SCENARIO RECOGNITION IN MODERN BUILDING AUTOMATION

R. Lang, D. Bruckner, G. Pratl, R. Velik, T. Deutsch

Institute of Computer Technology,
Vienna University of Technology
{langr, bruckner, pratl, velik, deutsch}@ict.tuwien.ac.at

Abstract: Modern building automation has to deal with very different types of demands, depending on the use of the building and therefore the persons acting within this building. To meet the demands of situation awareness in modern building automation, scenario recognition becomes more and more important to detect such demands and react to them. Two concepts of scenario recognition and their implementation will be introduced, one based on predefined templates and the other using an unsupervised learning algorithm using statistical methods. Implemented applications will be described and their advantages and disadvantages outlined.

Keywords: Building automation, Ubiquitous computing, Scenario recognition, Surveillance system

1. INTRODUCTION

Modern building automation has to deal with very different types of demands, depending on the use of the building (hospital, airport, soccer stadium, office building, etc.) and therefore the persons acting within this building as described in (Dietrich *et al.*, 2001). Ubiquitous computers become more and more a topic in building automation, supporting the persons in their actions and with relevant information needed as mentioned in (Tapia *et al.*, 2004).

But without knowing anything about the situation the persons are involved in, the demands that can be satisfied by an implemented system are very limited. Even a simple control variable like the temperature of a room can depend on various other values than only e.g. the time or date. But the essential values, a building automation system has to be aware of, like safety, security, convenience, etc. can very often depend on a huge variety of partly redundant sensor information.

To meet the demands of situation awareness in modern building automation, scenario recognition becomes more and more important to detect such demands and react to them as shown in (Pratl *et al.*, 2005b). The following two chapters will introduce two very different approaches of scenario recognition.

The first scenario recognition model is based on predefined perception patterns, called image templates, that combine different sensor outputs and gives them a semantic meaning. Recognized image templates are then used as transition conditions between the states of a scenario recognition process based on predefined pattern of possible scenarios. It will be shown how the scenario recognition has been designed, tested and implemented.

The second model of scenario recognition follows an approach based on unsupervised learning of behaviour pattern. During a learning phase, the system detects and learns all new scenarios that are taking place and recognizes them or remarks exceptional scenarios during the operational phase. Finally the concepts will be compared, advantages and disadvantages shown.

2. STRUCTURED SCENARIO RECOGNITION SYSTEM

The structured scenario recognition system has been designed to detect scenarios that can be predefined before the operational phase is started. As shown in Figure 1, there are three phases designated.

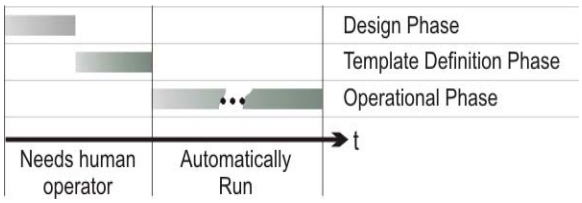

Fig. 1: Three phases of appliance

A *design phase* is used to predefine the different entities of sensors that are used in the application. During the *template definition phase*, the concept of condensing linked sensor data to symbols and scenario templates using these symbols has to be predefined. This architecture is sensor independent, which means that by following the concept of symbolization – defined in (Pratl, 2006) – as a standardized interface between sensor data and the structured scenario recognition system, any type of sensor entity can be used.
Based on the level of symbolization there are two types of templates necessary to guarantee scenario recognition. First, the *image template*, representing a typical set of perceived data within a single moment and second the *scenario template*, representing a perceived sequence in time. Finally, the *operational phase* can start, where the systems output are one or more recognized scenarios. The following chapters will describe why image and scenario templates are necessary and how they are implemented and defined. The concept of symbolization used is defined in (Pratl, 2006) and will not be described here.

2.1. The difference between a play card and a royal flush

During the research for a new scenario recognition model, the authors of this article investigated several models of how the human perceptive system handles data, symbolizes them to a higher semantic meaning and stores recognized scenarios. We came up with a concept using standard poker playing cards as a metaphor for what we had defined as an image template. Like the ace of hearts contains different kinds of information (e.g. colour, type, rank, value) we made several templates of a perceived image which can be compared to the currently ongoing situation. When a specific image template matches the current situation, we changed the state of the system that observed a room in their department and took several actions.
Using the picture of play cards of a standard poker deck, we presented this technical concept to a neuro-psychoanalytical advisor – apparently a passionate poker player. Following the explanations of the concept, the advisor summed up: “As a poker player, the fact of looking at an ace of hearts does not mean anything to me. Only if I am in possession of the whole set of cards that complete a royal flush, the ace of hearts becomes meaningful to me”.
In the same way, a playing card alone does not have any relevance to a poker game, sensory values are not sufficient to realize modern scenario recognition that needs a broad overview of information.

2.2. Image Definition

An *image template* consists of a set of rules, defining the perception of specific types of symbols. These symbols can be very simple like the number of persons in a room or the temperature or humidity in a room or they can contain complex information like ‘a meeting is taking place in the conference room’. Because of the generic way of the concept for defining image templates, the name and meaning of a symbol shall be abstracted in the further text as symbols S_1 to S_n.
As the perception module of such a system produces a constant stream of symbols, every calculation step contains a subset (P) of the set of all symbols (S) including the elements S_1 to S_n.

$S = \{S_1,...,S_n\}$ possible perceived symbols
$P = \{S_3, S_9, S_{17},...,S_m\} \subseteq S$ perceived symbols

$IT_1 = \{w_3*S_3 \text{ AND } w_{17}*S_{17}\} \subseteq P$ Image template

Since all the definitions of image templates (IT) have to be compared with the currently perceived and weighted (w_3 and w_{17} are the weight factors) set of symbols (P), a tree structure was created to define the content of one image template. Figure 2 shows the three basic elements of this tree: *image element* (iE), *image node* (iN) and *image leaf* (iL).

Name	Symbol	Base
iE		-
iN		iE
iL		iE

Fig. 2: Basic elements in image template definition

As a base class of all elements within the tree, the *image element* holds the following data. For debugging and visualization, any element contains a name and optional a description of the specific use. Further, any element – no matter if node or leaf – stores the information if this element is optional or mandatory.
Additionally to this information, the *image node* is equipped with the information about its child nodes. These can be further sub nodes as well as image leaves. It is also equipped with the Boolean composition (AND or OR) between these sub elements, a negation flag that represents a Boolean negation of all sub elements. The image leaf on the other hand defines the type of symbol that shall be part of the image template detection and a logical operator (==, !=, <, <=, >, >=) that compares the value of a symbol to the value defined in this image

leaf. This value depends on the kind of symbol generated from the corresponding sensors, which can be e.g. an integer value (counter, temperature ...) or a

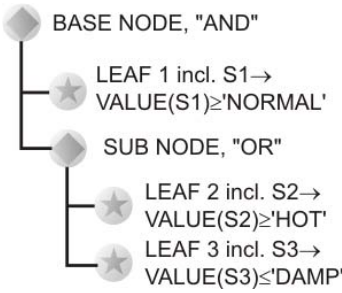

Fig. 3: Example for a generic image template definition accomplished in a tree structure

fuzzified value (freezing, cold, warm, hot, boiling in case of temperature) which is mapped to an enumeration. In Figure 3 an example is given for a generic image template definition accomplished in a tree structure. Described from bottom to the top, in this image template, either the value of symbol S_2 has to be equal or higher than 'HOT' (as a symbolic output for e.g. a temperature sensor – for what ever that means semantically) OR the value of symbol S_3 has to be equal or lower than 'DAMP' (as a symbolic output for e.g. a humidity sensor).

With this lower part of the illustrated tree, the first half of the image template is defined and matches, if the perceived data is within the defined range of values. The second half of the image template is matching, if the value of symbol S_1 equals or is greater than 'NORMAL'. The image template only matches fully, when both conditions – leaf 1 as well as the sub-node – are matching. This is defined by the logical operator AND in the base node.

2.3 The result of the matching algorithm

After comparison of the predefined image template with the actual perceived symbols, a value from zero to one must be generated that describes the quality of the match. To calculate this quality, a simple algorithm has been implemented. First, the number of elements within a node, containing the operator AND are counted and summed up with the nodes containing an OR-operator. Applying this algorithm recursively through all branches of the tree, the total weight of the tree is calculated, no matter if there is any match or not.

In the example tree of Figure 3, the total weight is two – one for leaf 1 and one for leaf 2 or 3 or for both of them. In the next step, the same algorithm counts only leaves which have a valid condition match. The number of matching conditions divided by the total weight of the tree gives the quality of the match. Referring again to the example depicted in Figure 3, the match would be 1.0 if leaf 1 and one or both of leaf 2 and 3 matches, 0.5 if leaf 1 but neither leaf 2 nor 3 are matching or only leaf 2 or 3 are matching. The match would be 0 if none of the leaf rules meet the conditions. Figure 4 shows these possibilities without the full and the zero matches.

MATCH	L1	L2	L3
1.0	X	X	-
1.0	X	-	X
0.5	X	-	-
0.5	-	X	X
0.5	-	-	X

Fig 4: Possibilities of matches in the example image template

The problem with this algorithm in common is the fact, that every element has the same weight. This could be useful for several applications, but driven by a rising demand in different applications, the image element was extended by a weight that can be additionally defined. The algorithm was slightly adapted to the given weight in each node.

As described in (Dornes, 2001) the concept of comparing the current perception to a set of templates that have been previously learned or in this case predefined, is following a bionic approach. By retracting the layer of image template recognition, a new and higher semantic level of symbolization has been reached. Based on this level a scenario recognition has been implemented that will be described in the following chapter.

2.4. Scenario Definition

Until now, the described concept only deals with a single moment in time. Every image template and the resulting match do not contain the variable of time. By adding the value of time to the concept, a new step in the hierarchy is made, containing the perceived data in the past.

A scenario is defined as a sequence in time of several recognized images that are perceived. Corresponding to chapter 2.1 it is the royal flush in a building automation systems perception unit. By looking at common sequence diagrams, it has been decided to use the concept of state charts to represent such scenarios. From the beginning to the end of a scenario recognition process, several circumstances may occur. For instance, there may be *more than one possibility* of events that have to be perceived by the image template recognition. *Different paths* have to be covered within the definition of a scenario process. It is necessary to make *global abort conditions* possible, either caused by a timeout or another event that triggers the abortion of the scenario recognition and resets it. Figure 5 shows the four basic elements of a scenario definition and their purpose: starting scenario state (SCB), ending scenario state (SCE), scenario state (SC) and scenario transition (ST).

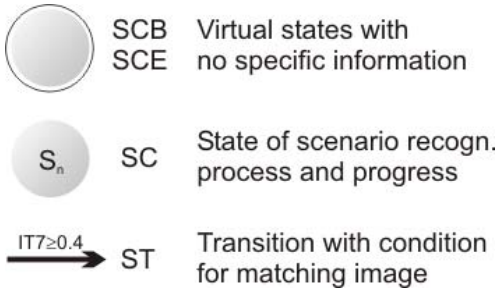

Fig. 5: Basic elements of scenario definition

The start and end states only have a virtual meaning in the system. All scenario recognition processes are initially set to the begin state, waiting for the first transition condition. As the end state is reached, the scenario recognition process reaches the end of its lifetime and does not include any further information. The scenario states between the start and end state are indicating the process of the scenario recognition defined by there position. Each scenario state (except of the end state) holds a list of transitions. These transitions specify the condition to switch to the next state. The match of a selected image template must meet the specified condition. In figure 5 the image template number 7 must have a perceived match higher than 40 percent.

Figure 6 shows a generic scenario template definition. In this case, the scenario recognition process will be triggered when the image template (IT) number 1 is perceived with a match of at least 60 percent. The current state is set to the state S1. This state contains two different transitions. The first transition leads to state S2 as soon as IT 2 is perceived with a match of hundred percent. The second transition would lead to the state S3 when IT 3 is perceived with at least 80 percent. With a match of at least 70 percent, IT 4 closes the path and the recognition process gets into the state S2. The final condition for a complete scenario recognition is fulfilled when IT 5 is perceived with a match of at least 50 percent.

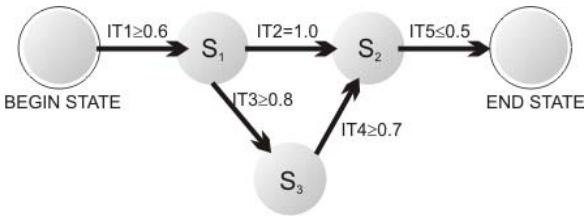

Fig. 6: Generic scenario template definition

The definition of both, the image templates and the scenario templates, have to be defined in an XML structured file, representing the knowledge of the system. Images or scenarios that are not defined in this database will not be recognized.

2.5. Test of Implementation

To test the implementation and verify the functional correctness of the image and scenario recognition, a simulation tool was created. This tool offers the possibility to generate symbols and build a stream of perception as described in chapter 3.2. without the necessity of real sensor data. The generated perception was used as an input for the image template matching algorithm. To test the scenario recognition unit, recognized image templates and their match had to be generated. It was shown, that scenarios could be recognized, an abort transition successfully aborted the recognition process and timeouts also lead to an abortion of the process.

During the test phase, the scenario recognition unit was migrated to a spin off part of the research project Artificial Recognition System (ARS) further explained in (Pratl *et al.*, 2005a). In this project, simulated, embodied, autonomous agents and their perception and decision unit are implemented as described in (Deutsch *et al.*, 2006). It has been shown, that the described scenario recognition model can also cope with these demands, completely different from modern building automation. Figure 7 shows the visualization of scenario recognition and the internal values of the embedded autonomous agents.

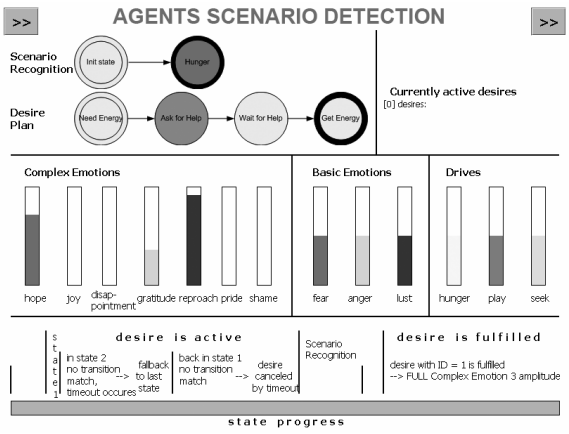

Fig. 7: Visualization of scenario recognition within embodied autonomous agents.

During the test it has been shown that a recognition boost of image templates that are expected by already started recognition processes leads to a better performance in scenario recognition. To boost a recognized image template means that the calculated match will be increased by a predefined value which increases the probability that a scenario transition condition will be met.

2.6. Application

A scenario recognition model was implemented in the kitchen of the institute. Several cheap sensors of different types (tactile sensor, light barriers, motion detector, etc.) were used to show the advantage of redundant sensor arrangement.

Based on the output of the project SmaKi (the smart kitchen at the institute of technology in Vienna) and the implemented symbol factory of the project ARS (Pratl and Palensky, 2005a) the scenario recognition module was embedded into the system. Recognized scenarios were visualized on a screen. As a standard scenario that has to be detected, the child in danger scenario was created as a test bench. For this scenario, the system must be aware of the situation, where a child enters the kitchen, comes close to the stove and the heater plate is still hot. If this scenario is recognized, an alert is produced indicating a safety critical situation.

3. SCENARIO RECOGNITION WITH STATISTICAL METHODS

A system which is intended to be aware of the context of persons or systems needs the ability of adapting itself to changing conditions that maybe haven't even been foreseen at design time.
Therefore, as a supplementary approach to scenario recognition algorithms that rely on pre-defined pattern, an approach based on unsupervised learning of behaviour pattern is presented in this section. Several ways have to be distinguished when considering introducing learning methods into scenario recognition: They range from making the definitions of pre-defined scenario building blocks fuzzy in terms of sensor values or allow various sequences of pre-defined events in pre-defined sequence templates to happen, and end up with completely unsupervised learning of building blocks, sequences, and their interconnections.

In the following sections the principles of unsupervised learning of behaviour are presented and an example is discussed. The ultimate goal of all the algorithms is to create a model of behaviour that can be easily interpreted by humans.

3.1. Scenario Learning Principles

Finding recurring behavioural pattern in sensor data that can be used to model scenarios in a way that humans can interpret heavily depends on the type of sensory data and the characteristics of the data's generation.

Each sensor therefore needs a pre-processing to output only "useful" data. E. g. *keep alive* messages with identical values in wireless networks or recurring bursts from motion detectors or other presence detection sensors have to be omitted in later learning procedures. Once the pre-processing is decided for all involved sensor types, no more human operation is necessary in order to let the system learn and later recognize behaviour pattern (Fig. 8).

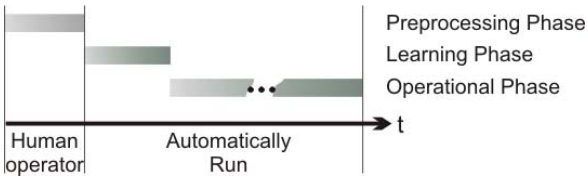

Fig 8: Three phases of implementation

The modelling process results in the creation of states. States are the fundamental building blocks of this kind of scenario learning algorithms. Each state comprises an emission distribution (describing its possible sensor values), a transition distribution (describing the connections between states in the model), and – for learning and merging purposes – a weight. The goal is that states represent the behaviour of the observed system or person.

Having processed the raw sensor data to represent some meaningful status or event, a data base with several chains of sensor values forms the base for scenario learning. Ideally, those sensor value chains have their starting point with the start of the desired scenario and their end should coincide with the end of the scenario. Of course, this prerequisite violates the intended unsupervised fashion of learning, but in several cases – e.g. the learning of daily routines, or the learning of behaviour in a room delimited with opening of the door – the time frame of the scenario(s) is known and can be used as prior knowledge.

After having obtained an initial data set, the value chains are compared. The idea is that the same scenario is expected to generate roughly the same sensor values in each occurrence and each value chain represents one particular form of the desired scenario. The comparison is undertaken in several steps with the goal to merge the values into states and to reduce the number of states until they reach a degree of expressiveness to be interpreted by a human.

One of the comparison steps looks for equal sensor values at equal times with equal delay times to another value; another step looks for consecutive recurring pattern to merge. For a detailed description of the algorithms see (Bruckner, 2007).

The underlying mathematical structure of the model is the Hidden Markov Model (HMM) (Rabiner *et al.*, 1986). It consists of a transition probability matrix, an emission probability matrix and the initial state distribution vector. The HMM provides three well described algorithms for the operator which makes it ideal for the purpose of representing scenarios:

The forward algorithm is used to infer the probability of an observed sequence of values to be generated by the model (evaluation).

The Viterbi algorithm is used to find the path in the model that most probably generated the observed sequence (decoding).

The forward-backward and Baum-Welsh algorithm is used to adjust the parameters of the model (not the structure!) in order to adapt to newly seen observation sequences (parameter learning).

3.2. Model Interpretation

For explaining the power of this approach an example located in an office environment is illustrated. The input sensor data comes from a motion detector that generates a "1" when it detects motion (at a maximum rate of every five seconds) and a "0" after one minute of no detected motion. For pre-processing the motion detector values are averaged over 30 min periods in order to represent "more" or "less" activity within this time frames. The model was learned with 15 of those 48 values long sensor value chains. Figure 9 shows the result. The model consists of 14 states (plus initial and final state) as a result of the merging of more than 15.000 sensor values emitted by the motion detector during the observation period of 15 days.

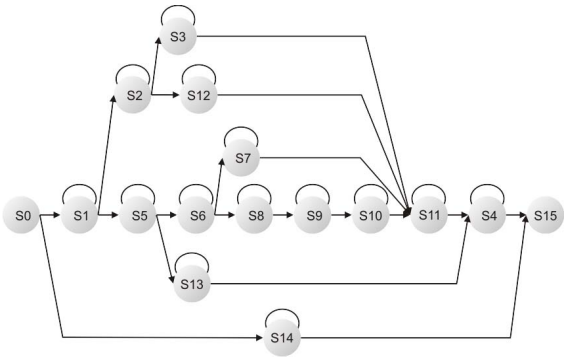

Fig. 9. The model. States are depicted with labelled circles, possible transitions with arrows. The initial and final states (0 and 15) appear at the start and end of every sensor value chain. The ellipses above the states show possible self-transitions.

The model identified 6 different daily routines represented by the various paths from the left to the right. One of these paths (0, 1, 5, 13, 4, 15) is discussed in the following paragraph:

Figure 10 shows a path in the graph of the model while figure 11 shows the sensor value chain which created the states and transitions of this path – its Viterbi path. It is important to mention that the graphical representation of the model contains only sparse information about time insofar as the transitions show the possible sequences of states; and states to the right cannot be visited timely before states to the left.

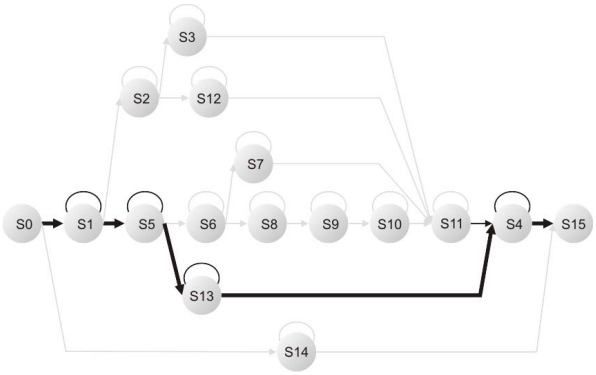

Fig. 10. A path in the learned model. The path (0, 1, 5, 13, 4, 15) represents a particular daily routine from 0:00 to 24:00 o'clock in the observed office.

But there is no information – and cannot be – when during observation the current state of the model changes from 13 to 4 for example. Therefore the path of states has to be viewed together with the sensor values which created it. The Viterbi algorithm then

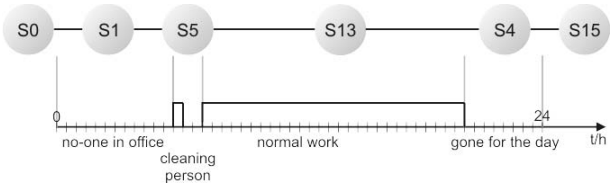

Fig. 11. Sensor values that created the path (0, 1, 5, 13, 4, 15). In the vertical middle of the figure the pre-processed 48 "sensor values" are drawn. On top the sequence of states that matches best the observed sensor values are shown and, finally, on bottom the interpretation thereof is given.

outputs the sequence of states as can be viewed on top of Fig. 11. In this illustration it is possible to interpret the – fully unsupervised learned – states of the model: State 1 represents the morning time where nobody is in the office and state 4 the evenings. As can be seen in the model graph, all daily routines, except the weekends represented by state 14, share those morning and evening states.

State 5 was a surprise, it represents the cleaning personal! They entered the room nearly every day at about 6:00 in the morning and emptied baskets, etc. Finally, state 13 represents the normal activity in the office which lasts from morning until evening.

This little example shows the power of the unsupervised scenario learning approach. It allows learning a model only from sensor data without human supervision in a way that a human operator can interpret the meaning of the model's elements.

4. MODEL COMPARISON

The two models that are described in the two sections above use different approaches for achieving the same goal. Perception of everyday activities and crude "understanding" of what is happening is vital for the future of automation systems. Both approaches have their advantages and a comparison has to consider the different underlying methods as well as the different applications for the models.

The Structured Scenario Recognition System uses predefined scenarios and can thus provide additional semantic information about the perceived scenarios. This is vital, if a human operator needs to evaluate the information that the system has perceived. In cases where privacy is most important, this additional information, which is entered into the system before the operational phase, greatly assists the operator. Privacy is, for example, one of the main concerns, when persons with special needs are observed: in retirement homes or in hospitals such a system can be well integrated, since it can provide constant 24-hour surveillance and only alarm a human user, if it has detected a predefined scenario that requires assistance. The disadvantage of the model is the initialization phase, which delays commissioning of the system. Although many scenarios may be predefined and not change from one installation to another, it will still be necessary to do adaptations depending on the layout and situation at a new venue. Such a commission may increase the costs of the whole system.

The Scenario Recognition with Statistical Methods does not suffer from a long and costly initialization phase, since it automatically adapts to the incoming sensor information and learns to tell apart usual from unusual situations. It is in principle possible to assign semantic information to a model that the system has learned; however, this may not always be the case or may simply be impossible. Since the associations between sensor values are learned completely autonomously, the system has as such no means to provide information about the scenarios it has detected: it can only provide information, whether a scenario is unusual. An experienced operator can use this information and attempt to assign "meaning" to the scenarios, but this is subject to personal interpretations.

It appears that the best solution is a combination of both systems – a predefined recognition system that is also able to tell apart common and uncommon situations. This way the system is equipped with basic knowledge of its whereabouts, but is still flexible enough to adapt to specific situations.

5. CONCLUSION AND OUTLOOK

The article introduces two developed models for scenario recognition – one model based on predefined scenario patterns and one based on unsupervised learned patterns. The work bases on the most recent research results of (Pratl *et al.*, 2007).
The model of the structured scenario recognition system shows an approach for detecting predefined scenarios within sensor equipped buildings. By predefining templates of perceived data and expected scenarios within an XML knowledge base, the operational phase of scenario recognition can be started. The output of such a system is recognized scenarios. On this basis further processing e.g. taking according actions can be done. The described system is not able to learn new scenarios during operational phase except the knowledge base is extended by a human operator.

The model of Scenario Recognition with Statistical Methods has been introduced which is able to learn scenarios by its own during a learning phase and therefore minimizes the effort of adapting the system to new surroundings. However, the semantic meaning of automatically learned scenarios will not emerge during learning or operational phase, except an operator filters redundant or meaningless scenarios and adds a semantic meaning to the detected scenarios.

Currently, both models of scenario recognition are implemented and in a test run in a modern kitchen and in the living rooms of an elderly home. Further applications in the field of safety critical surveillance systems (e.g. airport, modern soccer stadium) are planned.

REFERENCES

Bruckner D. (2007). Probabilistic models in building automation – recognizing scnearios with statistical methods. *Ph.D. theses, Univ. of Technology Vienna*

Deutsch T., Lang R., Pratl G., Brainin E., Teicher S. (2006). Applying psychoanalytical and neuro-scientific models to automation. In: *Proc. International Conference on Intelligent Environments.* pp. 111–118.

Dietrich D., Russ G., Tamarit C., Koller G., Ponweiser M., Vincze, M. (2001). Modellierung des technischen Wahrnehmungsbewusstseins für den Bereich Home Automation. In: *e&i*. Vol. 11, pp. 454–455

Dornes M. (2001). Der kompetente Säugling - Die präverbale Entwicklung des Menschen. *Fischer Taschenbuch Verlag*

Pratl G., Palensky P. (2005a). The project ARS – the next step towards an intelligent environment. In: *Proc. IEE International Conference on Intelligent Environments*, pp. 55–62.

Pratl G., Penzhorn W., Dietrich D., Burgstaller W. (2005b). Perceptive awareness in building automation. In *IEEE International Conference on Computational Cybernetics (ICCC)*, pp. 259–264. Publisher: IEEE.

Pratl G. (2006). Processing and symbolization of ambient sensor data. *Ph.D. theses, Univ. of Technology Vienna*

Pratl G., Dietrich D., Hancke G., Penzhorn W. (2007). A New Model for Autonomous, Networked Control Systems. In: *IEEE Transactions on Industrial Informatics*, Volume 1, Issue 3 (2007), p. 21 - 32. Publisher: IEEE.

Rabiner, Lawrence R., Juang, Biing-Hwang (1986). An Introduction to Hidden Markov Models. In*:* *IEEE ASSAP Magazine 3*. pp. 4–16

Tapia E. M., Intille S. S., Larson K. (2004). Activity Recognition in the Home Using Simple and Ubiquitous Sensors. In: *Pervasive*, pp. 158–175