Adapting to Resident Preferences in Smart Environments

Parisa Rashidi, Diane J. Cook

Washington State University
Pullman, WA
USA, 99163
{prashidi, cook}@eecs.wsu.edu

Abstract

In the past decade, smart home technologies have been a topic of interest for many researchers with the aim of automating daily activities. However, despite increasing progress in this area, less attention has been paid to smart environments that can adapt to changes in residents' preferences over time. Here we introduce CASAS, an adaptive smart home system that discovers and adapts to changes in the resident's preferences in order to generate satisfactory automation policies. The adaptation capability of CASAS is achieved by utilizing data mining methods as well as learning strategies that adapt to the resident's explicit and implicit preference feedback. In this paper, we present a description of the adaptation models employed by CASAS together with the results of experiments applied to both synthetic and real smart environment data.

Introduction

In recent years, smart homes have been a topic of interest for many researchers with the aim of automating home activities to achieve greater comfort, productivity, and energy efficiency (Abowd 2005, Helal 2005, Youngblood 2007). In a smart home, networked sensors and controllers try to assist residents' lives by acquiring and applying knowledge about residents and their physical surroundings (Cook 2004). The long-term goal of many smart home projects is to automate resident interactions with the environment that are repetitive or, in the case of individuals with physical limitations, are difficult to perform. To achieve a successful automation, it is important to consider the resident's preferences regarding different aspects of automations, such as events and their relative sequential order, each event's start time, and its duration. Unfortunately, despite progress in smart home research, less attention has been paid to the design of a smart home that can adapt to changes in the residents' preferences over time.

One of the few works done in this area is a primitive approach proposed by Vainio et al. (Vainio 2007), where a fuzzy controller adjusts the weights of a few predefined factors that determine whether performing a specific action is in accordance with the user's preferences or not, e.g. turning on the lamp if the light level is less than a certain amount. However, they do not consider the problem of adapting to more complex sequential activities, and do not discuss how such changes can be discovered from daily activity data. In addition, they consider a set of predefined factors, but in the real world, it is not possible to determine such factors in advance.

In our work, we consider automation of sequential complex activities that adapts to the user's preference. Our approach does not make any assumptions about the activity structure or other underlying model parameters, but leaves it completely to our algorithms to discover relevant patterns. This significantly increases the flexibility of our approach, particularly since we allow for different aspects of the resident's preference to be modeled, such as the activity structure and sequence order, start time, and duration. In addition, our model takes into account the resident's feedback by considering various levels of user involvement; the resident can guide the system by providing explicit preference feedback, or s/he can leave it to the system to automatically discover changes in preferences.

Input to CASAS consists of various sensor data collected by the smart environment, such as motion sensors, light sensors, and so forth. This data is mined by our mining algorithm called FPAM to discover activity patterns of interest for automation; and later these patterns are modeled by our Hierarchal Activity Model (HAM) to further utilize the underlying temporal and structural preferences. To achieve preference adaptation, the Preference Adaptation Miner (PAM) algorithm adapts to any changes in those patterns and responds to user guidance. The preference adaptation mechanism is based on using four different mechanisms: direct manipulation, guidance, request, and smart detection. Each method requires a certain balance between user collaboration and system autonomy, from no collaboration in smart detection method to no autonomy in direct manipulation method. Putting all the preference adaptation methods together, we are able to provide an adaptive smart environment solution which adapts to its residents' preferences over time, and which provides a wide range of collaboration and feedback methods such that residents can choose to act proactively or be passive.

FPAM Model

CASAS's first step to automate daily activities in a smart home is to discover user preferences regarding different aspects of automated activities. To achieve this, our mining algorithm called the Frequent and Periodic Activity Miner (FPAM) discovers patterns in the resident's daily activities that may provide a basis for automation. FPAM also learns basic temporal information associated with the patterns that later will be used by HAM to extract more comprehensive temporal information. In our discussion, we define an event as a single action such as turning on the light, while an activity is a sequence of such events, e.g. turning on the light – turning on coffee maker, which is composed of two events. The events can represent data generated by different sensors such as motion sensors or by a device that is manipulated by a powerline controller. Discovering how the resident performs these activities in daily life facilitates dedication of the resident's preferences for home automation. We consider frequent and periodic activities, as automating these activities makes the environment responsive to the resident. Such automation also removes the burden of manually performing these long repetitive sequences, and opens up the possibility for the environment agent to find better ways of accomplishing tasks, such as a more energy-efficient method for heating heavily-used portions of a home. FPAM is able to find frequent and periodic activity patterns with inexact periods, arbitrary length, and predictable timing constraints. Such a flexible mining method makes it easy to discover resident's preferences about structure, period and temporal constraints of activities.

We assume that the input data is a sequence of event tuples in the form of $<d_i, v_i, t_i>$, where i represents a single event, d_i denotes the event's data source (e.g., motion sensor), v_i denotes the state of the source (e.g., off, on), and t_i denotes the time that event i occurred. A window of size ω (initialized to 2) is passed over the data and every sequence of length equal to the window size, is recorded together with its frequency, f_i. (i.e., number of times the sequence occurs in the dataset) and initial periodicity (i.e., regularity of occurrence, such as every three hours)

After the initial frequent and periodic sequences have been identified, FPAM incrementally builds candidates of larger size. Instead of scanning the data again, FPAM extends sequences that made the cutoff in the previous iteration by one event that occurs before or after the current sequence. FPAM continues to increase the window size until no more frequent or periodic sequences within the new window size are found. Drawing on results from information theory, we evaluate the frequency of a sequence based on its ability to compress the dataset, by replacing occurrences of the pattern with pointers to the pattern definition (Rissanen 1989). We compute the compression value according to Equation 1, where f_a represents frequency of sequence a, t represents the input data length in hours and C represents the compression threshold. Therefore for a sequence to be considered as frequent, the following condition should hold:

$$\frac{|a| * f_a}{t} > C \quad (1)$$

To calculate the periodicity of each pattern, FPAM computes elapsed time between consecutive occurrences of the pattern. Two periodicity granules are considered: coarse-grained expressed in number of days and fine-grained expressed in number of hours. To construct periods, a lazy clustering method is used; if a sequence's period does not match previous periods (with a tolerance of one hour for fine-grained and one day for coarse-grained periods), a new candidate period is constructed. If the periodicity of a sequence is consistent a threshold number of times, the pattern is reported as periodic.

Hierarchical Activity Model: HAM

After activity structure and periods have been discovered by FPAM, the sequences will be organized in a Hierarchal Activity Model (HAM) structure, which filters out activities according to two temporal granule levels of day and hour (see Figure 1). Such an organization helps to deduce temporal preferences from basic temporal information provided by FPAM in form of timestamps. HAM also captures temporal relationships between events in an activity by explicitly representing sequence ordering in a Markov model. Each activity will be placed in a HAM leaf node according to its day and time of occurrence. For each activity at each node, we describe the start time and the duration of each individual event using a normal distribution that will be updated every time FPAM mines newly-generated data. If an activity is located in different time/day nodes within the HAM model, multiple Gaussian distributions for that activity will be computed, allowing HAM to more accurately approximate the start time and duration values, by using multiple simple local functions.

After the hierarchical model is constructed, HAM considers appropriate activities to be automated. Such automation should reflect our preference adaptation policy, such that recently discovered activities will not miss their chance of being selected for automation simply because enough time has not passed to allow them be explored by the learning algorithm. To achieve this, we require the following conditions to hold: 1) the highest-likelihood activities are given a greater chance of being automated, 2) less likely activities retain a chance of being automated (especially recently discovered ones) and 3) the temporal relationships between activities are preserved. CASAS is not currently designed to handle the intricacies associated with interleaving multiple automation activities that

overlap in time. As a result, not all actions that appear in the hierarchy for a particular time can necessarily be automated.

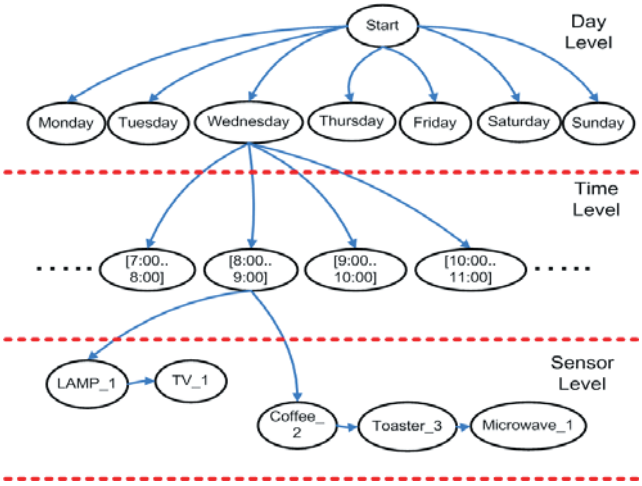

Figure 1. Example HAM model.

HAM selects activities for automation that maximize the expected utility (Sutton 1998), as shown:

$$EU(A) = P_T(A)Q(A) \quad (2)$$

In Equation 2, $Q(A)$ is A's potential value which is the same for all of its component events, and $P_T(A)$ is the total occurrence probability of activity A, defined as:

$$P_T(A) = P_d(A) \times P_t(A) \times P_r(A) \quad (3)$$

In the above equation, $P_d(A)$, the daily probability, reflects the occurrence probability of A on a given day of the week. The time probability, $P_t(A)$, reflects the occurrence probability of A in a given time interval (node); and the relative probability, $P_r(A)$, reflects the occurrence probability, with respect to the other activities that fall within the same time node. The daily and time probabilities are estimated using frequencies computed by FPAM and the relative probability is computed using P_{di} and P_{ti}.

To select an activity, CASAS balances between exploration and exploitation where exploring potential automated activities allows for potential improvement of the smart home, and exploitation avoids user frustration due to too many wrong choices. The probability of selecting a particular activity A for automation is calculated according to Equation 4. Here $EU(j)$ is the expected utility of activity j as defined before, $\beta * D(A)$ is a term that favors recently-added activities, $D(A)$ represents how recently activity A occurred (as a reciprocal of the number of days since its discovery); β adjusts the relative importance of recently-added activities vs. highly expected activities; and k is a parameter which initially is set high to promote exploration, but over time decreases, to allow for exploitation of stabilized automations.

$$P(A) = \frac{k^{EU(A)+\beta*D(A)}}{\sum_j k^{EU(j)+\beta*D(j)}} \quad (4)$$

Dynamic Adaptation

Most smart environments assume a static learned model, i.e., once the resident's activity preference patterns have been learned, no changes are applied to maintain the model over time. However, as we know, humans are likely to change their preferences over time, depending on many factors, such as social relations, seasonal and weather conditions and even emotional states. Therefore, a static model can not serve the purpose of a long term solution for a smart home; instead, we need to find a way to adapt to the changes that occur over time. Such adaptation should occur in response to resident's preferences, either by observing behavior changes, or by adopting explicit advices given by users.

CASAS achieves adaptation based on the resident's explicit preference feedback, provided through the CASAS user interface, or based on implicit preference feedback which can be described as any alteration in the resident's habits and lifestyle. For example, consider a resident that used to turn on the coffee maker every day at 7:30am, but later changes his habit and turns it on at 6:30am. This is an example of implicit user preference feedback that should be detected by CASAS.

We employ four different adaptation mechanisms to consider the resident's explicit and implicit preference feedback: direct manipulation, guidance, request and smart detection. In direct manipulation, residents provide the system with the most explicit form of preference feedback, by manipulating automated activities using the CASAS user interface. Using the guidance method, residents guide CASAS by rating the automated activities based on their preferences on scale of 1..5, thus providing CASAS with explicit feedback or advice. With the request method, residents can highlight any activity to be monitored by CASAS for possible changes, therefore providing a mixture of explicit and implicit preference feedback. In the last approach, called smart detection, CASAS utilizes the most implicit form of feedback by monitoring resident activities and updating the HAM model. The difference between the last two methods is first how fast the changes will be detected; and second the required amount of user interaction, as smart detection method doesn't require any user interaction. Using the request method, the change detection process starts immediately and hence residents do not have to wait for the regular mining schedule. CASAS uses all of these mechanisms to provide a flexible, user-centric solution to the dynamic preference adaptation problem that allows for various degrees of resident involvement. Residents can choose any of above methods to provide feedback to CASAS, and can choose to act proactively or be passive.

For every activity, we maintain a potential value, Q, which reflects the amount of evidence against or for an activity as being frequent or periodic, in other words the degree to which it should be considered for automation. The potential value can be increased or decreased through a compensation effect or a decay effect, as will be described. If potential value falls below a certain activity

threshold, the activity is discarded from the model, in other words it will be forgotten. Maintaining a potential value for each discovered activity can help us distinguish transient changes from long-term changes that still might be accompanied by a noise element. The potential value is increased by using the following formula:

$$Q = (Q + \alpha * r) \quad (5)$$

In the above equation, $r \in$ [-1...+1] denotes the preference value, and $\alpha \in [0,1]$ denotes the learning rate. To simulate the overriding nature of learning in direct manipulation, and guidance methods, we set the learning rate to a relatively high value; while for request and smart detection methods we set it to a small value to simulate their gradual history-preserving nature. Note that when updating the potential value, we do not differentiate between different events that comprise an activity and consider it as a whole; therefore we assign a single value to an activity.

In addition to the compensation effect, we also employ a decay effect which subtracts a small value ε from all activities' values at each time step θ. Applying decay function, the value of any activity during an arbitrary time interval Δt is decreased by:

$$Q = Q - \frac{\varepsilon * \Delta t}{\theta} \quad (6)$$

The decay effect allows for those activity patterns that have not been perceived over a long period of time to descend toward a vanishing value over time, or in an intuitive sense to be forgotten. This helps to adapt to changing preferences of residents. The effect of the decay function is compensated through compensation effect in a way that the potential value remains bounded.

Explicit Preference Feedback

As we already mentioned, residents can directly manipulate automated activities by changing event start times and durations to their preferred values. They can also add activities, delete activities, or modify entire activities by adding, deleting, or reorder the events that comprise the activity. Residents can also designate an activity as a priority activity that overrides other activities. We call this preference feedback method as direct manipulation approach, which involves the highest degree of user collaboration. Considering given complexity of each activity's definition, it is essential to provide the user with adequate guidance. In our system, whenever a user wants to define or modify an activity, s/he is guided through a series of wizard dialogs where each dialog asks the appropriate question based on previous steps and in each step, a brief description about that step is provided in order to help users better understand the underlying conceptual model.

A less demanding yet collaborative approach, called guidance method, works based on rating of the automated activities by users on a scale of 1 to 5, where 5 represents that the resident likes the automated activity. These ratings are mapped into corresponding preference values, to increase or decrease potential value of an activity, and help CASAS decide what activities to automate next time.

Implicit Guidance

In this method, whenever an activity is highlighted to be monitored for any changes in user preferences, a mining algorithm called Preference Adaptation Mining (PAM) analyzes recent event data and looks for any changes in the preferences pattern, such as the start time, durations, periods, or the activity structure (the component events with their temporal relationships). Without loss of generality, we refer to two different categories of preference changes: changes that preserve the structure and changes that alter the structure.

Structure change is detected by finding new patterns that occur during the same times we expect the old pattern to occur; assuming that start time can act as a discriminative attribute. First, PAM looks for an activity pattern, a, such that its start time, s_a, is contained within the interval $\Delta\delta = \mu_a \pm \sigma_a$, where μ_o and σ_a denote the mean and standard deviation of the original pattern's start time distribution. PAM is looking for different activity patterns within the same start time interval in which we expect to see the original activity pattern, and marks the beginning of all such intervals. It then moves a sliding window of size ω (initially set to 2) over the interval and incrementally increases the window size at every iteration. The window size does not increase when no more frequent or periodic patterns of length ω can be found. PAM does not examine all of the data, rather just the marked points. A frequent pattern can easily be extended beyond the marked point, as we require only its start time to be contained within the marked interval. This process results in finding a new pattern which may be longer, shorter, or have different properties than the original one.

In the case where structure is preserved, we first mark all the occurrences of the original activity in the data, and based on these occurrences calculate properties such as new durations, new start times or new periods. After results from both cases have been collected, the PAM algorithm reports the list of changes that can be accepted or rejected by the user.

Smart Detection

In the smart detection method, CASAS automatically mines data regularly to update the preference model. This approach doesn't need any collaboration from user and is slower than the three previous approaches because there is no explicit user feedback and changes might not be detected until the next scheduled mining session. After every mining session, the discovered activities will include a mixture of new and previously-discovered activities. For new activities, we simply can add them to the HAM model. For previously existing patterns, if the pattern shows no change, then PAM applies the compensation effect to indicate observation of more evidence for this pattern. However, if the pattern shows some changes, we will add

the modified patterns to the model, while also preserving the original pattern, as there is no explicit evidence that this change is a permanent preference change. To achieve adaptation in this case, we will leave it to the compensation effect and decay functions to decide over time which version is more likely. The compensation effect will increase the value of the more frequently-observed version of the pattern while the decay function will be dominating for a less-observed pattern. As a result, the value of patterns that have not been observed for a long time will fall below the activity threshold σ; and will eventually be removed from the model.

If we denote the original pattern as P and the modified version of the pattern as C, then we can calculate the number of times the decay function should be applied for P to be dominated by C. Assume at time t_i, pattern C is discovered for the first time and its potential value is assigned an initial value of Q_i^C. After time t_i, the decay function will periodically decrease the value of both patterns while Q_i^C also increases each time C is observed. Therefore, P's chance of being automated decreases while C's chance increases due to the increase in its potential value. Even as C emerges it has an opportunity to be selected for automation as a recently-discovered pattern. The potential value for pattern P, Q_u^P, after j applications of the decay function and at time t_u, will be:

$$Q_u^P = Q_i^P - \frac{\varepsilon * \Delta t}{\theta} \quad (7)$$

We also know that in order for the original pattern to be perfectly forgotten, its potential value should be below an activity threshold, i.e. $Q_l^P < \sigma$. Substituting Equation 7 into $Q_l^P < \sigma$ leads to:

$$\frac{Q_i^P - \sigma}{\varepsilon} < j \quad (8)$$

The above inequality shows the minimum number of times that the decay function should be applied to a pattern before it's forgotten. At the same time, if we consider l observation-based learning cycles due to regular mining sessions, Q_i^C will be changed as following:

$$Q_u^C = Q_i^C + l\alpha_o r_o - j\varepsilon \quad (9)$$

In order for Q_i^C to have a value greater than the activity threshold, we require that $Q_u^C > \sigma$. If we consider ΔT as an arbitrary time interval, and m as the period of regular mining (e.g. every week), then l can be defined in terms of m and ΔT, as $\Delta T/m$. Substituting Equation 9 into $Q_u^C > \sigma$ and considering ΔT and m leads to:

$$m < \frac{\alpha_o r_o * \Delta T}{\sigma + j\varepsilon - Q_1^C} \quad (10)$$

Equation 10 shows how frequently the PAM mining algorithm should be performed in order for the decay effect to be compensated by the reinforcement function.

Generalizing to Similar Cases

The above procedures showed how preference feedback can guide the automation of a specific activity, a_i. This might raise a question regarding similar activities, such as activities that have the same structure, but different time constraints. To address this problem, we consider the degree of similarity between two activities as the degree to which we generalize user preference feedback. Here we focus on attributes of duration D, start time s and period P. The similarity, Θ, is defined as shown in Equation 11.

$$\Theta = \frac{1}{\mu_{\Delta D} + \Delta s + \Delta P} \quad (11)$$

In this equation, $\mu_{\Delta D}$ is defined to be the average difference between the durations of all of the events in the two activities of interest, Δs represents the average difference in start times, and ΔP represents the average period difference. We will use the similarity degree to determine to what extent to generalize the given user feedback. To use compensation equation for activity a_j based on feedback given to a similar activity a_k, we compute its preference value according to Equation 12:

$$r_j = \beta * \Theta * r_k \quad (12)$$

In this equation, β is a tuning parameter that can adjust the degree of generalization.

Experimental Results

In order to evaluate CASAS' ability to adapt to new activity preferences, we tested it on both synthetic and real data obtained from a smart home system. We hypothesize that CASAS can adapt to changes in resident behavior patterns as well as to explicit advice. To test the hypothesis, we designed a synthetic data generator that generates event data corresponding to a set of specified activity descriptions with a specified percentage of randomly interjected events to give the data realism. In addition to synthetic data, we evaluate CASAS on real-world collected data from a physical smart environment test-bed at Washington State University. This physical test-bed is equipped with motion sensors, rug sensors, light sensors, and controllers for the lamps (see Figure 2).

For our first experiment with synthetic data, we generated one month of synthetic data with six embedded activities (see Table 1). After FPAM and HAM constructed corresponding models of these activities, we highlighted the third activity to be monitored for changes. We then changed the activity description in the data generator such that all event durations were set to 7 minutes, instead of 5 minutes. CASAS detected the changes accordingly by finding a new duration of 6.44 minutes, which is quite close to the actual 7 minute change. The data generator does have an element of randomness, which accounts for the discrepancy between the specified and detected time change. In similar tests, PAM was also able to detect start time changes from 13:00 to 13:30, and structure changes (omission or addition). These results support our hypothesis that the CASAS provides the ability for the smart home to adapt to changes in resident preference patterns.

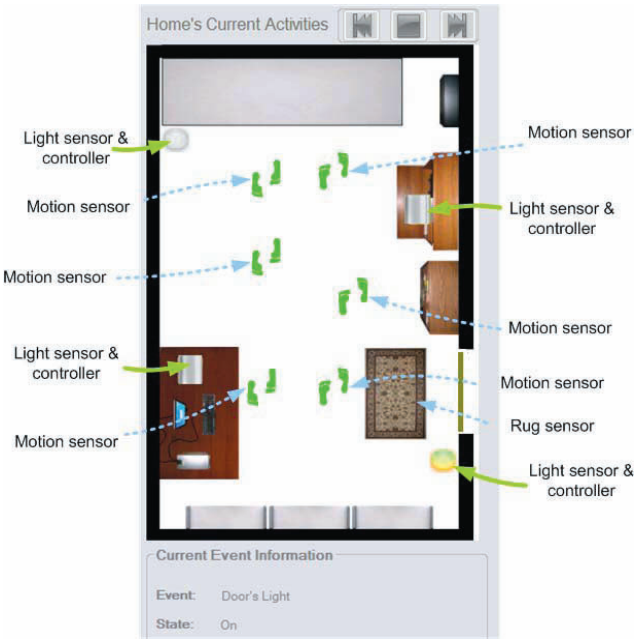

Figure 2 Sensor Layout of Washington State Univ. test-bed.

Start Time	Period (hourly)	Events
13:00	2	DoorLight, LeftDeskLamp
13:30	3	WhiteboardLight, OpenDoor
14:25	5	RightDeskLamp, WhiteboardLight
14:55	2	LeftDeskLamp, OpenDoor, RightDeskLamp
15:35	3	LeftDeskLamp, WhiteboardLight
15:55	3	RightDeskLamp, DoorLight

Table 1 Activities simulated by synthetic data generator.

In the next step, we tested CASAS on real world data using our smart environment test-bed. A volunteer participant entered the room and executed two different activities:

- Turn on right lamp(1 min), perform random actions
- Turn on left lamp(1 min), perform random actions

The first activity was repeated 10 times over the course of two hours with random events in between. Then the participant highlighted the activity for monitoring and performed the second scripted version by changing the duration from 1 to 2 minutes. CASAS detected the duration change as 1.66 minutes. The change was made to the correct parameter and in the correct direction, but did not converge on an accurate new value due to the detection of other similar patterns with different durations. These experiments validate that PAM can successfully adapt to resident's preference changes even in real-world data.

We also empirically validated our theoretical analysis to see how fast original preference patterns will be replaced by modified versions. To evaluate this, we designed an experiment in which we generated two sets of synthetic data similar to the first experiment. We then validated the adaptation capability for different decay values and different initial potential value (see Figure 3). Our findings are consistent with our expectation, validating that PAM can successfully adapt to resident changes even in real-world data.

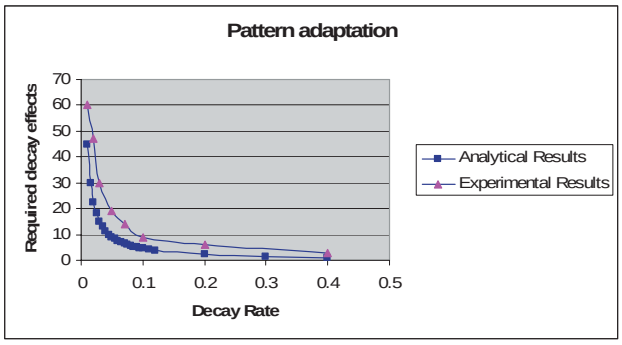

Figure 3 Changes in decay rate.

Conclusions

In this paper we presented a method of adapting to user's dynamic preferences. We introduced these concepts in the context of a smart home design and describe CASAS as a learning-based approach that can adapt according to its residents habits or advice. Our hypothesis was validated by our experiments in which CASAS successfully detected and adapted to the changes in resident preference patterns. In our ongoing work, we plan to add additional features such as voice recognition capability to increase CASAS' ease of use, and to enable users to express their preferences regarding automations more easily.

References

G.D. Abowd and E.D. Mynatt. Designing for the human experience in smart environments. In Smart Environments: Technology, Protocols nd Applications, pages 153-174, 2005.

R. Agrawal and R. Srikant. Mining Sequential Patterns, Proc. 11th Int'l Conf. Data Eng., pp. 3-14, 1995.

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

S. Helal, W. Mann..The Gator Tech Smart House: A programmable pervasive space. IEEE Computer, 38(3):50-60, 2005.

R. Sutton, A. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

J. Rissanen. Stochastic Complexity in Statistical Inquiry. Series in Computer Science-Vol. 15. World Scientific, Singapore, 1989.

V Vainio, A.-M., Vanhala, J. Continuous-time Fuzzy Control and Learning Methods. In: Proceedings of the 7th International Symposium on Communications and Information Technologies, ISCIT 2007. October 16.-19., 2007. Sydney, Australia.

G.M. Youngblood and D.J. Cook. Data mining for hierarchical model creation.IEEE Transactions on Systems, Man, and Cybernetics, Part C, 37(4):1-12, 2007.