single users' movements, positions, etc. among a lot of data describing the current situation, and also has to consider the connections and mutual dependencies between different users fulfilling different (and dynamically changing) roles in order to achieve a common objective. As the variability in human behavior is very high, and potentially unlimited, a real *understanding* of such a complex situation is beyond the possibilities of any computer system. Thus, it might be helpful to provide the system with certain information about what activities to be supported. Applying this concept means to select ("load") a use case description before using the room assistance. Instantiating a use case may involve some customization, for instance the amount of talks in a conference session or the number of groups working temporarily separated in a workshop. Beyond this, an instantiation can include the combination of predefined parts of scenarios, tailored to the current needs of the user(s). Obviously, such adaption operations must be simple to use since the user should not have to delve that deep into the technical details of the room's *intelligence*. Once started, the Smart Environment continuously tries to recognize the given situation and, as soon as the recognition reaches a reasonable degree of certainty, it should give (or propose) assistance as modeled in the previously designed model. Bringing together the room's hardware and high-level models requires some additional effort: The lack of any semantics in the sensor data suggests the introduction of an intermediate language by mapping patterns of sensor signals to basic actions like *sit*, *walk*, *talk* etc. which can be seen as triggers and preconditions for specific tasks in the models. Basic actions are supplemented with attributes like timestamp, position information, confidence, and others.

Essentially, the elaboration of task models in the context of Smart Environments comprises:

1. the identification of entities for which tasks have to be modeled
2. the elaboration of these models from the top element (the abstract main goal) down to the tree's leaves describing all single actions necessary for each sub goal (refinements/alternatives)
3. describing the interdependencies between these entities (see [15])

These steps entail a lot of work. For the first step, a thorough understanding of the domain and the users is essential as the defined roles should reflect the users' perspective of the modeled processes. However, the set of roles cannot be determined automatically. As far as the identification of devices involved is concerned, we can derive them from observations of user performances (scenarios) conducted in the Smart Environment. For the second step, domain knowledge and user knowledge are also required. Nevertheless, with increasing level of detail and decreasing level of abstraction (which is the most laborious part of the model as the task tree progressively broadens downwards) some useful time and labour saving assistance can be given to partially automate the construction of the models.

In Section 3, the above-mentioned support type is described in more detail.

2 Related Work

Several approaches adress 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 [3]. Some of the numerous applications beyond the "basic" mining of processes are the identification of bottle-necks, verification of business rules, and creation of social network graphs. Import from a number of different sources is possible and many plugins 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.

An example of a system generating task models is *ActionStream* [10] that records user activities for a long period 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 attempt to establish a means of communication between different skilled individuals that is as decisive as easy-to-understand.

Some further references that are very much related to specific parts of our approach will be discussed together with the following presentation of our ideas.

3 Design and Extraction

After the definition of roles and devices (introduced as step 1 in the introduction) the elaboration of task models for all identified entities is our next step. The procedure of constructing role-based task models as proposed in this paper consists of two complementary parts: Taking the role's main goal as starting point, the modeler refines the top goal into sub goals, declares alternatives, temporal dependencies and so on as far as it is known and reasonable from the user's point of view. Part two makes use of recorded scenarios as additional input and synthesizes the task tree following a bottom-up technique, thus completing the manually designed parts. Both parts are highlighted below.

3.1 Top-down Construction of Models

The development process for Smart Environments as proposed here has to start with a careful requirement analysis resulting in, among other things, a set of use cases each comprising a set of roles and devices involved as well as an informal description of what the use case is about and how the roles and devices contribute to the overall objective.

For each role within a use case, the main purpose of a user fulfilling that role has to be identified and is modeled as the root node of the task model. That node is the most abstract description of why a user assumes a certain role for a specific time. Based on the results of the requirements analysis this main task is further decomposed into sub-tasks (i.e. nodes representing tasks with a lower degree of abstraction and vital to achieve the goal related to the parent task) or choice tasks (i.e. the goal related to the parent node can be achieved in more than one way). Between tasks at the same level of abstraction temporal operators, as far as known, can be used to specify the order of task execution. This procedure is continued until all identified activities and their discovered relations are formalized. What emerges from this procedure is a hierarchical model of what constitutes the respective role from the perspective of the user in domain specific terms. It can be supported by patterns that reflect already analysed and specified behavior and characteristics of roles, devices or spaces. Such a pattern-based approach is discussed in [16]. In [17] it is presented how this approach can be used to develop supportive user interfaces for Smart Environments.

However, patterns are available only if the domain is already known and analysed by experts. Knowledge of the domain is a precondition for the top-down approach. Sometimes such kind of knowledge does not exist. In this case the opposite buttom-up approach is worth to be studied. Based on observed scenarios hierarchical models can be constructed. Also combining both approaches seems to be promising. In that way some knowledge of the domain can be combined with information of observed scenarios.

This idea will be explained in more detail in the rest of this paper.

3.2 Bottom-up Synthesis of Models

For the bottom-up synthesis of task models previously recorded traces are essential. These sequences list all events emitted by sensors and devices (e.g. projectors, lamps) during the scenario executions. Thus, the user activities as detected by the system as well as the user interaction with the devices can be analyzed. By applying appropriately configured and trained complex pattern detection algorithms on the sensor traces the event stream is filtered and transformed into new events (in the following referred to as "basic actions") that reach a semantic level suitable for further processing. This transformation itself is a challenging task and requires considerable investment of technical knowledge, experience and work. It has to be set up for each environment separately and has to be extended with the addition of new sensor types, considering the possibilities of new forms of sensor fusion.

Detecting Basic Actions from Sensor Stream. Figure 1 illustrates the detection of complex events and the generation of a basic action stream.

Given a potentially unlimited stream of sensor data from different sources as input, a stream-to-stream transformation is achieved by concurrently running queries in order to find occurences of predefined event patterns. Reducing the

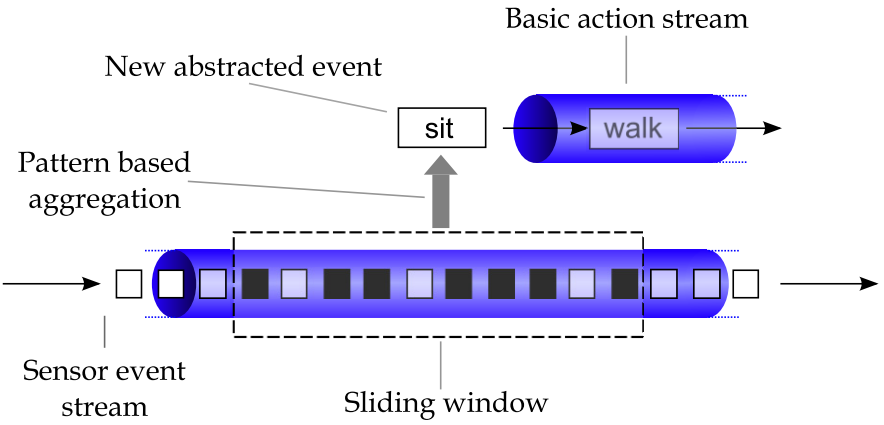

Fig. 1. Complex event detection from sensor stream

scope of queries via *sliding windows* helps restricting resource usage. Krämer et al. [9] suggest a stream processing infrastructure based on an operator and a physical algebra that carries over a wide range of theoretical aspects of stream processing (e.g. windowed aggregation, adaptive resource mangement, and query optimization) into a consistent implementation concept. Furthermore, Hoßbach et al. describe a Java middleware for event processing (JEPC: Java Event Processing Connectivity) in [8].

In the following, we assume that such a transformation with sufficient performance (real time is required for event detection during usage) is available and continue with the basic action stream as the starting point.

Linking Basic Actions to Task Models. After transforming the sensor event stream into a stream of semantically meaningful basic actions, the bottom-up task model construction comprises the following three steps:

1. Pattern recognition
2. Hierarchy creation
3. Identification of temporal relations

Pattern recognition. – Frequently occurring action sequences in the recorded scenarios might indicate a correlation between these actions that can be interpreted as a more abstract task in some cases. Using algorithms like the *Apriori algorithm* [1], *PrefixSpan* [12], and *BIDE+* [14] a set of such sequences can be detected, providing the modeler with groups of potentially combinable actions. Because of the fact that these groups are of algorithmic origin and initially nameless, the reviewing of these groups comprises the check of their reasonableness as well as the specification of a proper label. Once a suggested group is confirmed and labeled, its name is treated as the name of a new task that references the actions of that group as child nodes. Our implementation of the pattern recognition

provides the selection of one of the implemented algorithms and the definition of the usual parameters for pattern recognition like "Min support", "Max support", and minima and maxima for the number of actions in a sequence and the time interval a sequence has to fit in. In our prototype, all detected sequences are shown in tabs ordered by sequence length while an input field per sequence is offered to confirm a group by entering a name for it.

Hierarchy creation. – Combining the model fragments created so far with the basic action tasks from the scenario traces (protocols) is the main goal of this step. Taking the identified main goal as the root node, the tasks are successively decomposed into less abstract tasks in a top-down technique, while the recorded action traces are arranged in a bottom-up manner, initially quite loosely coupled to the upper part of the task tree. This step combines the analysis results with the output of the synthesis. To achieve this, we use a modified version of the *LearnModel* algorithm [6] which originally exploits objects links as an indication for probably connected activities. Since we cannot assume the presence of object-related information we decided not to integrate the propagator concept from *LearnModel*. The second modification concerns the set of temporal operators: As the models in our approach should be usable with other tools, our notation and operators have been adapted to the operators of CTT.

During the creation of a hierarchy the *Choice* relation has a special role to play as it crucially influences the emerging structure.

Among the approved action groups detected by the pattern recognition there may be more than one group assigned to a task (called *group set*). In each group set the minimal group (i.e. the group with the lowest number of sub tasks) is determined. The smaller number of basic actions is used as the secondary criterion, in case of equality one group is randomly selected.

Each occurrence of the minimal group within the action sequences of other groups is replaced by this group's assigned task. Then, the group is removed from the group set. This process is repeated until either only one group is left or no more substitutions are possible.

Tasks that were used as replacements for action sequences in other groups as well as the remaining action sequences are considered as alternative realizations of the common super-task.

Finally, the *Align* and *InduceOrdering* algorithms are applied. In short, *Align* generates "recipes" for every group of sequences assigned to the same non-basic task while *InduceOrdering* adjusts the ordering of the sequences. This procedure is repeatedly executed until all protocol actions are assigned directly or indirectly (one or more nodes between the nodes were created during the pattern recognition) to a task node modeled in the top-down modeling process described in section 3.1. Note that only those actions occurring in any of the trace protocols selected for this step are considered.

Identification of temporal relations. – Each task and action can be flagged as optional. Within the establishing of the temporal relations the default value ("mandatory") is considered to be correct until a conflict with the analyzed action

traces occurs. Initially, all child nodes of a given node are considered to be in an *enabling* relation with respect to their first occurrence. This assumption may be proved to be incorrect:

- Two actions or tasks a_1 and a_2 with a_1 being assigned to a_i and a_2 to a_j are found to occur in the order a_2, ..., a_1. Now, an *order independent* relation between a_i and a_j is suggested because a_1 and a_2 were previously detected in a reversed order and an *enabling* relation to be revoked has been suggested.
- Iterations are suggested for each task or sequence that occurs more than once. Two parameters can be set to control the detection: The minimum number of occurrences of a sequence and the maximum length of sequences that are examined during this step. Thus, the modeler can reduce the computational effort of analyzing large protocols. The number of repetitions is attached as annotation to the iteration.
- End timestamps are synthesized for tasks from the timestamps of the connected basic actions. Based on that, the concurrency of tasks is detected and denoted in the model.

During the derivation of temporal operators each time an already found operator is modified, all operators up to the root are checked and modified, if violated. The process of temporal operator derivation can be repeated any number of times, selecting different sets of task traces recorded before and with varying settings for operator detection. Thus, different rival model versions can be created, discussed and evaluated.

3.3 Example

This section will exemplify the task model synthesis. Space restriction does not enable the example to cover all aspects like pattern recognition but allows us to demonstrate the principle of the proposed approach. Figure 2 depicts the analysis model of a *lecture event* use case: One model (a) describes the whole process and another model (b) defines the tasks of a role *presenter*.

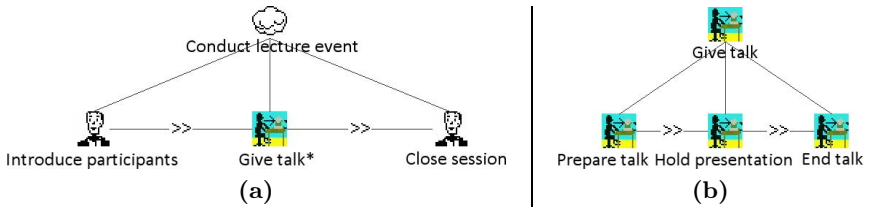

Fig. 2. Analysis models: **(a)** whole process, **(b)** model for role *presenter*

In Figure 3, some excerpts from an example scenario trace (basic action stream) are shown, illustrating how users behave to accomplish the task *Conduct lecture event*.

Timestamp	PID	Basic action	Attr1	Attr2
00:00:24	1	sit	seat3	
00:00:26	2	walk		
00:00:32	2	sit	seat1	
00:00:33	2	talk		
00:01:05	1	walk	pres_zone1	
00:01:17	1	operate	beam4	sw_on
00:01:28	1	operate	beam4	connect
00:01:28	1	operate	lap1	connect
00:01:43	1	operate	screen3	lower
00:01:49	1	walk		
00:01:54	1	talk		
00:02:40	1	talk		
00:03:12	1	operate	lap1	nxt_slide

Timestamp	PID	Basic action	Attr1	Attr2
00:09:02	1	operate	lap1	disconnect
00:09:02	1	operate	beam4	disconnect
00:09:40	1	walk		
00:09:42	2	walk		
00:09:59	1	sit	seat3	
00:10:01	2	walk	pres_zone2	
00:10:36	2	operate	beam2	sw_on
00:10:50	2	operate	lap3	connect
00:10:50	2	operate	beam2	connect

Timestamp	PID	Basic action	Attr1	Attr2
00:23:44	2	operate	lap3	disconnect
00:24:19	2	operate	beam2	disconnect
00:24:30	2	operate	screen2	raise

Fig. 3. Snippets from example scenario trace

Now the modeler has to mark sequences of actions as belonging to tasks or to task trees, be they leaf nodes or inner nodes. Depending on the algorithm selected, a synthesized structure with temporal relations and modifications of existing relations is presented to the modeler (Figure 4).

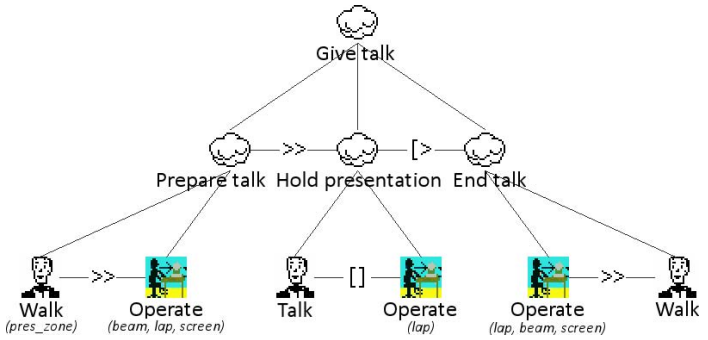

Fig. 4. Combined task model for role *presenter*

Here, the design model has been extended with the basic actions from the scenario trace. Multiple *Operate* actions are depicted in an aggregated way to save space; temporal relations for those tasks have also been generated. The modeler now decides, if and to which extent the suggestions are appropriate, and finishes the model creation. Similarly, the model for the whole process is enriched with generated model fragments.

This partly automated elaboration process results in models bridging the gap between basic action streams as detected from sensor data and the more abstract analysis task models.

3.4 Discussion

The combination of manual creation and semi-automatic generation of models can substantially contribute to a user-centered design of assistance in smart environments. Nonetheless, some effort is needed to prepare and control the generation. The main purpose of this effort is to provide reasonable assurance that the resulting models do not only describe the scenarios in a formally correct way (which could be achieved by fully automated generation) but also in a way that properly reflects the use case from the users' domain specific perspective. The usage of such models is not limited to the derivation of proactive assistance, but also includes the interaction between users and the environment in specific situations: Models can be used as understandable visualizations of the system's state ("*Why* has something just happened?", "What will happen *next*?") as well as for giving specific instructions ("The talk is not over yet.") that may include (or prevent) a number of device actions.

The approach presented here enables modelers to create task models in a more effective and proportionate manner.

4 Summary and Outlook

Using models as primary artifacts in software development to specify user behavior and user tasks is valuable, but requires a lot of work. This is escpecially true for Smart Environments, since the establishing of a link between the rather abstract models and the sensor data as the main input of the system is a very time-consuming task. The combination of manual top-down creation and semi-automatic bottom-up generation of task models based on scenario traces can lead to a considerable reduction in cost and effort. Particular priority is attached to the retaining of the models' character as an easily understandable formalization of tasks and activities from the users' point of view.

Further investigation of possibilities for improving the support of model creation for Smart Environments may include the more effective gathering of scenario data: Thus far, traces are recorded from real users in the environment. It may be useful to allow the recording of traces in a simulated environment. Such an approach would benefit from and rely on the introduction of an intermediate language as outlined in this paper (basic action stream).

Acknowledgements. The work of the first author is supported by DFG graduate school 1424 (MuSAMA) at the University of Rostock, Germany.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. VLDB 1994, pp. 487–499. Morgan Kaufmann (1994)
2. Chikhaoui, B., Wang, S., Pigot, H.: A Frequent Pattern Mining Approach for ADLs Recognition in Smart Environments. In: IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 248–255 (2011)

3. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
4. El-Ramly, M., Stroulia, E., Sorenson, P.: Recovering Software Requirements from System-user Interaction Traces. In: Proc. SEKE 2002, pp. 447–454. ACM Press (2002)
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. Garland, A., Lesh, N.: Learning Hierarchical Task Models By Demonstration. Technical Report, Mitsubishi Electric Research Laboratories (2003)
7. Hamou-Lhadj, A., Braun, E., Amyot, D., Lethbridge, T.: Recovering Behavioral Design Models from Execution Traces. In: 9th European Conference on Software Maintenance and Reengineering, pp. 112–123. IEEE Computer Society (2005)
8. Hoßbach, B., Glombiewski, N., Morgen, A., Ritter, F., Seeger, B.: JEPC: The Java Event Processing Connectivity. Datenbank-Spektrum 13(3), 167–178 (2013)
9. Krämer, J., Seeger, B.: Semantics and Implementation of Continuous Sliding Window Queries over Data Streams. ACM Trans. Database Syst., 4:1–4:49 (2009)
10. Maulsby, D.: Inductive Task Modeling for User Interface Customization. In: Proc. IUI 1997, pp. 233–236. ACM (1997)
11. Paris, C., Lu, S., Linden, K.V.: Environments for the Construction and Use of Task Models. In: The Handbook of Task Analysis for Human-Computer Interaction, pp. 467–482. Lawrence Erlbaum Associates (2004)
12. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.-C.: Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. IEEE Transactions on Knowledge and Data Engineering 16(11), 1424–1440 (2004)
13. Seyff, N.: Exploring how to use scenarios to discover requirements. Requirements Engineering, 91–111 (2009)
14. Wang, J., Han, J.: BIDE: Efficient Mining of Frequent Closed Sequences. In: Proc. ICDE 2007, pp. 79–90. IEEE Computer Society (2007)
15. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and Task Modeling for Collaborative Environments. Journal of Universal Computer Science 14(19), 3188–3201 (2008) (Special Issue on Human-Computer Interaction)
16. Zaki, M., Wurdel, M., Forbrig, P.: Pattern Driven Task Model Refinement. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J.F. (eds.) International Symposium on DCAI. AISC, vol. 91, pp. 249–256. Springer, Heidelberg (2011)
17. Zaki, M., Forbrig, P.: A methodology for generating an assistive system for smart environments based on contextual activity patterns. In: EICS 2013, London, pp. 75–80 (2013)