IADIS International Journal on Computer Science and Information Systems
Vol. 6, No.1, pp. 152-163
ISSN: 1646-3692

A SMART CONTEXT-AWARE DOMOTICS SYSTEM TO PREDICT USER BEHAVIOR

Yves Vanrompay. *Katholieke Universiteit Leuven, Department of Computer Science,Celestijnenlaan 200A, 3001 Heverlee, Belgium*

Niels Pardons. *Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium*

Natalie Kcomt Ché. *Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium*

Yolande Berbers. *Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium*

ABSTRACT

Home automation or domotics systems are ambient intelligence systems that are designed to help people proactively, but sensibly. In this paper we propose a system that learns and automates patterns in the interactions of the user with the home automation devices. We show our approach and architecture. An event processing tool is used to handle the events from the home automation devices, prediction algorithms predict the next action and both rule-based algorithms as reinforcement learning decide which actions are suitable to be automated. We show the results of our system on both a synthetic data set and a real data set. The automation system manages to automate a significant number of interactions for the user.

KEYWORDS

domotics, ambient intelligence, embedded, user action automation

1. INTRODUCTION

Today many homes are filled with home automation devices and sensors. These work together intelligently to increase the user's comfort by automating actions. The home is then called a smart environment. Mark Weiser described a smart environment as follows: "a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network" (Weiser, 1991). Strongly related to smart environments is the concept of ubiquitous computing. Ubiquitous computing is a paradigm that represents computer technologies that are present everywhere and almost invisible to the user. A smart environment is an example of ubiquitous computing. To realize a smart environment the principles of ambient intelligence (AmI) need to be adopted. A possible definition for an AmI system is as follows: a digital environment that proactively, but sensibly, supports people in their daily life (Cook, 2009).

A person who lives in a house often uses the same devices every day and in the same order. The goal is to learn, recognize and automate these patterns of interactions with the various home automation devices. Automating user patterns and making predictions about future user actions increases user comfort and could make life easier for elderly and disabled. A classic example of such a pattern is to wake up, turn off the alarm clock and then make coffee. An intelligent system could learn this pattern and automatically make coffee in the morning, when the person turns off the alarm clock. The main assumption made by the system is that humans indeed follow such patterns which make reliable predictions possible.

To realize such an intelligent system the various domotics devices and sensors in the smart environment need to offer services over an intern network. Furthermore the inhabitants need to be observed for a while. Based on these observations the intelligent system can learn patterns, which can be used to predict and automate user actions. The main challenge for such a system is to predict and automate enough actions in a timely fashion, avoiding that the user manually has to start these actions, while preventing to automate actions that the user does not want. An intelligent learning system should be unobtrusive and when there is small noise, because of the users not following patterns, the system should not automate the action and should also detect and ignore this noise.

The following is a possible user scenario for an intelligent system. "Somewhere in a house in the future Jane wakes up and turns off the alarm clock. The curtains open automatically. Jane gets up and goes to the bathroom. The light turns on automatically as soon as she enters. The system has learned that she always showers in the morning and the water is heated to the preferred temperature. After showering Jane goes to the kitchen where her black coffee is already made. The system has learned that she prefers black coffee in the morning."

This paper is organized as follows. In section 2 we explain the architecture of the proposed system. Results of our experiments are presented in section 3 while section 4 discusses related work. Conclusions follow in section 5.

2. THE INTELLIGENT DOMOTICS SYSTEM

In this section the high-level architecture of the learning automation system (Kcomt Ché, 2010) is presented, followed by a detailed description of the various components.

2.1 High-level Architecture

The learning automation system needs to be able to send actions to the domotics devices. The principles of Service Oriented Architecture (SOA) are adopted for the creation of the services provided by the various domotics devices and sensors in the environment. To maximize platform independence these services are offered using WSDL, an XML based language to describe web services. Each of the devices and sensors need to offer its services using WSDL over an intern network. These web services make it possible to easily describe the services and allow devices of different manufacturers to communicate via this standardized language. It also works independently of the underlying hardware. Furthermore, the principles of Event Driven Architecture (EDA) are incorporated as they are used for systems that send events between independent software components and services. Events are generated by domotics devices and sensors in the environment and the learning system must be able to respond to these event. The communication in an EDA follows the publish-subscribe pattern, i.e. communication of events is asynchronously and possibly one-to-many.

The architecture of the system presented in this paper thus uses a combination of EDA and SOA, called event driven service oriented architecture (ED-SOA) (Ghalsasi, 2009). Here EDA is used to expand SOA with the publish-subscribe communication pattern and the ability to collect events over a long period of time, after which they can be processed and correlated. SOA is needed because the domotics devices provide services to each other and to the user. EDA handles the events that the devices generate asynchronously and that need to be collected over a long period of time.

2.2 Detailed Design: The Information Layer

Figure 1 shows the design of the learning domotics system, consisting of four different layers. The physical layer contains the various sensors and home automation devices. The communication layer provides communication between the different devices. The information layer retrieves information and higher level abstraction from the sensors and home automation devices. Finally, the decision layer controls the devices. We will first discuss the information layer and then the decision layer.

The information layer includes the event processing tool, Esper (Esper), which receives, processes and forwards the events to the other components. Esper can do filtering, but also aggregates events to represent hierarchical or composite events. The prediction algorithms in this layer will predict the next action based on recent history. We use the Jacobs-Blockeel algorithm (Jacobs, 2003) and the FxL algorithm (Hartmann, 2007) for the prediction. Jacobs-Blockeel (JB) is an algorithm that is based on IPAM (Davison, 1998). IPAM uses a first order Markov model, i.e. the prediction made is based solely on the most recent event of the input sequence. Jacobs-Blockeel uses a mixed order Markov model instead to calculate the probability distribution for the next event. Initially first order Markov models are used. Whenever the algorithm makes a correct prediction, a higher order is activated, otherwise the

order does not change. For example if the system correctly infers the rule that after observing 'a', 'b' will be observed then the system treats 'b after a' as a single observation. The advantage of using a mixed order Markov model is that high orders are only used when necessary, which benefits the storage and processing requirements. Jacobs and Blockeel claim that the highest order for the Markov model is not always the best choice to determine the probability for the next event. We chose for the Jacobs-Blockeel algorithm because it is based on IPAM, which yields very good results, but it also extends the algorithm by taking into account more than only the most recent event. In a home automation environment there are lots of possible actions to take and using a system such as IPAM that only takes the most recent action into account is not sufficient here. Furthermore the Jacobs-Blockeel algorithm mixes orders intelligently, treating frequently occurring sequences as single observations.

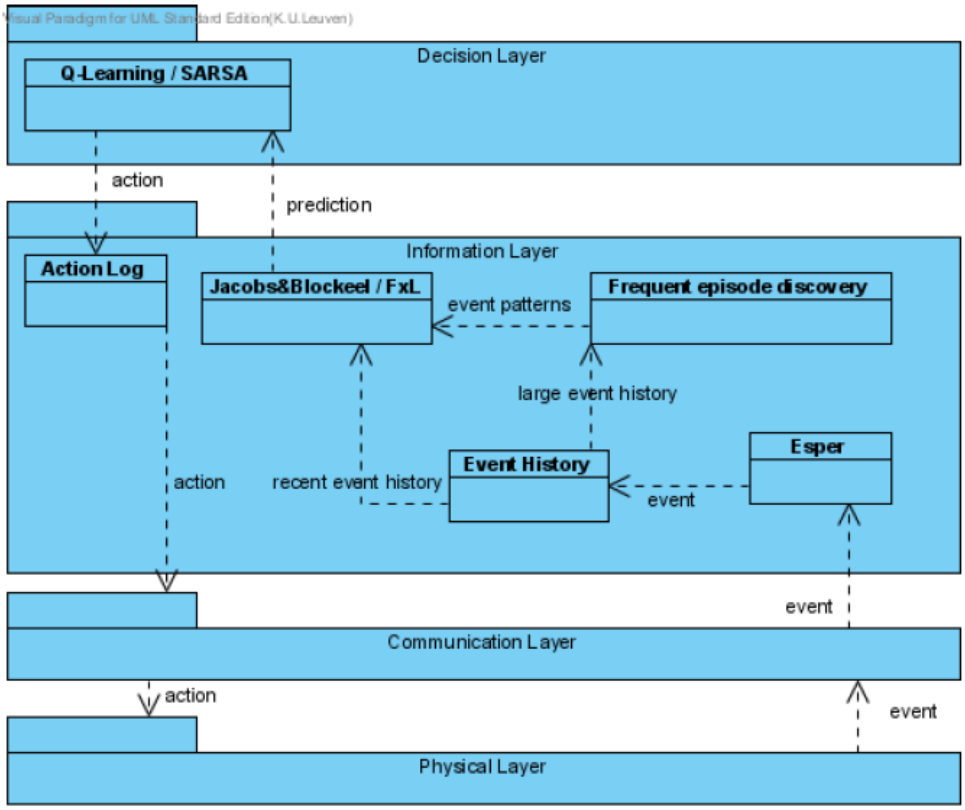

Figure 1. The architecture of the learning domotics system

FxL is an approach for combining the results of different order Markov models. The algorithm is based on an n-gram tree that contains the frequencies of different input subsequences with a length up to a specified value k. An n-gram tree is an ordered tree data structure of n-grams. An n-gram is a subsequence consisting of n items of a given sequence. These n-gram models are combined with weights to assign a score to each symbol, which represents the probability that a symbol appears next in the input sequence. AFxL is a variant

of FxL that uses a slightly different weighting scheme. This relatively new algorithm yields very good results in the context of predicting commands in a Unix terminal. FxL was chosen because it is able to get these good results while keeping the storage costs limited by the specified k and the amount of possible user actions, whereas other algorithms like Active LeZi, with similar results, have storage costs that grow with the dataset size.

2.3 Detailed Design: The Decision Layer

The decision layer contains the decision algorithms to determine when to execute which action. To be able to keep the problem tractable, the decision algorithms can only decide to automate an action after an event has arrived from a domotics device or sensor. Two reinforcement learning algorithms were tested: Q-learning and SARSA. Also three rule-based algorithms were tested: based on a prediction-rule, distance-rule and confidence-rule.

The rule-based systems only use the prediction done by the prediction algorithm to determine which action to automate. Suppose *a* is the action that is predicted with the highest probability, P_a. The prediction-rule system will automate action *a* if P_a lies above a specified threshold parameter. Suppose that *b* is the action that is predicted with the second highest probability, P_b. The distance-rule system will automate action *a* if P_a - P_b lies above a specified threshold parameter. Suppose that the prediction algorithm made a prediction with a probability larger than a specified *eps* for *m* actions. The confidence-rule system (Vanrompay, 2010) will automate action *a* if $(P_a - 1/m)/(1 - 1/m)$ lies above a specified threshold parameter. A special case occurs when $m = 1$: then this formula is not used but simply P_a.

Next the reinforcement learning algorithms are discussed. The predicted action of the prediction algorithm is recorded in the state of the Q-learning algorithm together with the last events in the system and the time and date. The Q-learning algorithm is a reinforcement learning algorithm that is able to compare the different actions to take without explicitly modelling the whole environment with very good results. SARSA is a variant of Q-learning that does not make the assumption that the optimal policy can be followed. The reinforcement algorithm receives rewards by implicit or explicit feedback from the user. Based on these rewards the algorithm can learn to correctly automate user actions. Initially the algorithm takes random actions or the action that is predicted with the highest probability by the prediction algorithm to learn which actions in which states lead to good rewards. In a later stage the algorithm will prefer actions that lead to good rewards. The final action to be executed is sent by the decision algorithm to the home automation device in question. This layer works on top of the prediction system as an extra precaution to protect against possible errors the prediction system might make.

3. EXPERIMENTAL RESULTS

The system was tested using different data sets. The first data set was created by using a synthetic data generator (Cook) and modified to support some extra options. The synthetic data generator generates events based on a user scenario. The scenario used here represents 70 days of a user living a repetitive life with no irregularities: the patterns during the weekdays differ from those during the weekend. The data set consists of 2582 events generated by 20 devices during this period. We will refer to this data set as SDG. The second data set is the

MavLab data set. This data set is created by the MavHome (Cook, 2003) project and is based on real data from a real environment, namely a workspace consisting of different areas and equipped with various sensors and domotics devices. The data set was collected by gathering the events from the domotics devices and sensors during two months while six students worked in the lab on a regular basis. This dataset consists of 1362 events from 111 possible event types.

To evaluate the system we use two measures. The first measure represents the accuracy of the system. This is the ratio of the number of correctly automated events to the total number of automated events. The second measure is the correct automation rate. This is the ratio of the number of correctly automated automatable events to the total number of automatable events. Automatable events are events that represent interactions between the user and a domotics device that can be automated by the system. Both these measures are important and they are sometimes conflicting: a higher accuracy can lead to a lower correct automation rate.

Figure 2 compares the different prediction algorithms on MavLab while using the prediction-rule decision algorithm. The different accuracy and correct automation pairs are shown for the threshold parameter ranging uniformly from 0 to 1 with steps of 0.01. The data points are connected with a straight line as an interpolation to show where accuracy and correct automation rate would lie when using thresholds between the ones tested. Graphically the line of the best system lies to the right and above of the other lines. This isn't the case here: the lines cross each other. There is no clear winner. Jacobs-Blockeel is the best for higher accuracies while for lower accuracies Jacobs-Blockeel, FxL and AFxL lie closely together. IPAM performs significantly less than the other three systems, which justifies choosing (A)FxL and Jacobs-Blockeel. Jacobs-Blockeel and IPAM both clearly make a trade-off between accuracy and correct automation rate when changing the threshold value. When the threshold parameter is high the accuracy is high, but the correct automation rate is low.

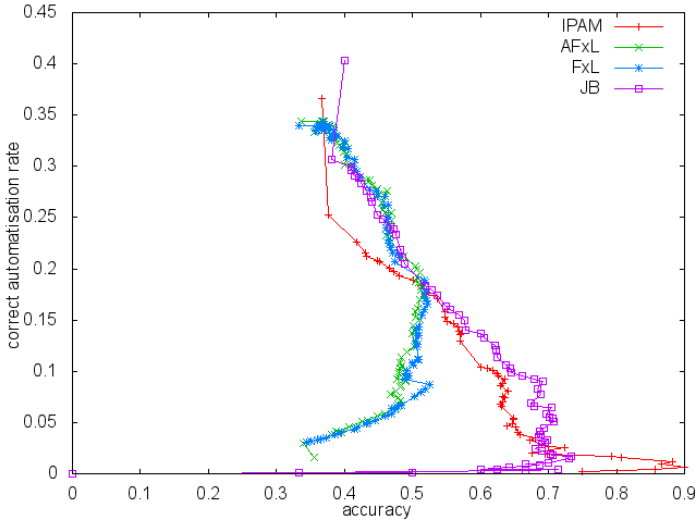

Figure 2. Comparison of prediction algorithms on MavLab using prediction-rule decision algorithm

Figure 3 compares the different decision algorithms on SDG while using the Jacobs-Blockeel prediction algorithm. Again the rule-based algorithms are shown by a line connecting the accuracy and correct automation pairs for various thresholds.

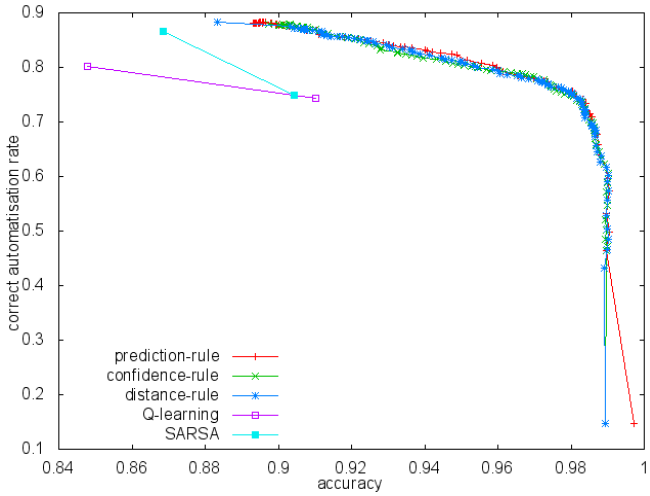

Figure 3. Comparison of decision algorithms on SDG using Jacobs-Blockeel prediction algorithm

The reinforcement learning algorithms do not make a clear trade off when changing a certain parameter. In our experience the results lie closely together. For Q-learning and SARSA we just show two data points: the one with the maximum accuracy and its correct automation rate and the one with the maximum correct automation rate and its accuracy. The results for Q-learning and SARSA are clearly worse than the three rule-based algorithms. Q-learning and SARSA itself lie closely together. The results for the three rule-based algorithms are quite high: it is for example possible to get 75% correct automation rate at 98% accuracy. These high results are explained by the fact that there is no noise in this data set. Between the three rule-based algorithms there is no clear winner: the lines lie closely together and intersect often.

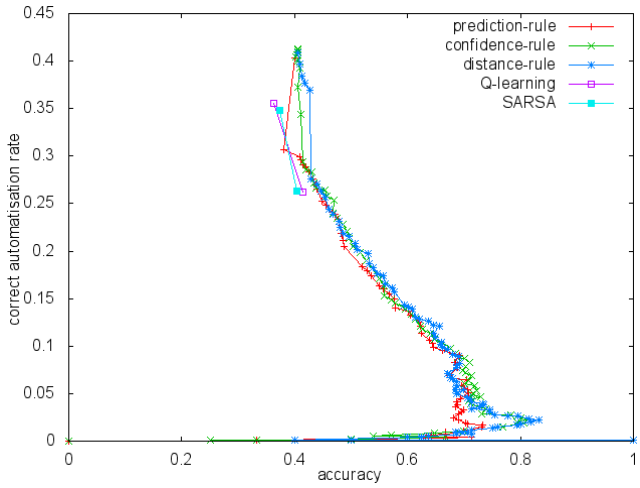

Figure 4. Comparison of decision algorithms on MavLab using Jacobs-Blockeel prediction algorithm

Figure 4 is similar to Figure 2 and compares the different decision algorithms on MavLab while using the Jacobs-Blockeel prediction algorithm. Again the reinforcement algorithms are worse than the three rule-based algorithms. There is again no clear winner between the three rule-based algorithms. MavLab is a real data set so the results are lower here: it is for example possible to get 10% correct automation rate at 66% accuracy.

4. RELATED WORK

There already exist some projects that try to analyze, predict and automate user actions in a smart environment. There are three big projects that try to achieve this: The Adaptive House, iDorm and MavHome. The Adaptive House (Mozer, 1998) tries to automate lights, warm water, heating and ventilation. It uses artificial neural networks to predict the next state of the house. It balances energy cost and discomfort cost using Q-learning to determine which action to automate. The iDorm (Hagras, 2004) project tries to automate domotics devices in a dorm using techniques from fuzzy logic. Fuzzy membership functions are retrieved from the collected data. These are then used to construct fuzzy rules that capture the user behavior. When a user overrides a setting that the system has automated, the weights of the fuzzy rules are adapted or new fuzzy rules are created. MavHome (Cook, 2003) uses Episode Discovery (ED) to discover frequent patterns in the usages of devices. Furthermore a prediction algorithm, Active Lezi (ALZ) is used to predict the next user action. The frequent patterns discovered through ED are used to improve the accuracy of the ALZ algorithm. The decision algorithm is TD(0) reinforcement learning, which determines what action has to be executed.

The various projects-MavHome, iDorm and The Adaptive House will now be compared on the basis of criteria that are important for learning home automation systems.

Prediction approach: The three projects each use a completely different technique to make predictions. MavHome uses the data mining algorithm ED and the sequential prediction algorithm ALZ to make a prediction. The Adaptive House in turn makes use of artificial neural networks and iDorm makes predictions based on fuzzy rules.

Architecture: There are major architectural differences between the three projects. The architecture of MavHome is layered and modular, making it easy to incorporate other techniques. The Adaptive House also has a modular, but not a layered architecture. The disadvantage is that there is always a slightly different architecture needed for the different subsystems. So the architecture of the system for the lighting control differs from the architecture of the system that controls the heating. The architecture of iDorm in contrast exhibits a very tight coupling, making it difficult to incorporate other learning techniques.

Evaluation: Also regarding the evaluation of the three projects there are differences. The MavHome approach has been thoroughly tested in two real environments: a student home and a laboratory with different rooms. The system has been tested with different people and the ability of the system to adapt to changing user patterns is tested. However, the experiments are limited to the automation of the lighting. The Adaptive House is also tested in a real environment, being a house with several rooms. The evaluation is rather limited, e.g. because it is not evaluated whether the system can adapt to changing user patterns. Also, the system is not tested with different people. Lastly, iDorm has been tested in a student house with one room and a limited number of devices. The approach is evaluated by comparing with other systems. These systems are not adapted to the problem domain, what makes the comparison not representative.

Computational complexity: MavHome uses ED, which requires extra processing time. This data mining algorithm requires running several times over the entire database of events. The Adaptive House employs artificial neural networks. If these neural networks are large, they are computationally intensive. The approach by iDorm embeds agents on small embedded devices with limited memory and processing power, and must therefore take into account the limited resources. Thus, the techniques used in the Fuzzy Logic Controller (fuzzificator singleton, triangular membership function, product inference, max-product composition and height defuzzification) are known for their computational simplicity and real-time considerations. Furthermore, they use a one-pass method to construct the fuzzy rules which is not computationally intensive.

Learning speed: MavHome uses the ED data mining algorithm, for which a large amount of data must be collected before it can be used. Therefore the use of ED will be at the expense of the learning speed. The Adaptive House uses artificial neural networks trained with backpropagation, which exhibits very slow learning. The evaluation of iDorm showed that the system can quickly learn user patterns.

Unsupervised learning: The learning of the prediction model in MavHome is completely unsupervised. To learn which decisions are good and which not the reinforcement learning algorithm TD (0) is employed. If the user manually changes an automated action, which indicates a lack of comfort for the user, this is seen as negative feedback and consequently TD(0) receives a negative reward. Corrective user decisions are made so the learning is semi-supervised. The forecasting system used in The Adaptive House works unsupervised. The system learns based on observed data patterns in a house. But by using reinforcement learning, specifically Q-learning, to learn which are decisions are good and which bad, this means that the user can manually override an automatic action. The Q-learning algorithm receives rewards for decisions that are made based on the total cost (energy cost and discomfort) of the status of the house. Learning to make decisions is also semi-supervised. iDorm works completely unsupervised. The system learns based on observed data and a fuzzy logic controller that the user behavior approaches. If the user manually adjusts an automated action, existing rules are amended or new rules added.

Ability to learn changing user patterns: In MavHome there are facilities for the learning model to be adjusted if it is necessary to construct new user patterns. These features are also tested, showing that the system actually adapts to changing user patterns. In the Adaptive House, there are no changing user habits and hence nothing new has to be learnt. iDorm has the ability to learn new user patterns. This is done by modifying existing rules or adding new rules. The evaluation of the system showed that adaptation to new patterns is no problem.

Response time: None of the three projects report the response time results. But they are all used in real environments and thus should manage to keep the response time low.

Automation rate / error rate: The only project that gives numbers on the automation rate is MavHome. The error rate is only reported by iDorm and is 12%. The Adaptive House is not talking about automation rate or error rate, but experiments have shown that energy costs were indeed reduced.

Scalability: In none of the three projects, the scalability of the approach is tested. Based on the architecture and the techniques however we can get an idea on how scalable the approaches are. The use of ED in MavHome leads to scalability problems. ED is a computationally intensive process, which can cause problems in large environments. The system used in The Adaptive House is far from being scalable. Each different comfort goal (lighting, heating, ventilation ..) is a separate system with a different architecture. Also, parts of the system must be manually adjusted depending on the type of comfort system. iDorm does not give enough information to be able to assess the scalability.

Memory consumption: The episode discovery algorithm in MavHome consumes a lot of memory. The prediction algorithm used ALZ and consumes a lot of memory as the storage costs increase with the size of the dataset. The Adaptive House makes extensive use of artificial neural networks. The memory usage of these networks depends on the size of the neural network and the actual implementation. This data is not available. In iDorm memory is reduced by keeping the number of rules that are stored to a minimum. Only the most frequent and most recent rules are kept. This is possibly at the expense of user comfort.

This comparison is summarized in the table below.

	MavHome	**Adaptive House**	**iDorm**
Prediction approach	ALZ and ED	Neural network	Fuzzy rules
Architecture	++	-	--
Evaluation	+	+-	-
Comput. complexity	-	?	+
Learning speed	+-	+-	+
Unsupervised learning	+-	+	++
Changing patterns	+	--	+
Response time	++	++	++
Automation/error rate	++ / ?	? / ?	? / +
Scalability	-	-	?
Memory consumption	-	?	+

It should be noted that the architecture of the system presented in this paper is inspired on the architecture of MavHome. However, the system described here differs from MavHome in

the use of an event processing tool to process events in a structured manner. Also other learning techniques are used. Here ED is not employed for scalability reasons and FxL and Jacobs-Blockeel are used as prediction algorithms instead of the ALZ algorithm. The decision algorithm also differs. Here Q-learning, SARSA and rule-based algorithms are used instead of TD(0).

5. CONCLUSIONS

Home automation devices, sensors and smart devices are more and more deployed into the houses of people. Ambient intelligence is defined as a digital environment that proactively, but thoughtfully, supports people in their daily lives. In this paper, a self-learning context-aware automation system was proposed. The aim is to automate the interactions of users with home devices. The main challenge system is to automate sufficient interactions in limited time with minimal false automations. Additional challenges are the ability to rapidly adapt to changing usage patterns, scalability and making the system suitable for use in an embedded device with constrained memory and CPU power.

This paper proposed a system that relies on the principles of Event-Driven Service-Oriented Architecture (ED-SOA). The self-learning home automation system needs to communicate with different devices, which requires an agreed protocol. For this, the principles of SOA are adopted where it is assumed that the various home automation devices offer their services through the XML-based open standard WSDL. To keep the problem tractable, the system will consider whether an interaction has to be automated only after a user has effectively interacted with a home automation device. The system decides whether an interaction will be automated mainly relying on the most recent interactions or sensor events. The self-learning system developed here consists of three components. First, there is an event processing tool, Esper, which receives and filters events of the home automation devices and sensors in a structured way. Esper can apply various operations such as aggregation, sorting and separation of events. In the second step, a sequence prediction algorithm is used: this is an algorithm which predicts the next event based on the history of the sequence of events of the home automation devices. In this work, four algorithms were tested: IPAM, FxL, AFxL and Jacobs-Blockeel. In the third phase the decision algorithm will decide whether an interaction will be automated and if so which one. Three algorithms were tested which consist of simple rules to decide whether the prediction will be effectively carried out. Furthermore, two reinforcement learning algorithms, Q-learning and Sarsa, were tested as decision algorithms. These two reinforcement learning algorithms were tested on their effectiveness with or without the sequence information of the prediction algorithm.

The system was tested on both synthetic and real data sets to evaluate whether the established requirements could be met. The results on a synthetic data set with no noise are good concerning accuracy. On synthetic data sets with excessive noise or on the real data sets the accuracy and correct automation rate are significantly lower. It is doubtful whether users would adopt an autonomous system to automate interactions if the accuracy is so low. In the tests it became clear that the decision algorithms based on Q-learning and Sarsa perform worse than the rule-based decision algorithms. The accuracy and automation rate in the latter group is significantly higher. None of the rule-based decision algorithms is superior to the others in terms of accuracy and amount of correct automations. The experiments also showed

that the Jacobs-Blockeel and AFxL sequence prediction algorithms outperform FxL and IPAM. Additionally, he various approaches were tested for their adaptability to changing user patterns. Because of the high degree of randomness in Q-learning and Sarsa, it proved to be very difficult for these algorithms to take into account changing patterns in a well-tuned manner. It is clear that the exploration algorithm used here is of significance. The rule-based decision algorithm adapts quickly to new or changing user patterns. Concerning the sequence prediction algorithms, IPAM and Jacobs-Blockeel are best able to adapt to changing patterns. These two algorithms have a parameter that allows to make a tradeoff between rapid adaptability and accuracy. For an embedded system with constrained memory and CPU power, the rule-based algorithms are preferable to Q-learning and Sarsa since they do not store additional information. If the available memory is very small, one should opt for IPAM as a sequence prediction algorithm. The various systems approaches scale very well in terms of time and memory complexity as the number of devices or number of interactions increases: the increase is usually sublinear. This is important since it is expected that in the future the number of sensors and automation devices in homes will exhibit a major increase.

REFERENCES

Cook, D. et al, 2003. MavHome: An Agent-Based Smart Home. Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), pp. 521.

Cook, D. and Das, S., 2004. *Smart Environments: Technology, Protocols and Applications*. Wiley-InterScience.

Cook, D., 2009. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive Mobile Computing*, Vol. 5, No. 4,pp 277-298.

Cook, D., *Synthetic Data Generator*. http://ailab.wsu.edu/mavhome/datasets/sdg.zip

Davison, B. and Hirsch, H., 1998. Predicting sequences of user actions. *Proceedings of AAAI-98/ICML-98 Workshop Predicting the Future*, pp. 5-12.

EsperTech Inc. *Esper, Complex Event Processing*. http://esper.codehaus.org/

Ghalsasi, S. Y., 2009. Critical success factors for event driven service oriented architecture. *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human.* Seoul, Korea, pp. 1441-1446

Hagras, H., 2004. Creating an Ambient-Intelligence Environment Using Embedded Agents. *IEEE Intelligent Systems*, Vol. 19, No. 6, pp. 12-20.

Hartmann, M. and Schreiber, D., 2007. Prediction Algorithms for User Actions. *Proceedings of ABIS* 2007, pp. 349-354.

Jacobs, N. and Blockeel, H., 2003. Sequence Prediction with mixed order Markov chains. *Proceedings of the Belgian/Dutch Conference on Artificial Intelligence.*

Kcomt Ché, N. et al, 2010. An intelligent domotics system to automate user actions, *Advances in Intelligent and Soft Computing*, Vol. 72, pp. 201-204.

Mozer, M., 1998. The neural network house: an environment that adapts to its inhabitants. *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, pp. 110-114.

Vanrompay, Y. et al, 2010. An effective quality measure for prediction of context information. *Proceedings of the 7th IEEE Workshop on Context Modeling and Reasoning (CoMoRea) at the 8th IEEE Conference on Pervasive Computing and Communications (PerCom'10)*. Mannheim, Germany.

Weiser, M., 1991.The computer for the 21st century. *Scientific American*, Vol.265, No.3, pp. 66-75.