Keeping the Intelligent Environment Resident in the Loop

Parisa Rashidi and Diane J. Cook*

*School of Electrical Engineering and Computer Science
Washington State University
{prashidi, cook}@eecs.wsu.edu
Fax: +1 509-335-3818

Keywords: machine learning, user modeling, user interfaces, adaptation, usability study

Abstract

Recent technological advancements have increased the likelihood that smart home technologies will become part of our everyday environments. However, many of these technologies are brittle and do not adapt to the user's wishes or changes in residents' habits and lifestyle. Here we introduce CASAS, an adaptive smart home that utilizes machine learning techniques to discover patterns in user behavior and to automatically mimic these patterns. Our goal is to keep the resident in control of the automation. Users can provide feedback on proposed automation activities, modify the automation policies, and introduce new requests. In addition, CASAS can discover changes in resident's behavior patterns automatically. In this paper we describe the CASAS technologies and evaluate its ability to adapt to the resident's activities and requests.

1 Introduction

Recently there has been extensive research to develop smart environments by integrating machine learning and AI techniques into environments that are equipped with sensors and actuators. A smart environment can be viewed as an intelligent agent, which perceives the state of the environment using sensors and acts upon the environment using controllers in order to optimize a number of goals including maximizing comfort of the residents, minimizing the consumption of resources, and maintaining safety of the environment and its residents [2]. Smart environments have the potential to aid people with cognitive and physical limitations, to reduce resource consumption, and to make our lives more comfortable and productive.

As the need for automating these personal environments grows, so does the number of researchers investigating this topic. Application of related technologies has encompassed environments including conference rooms, offices, kiosks, and furniture [4]. However, a primary hindrance to realizing smart environments' potential is the ease with which smart environment technology can be integrated into the lifestyle of its residents. Our goal is to design a smart environment that adapts to its residents.

With our approach, the resident plays a critical role in guiding the environment's automation policy. We introduce an infrastructure for an adaptive smart home model which includes an adaptive activity mining component to find changes in activity patterns and a feedback-based learning component that adapts to new or modified patterns. To facilitate resident feedback, our design includes a simulator-based interface. Despite increasing progress in smart home technologies, little attention has been paid to design of easy-to-use smart home interfaces that promote greater control of the environment. There are a few related efforts that aim to provide a direct manipulative smart home user interface, such as the ResiSim residential simulator [6] and virtual 3D [1]. Both of these rely on a direct manipulation paradigm, but neither addresses the issue of manipulating event sequences, collecting and responding to user feedback, and representing temporal information with its relation to spatial elements. We hypothesize that our approach will allow the environment to converge more quickly on an automation policy that is considered beneficial by the smart home resident, and we evaluate this hypothesis in the context of a real smart environment.

2 CASAS Components

Our smart home system, CASAS, contains cooperating components that find activity patterns, generate policies to automate these patterns, predict and schedule automated activities, and adapt to explicit user feedback or observed changes in resident behavior. Figure 1 depicts the interaction between these components. These components provide the foundation for automating an environment, which we will further enhance through adaptation to the user's implicit and explicit feedback.

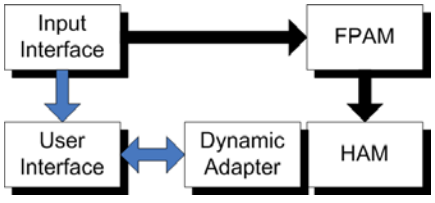

Figure 1: CASAS components.

Input to CASAS consists of sensor data that is collected in the smart environment. Our FPAM (Frequent and Periodic Activity Miner) algorithm mines this data to discover frequent and periodic activity patterns. These activity patterns are modeled by our Hierarchal Activity Model, called HAM, which utilizes the underlying temporal and structural regularities of activities to achieve a satisfactory automation policy.. The Dynamic Adapter notices change in resident behaviors and modifies the automation policies. In addition, users can explicitly request automation changes through the user interface, CASA-U. By keeping the resident in the loop through explicit and implicit feedback, the automation sequences will be useful for the resident and not merely a demonstration of the smart home's automation abilities. We will next describe each component in greater detail.

2.1 Frequent and Periodic Activity Miner: FPAM

In a smart home environment, the first step is to discover the patterns of resident activity that have the potential to be automated. In such a setting, not only it is important to find frequent sequences, but also to find the most regular ones which occur at certain periods, such as weekly. Those frequent and periodic activities can be expressed as a set of time ordered sequences. In order to find these sequences, we defined a new method, called Frequent and Periodic Activity Miner, or for short FPAM, which is able to discover evolving frequent and periodic patterns with inexact periods and of arbitrary length from sensor event data. In addition, FPAM captures triggers and considers temporal information such as duration and start time distributions of events.

To construct FPAM, we exploit a subset of the temporal knowledge discovery field, usually referred to as "discovery of frequent sequences" [9], [10], "sequence mining" [11], or "activity monitoring" [12] where the pioneering work of Agrawal's Aprori algorithm [11] was the starting point. The Apriori algorithm employs a bottom-up approach to incrementally extend frequent sequences until no more acceptable extensions can be extracted from the data The Apriori algorithm identifies sequences that occur at least as often as a minimum number, *C*. While the Apriori algorithm is historically significant, it suffers from a number of inefficiencies which have spawned alternative algorithms and which we address in part with our FPAM algorithm.

We assume that the input data is a sequence of tuples that appear in the form $<d_i, v_i, t_i>$, where d_i denotes a data source (e.g., motion sensor, light sensor, appliance), v_i denotes the state of the source (e.g., off, on), and t_i denotes the time that the event occurred. FPAM makes one pass through the data to calculate frequencies (i.e., number of times the sequence occurs in the dataset) and periodicity (i.e., regularity of occurrences, such as every three hours or once a day). A window of size ω (initialized to 2), is passed over the data and every sequence of length ω, is recorded together with its frequency, or period, if applicable. To provide more efficient access to sequences, they are stored in a hash table.

After the initial frequent and periodic sequences have been identified, FPAM incrementally builds candidates of larger size. Instead of scanning the whole data again, FPAM extends sequences that made the cutoff in the previous iteration by one element that occurs before or after the current sequence. This approach contrasts with Apriori which generates all possible candidate sequences. During iteration *i*, FPAM thus builds candidate sequences of length *i+1* and discards sequences from the previous iteration that are subsequences of the new discoveries Each iteration thus generates *2m* extensions of patterns from the previous iteration, where *m* is the number of discovered patterns in the previous iteration. This is a significant efficiency improvement over the Apriori algorithm, which generates an exponential number of new candidate patterns in each iteration. Continuing the process results in the generation of a multi hash-table structure where each hash table holds sequences of increased size over the previous one. FPAM will continue to increase the window size until no 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 by pointers to the pattern definition. Calculating this compression is tricky for smart home data; as the size of the dataset may not be fixed due to varying activity levels (e.g. an active day will generate a lengthy event data stream).. We compute it using the following formula where f_a represents the frequency of sequence a, $t|$ represents the input data length in hours and C represents a compression threshold. For a sequence to be considered as frequent, the following condition should hold:

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

In addition to finding frequent patterns, FPAM also discovers periodic patterns. To calculate the periodicity of each candidate pattern, the time that has elapsed between occurrences of the pattern is computed. Two different periodicity granules are considered. Coarse-grained periods are expressed in numbers of days (e.g., "every 3 days") and fine-grained periods are expressed in numbers of hours (e.g., "every 4 hours"). None of these periodicity granules require a period to be exact, in fact, fine grained periods have a tolerance of up to one hour and coarse grained periods have a tolerance of up to one day. A lazy clustering algorithm is applied to construct candidate periods. If an activity's period cannot be matched with previous occurrences (with a tolerance of one hour for fine-grained periods and one day for coarse-grained periods), a new candidate period is constructed and is placed in the candidate periods list. If the periodicity for an event sequence is consistent a threshold number of times, the pattern is reported as periodic and is moved to consolidated periods list. Updating candidate and consolidated lists is performed dynamically and a period can be moved from one list to another several times. Such a schema helps to eliminate any transient periods based on current or future evidence, resulting in an adaptive

evolvable mining approach. We will later in more detail how changing patterns will be discovered and incorporated into the model.

Another important notion that has been considered in our smart home data mining procedure is the notion of triggers. Basically, a trigger is an event which causes an activity to start every time the event occurs, such as a specific sensor data (e.g. motion sensor) that can trigger lights to go on or off. From a practical point of view, a trigger or better said a sensor can not be part of a scheduled activity; therefore it is necessary to post-process activities that include a trigger as one of their events. We adopt the following policy, depending on trigger position within an activity:

- At the beginning: remove the trigger from the sequence; instead consider it as a firing condition.
- At the end: has no effect, hence ignore it.

In the middle: split the sequence into two sequences where the trigger becomes the firing condition of the second sequence (see

- Figure 2).
- If a sequence contains more than one trigger, the above steps are repeated recursively.
- If several triggers happen consecutively, we will just consider the last one.

Note that we assume that frequency and period would be the same for split sequences as the original sequence; however the compression value may change as it depends on a sequence's length. Also during the sequence splitting process, there might be a case where the resulting sequence reduces to one of the already existent sequences. In this case, one approach is to repeat the data mining process again to find any existing relation between these two sequences (e.g., they might have different periods).

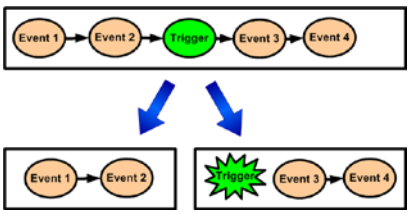

Figure 2: Trigger splitting.

2.2 Hierarchical Activity Model: HAM

Our Hierarchal Activity Model, HAM, dynamically models the smart home sequences that are reported by FPAM. The HAM structure can be considered as a hybrid model of a predictive decision tree, combined with Markov decision processes (MDPs) to capture temporal relationships between individual events. Each frequent activity is a leaf of the decision tree that corresponds to an MDP.

Figure 3 shows a sample HAM model, as the figure shows, HAM decomposes activity information in terms of two temporal granules (day of week and time of day); besides activity metadata, and the activity data that is represented as an MDP. CASAS automatically constructs a HAM model from FPAM data then uses the model to identify activities that need to be automated at a given time. Reinforcement in the form of user advice or inhabitant's lifestyle and habit change is applied to the HAM model using a reinforcement learning method [7]. At any given time, CASAS selects an event to perform that maximizes expected utility [7] based on the feedback the resident has provided for the automated activities.

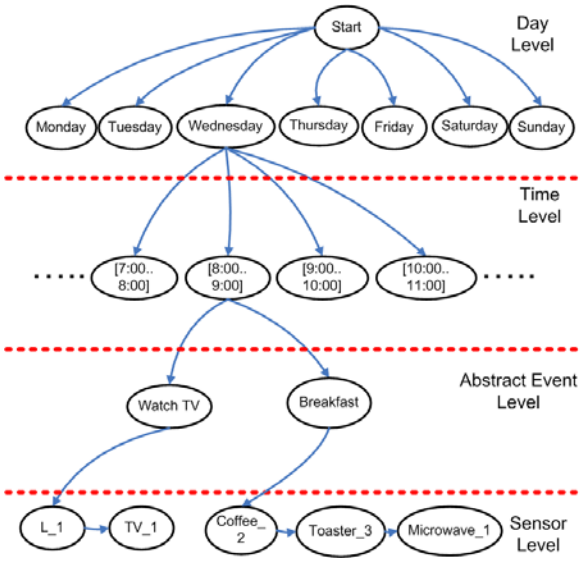

Figure 3: Example HAM model.

HAM captures temporal relationships between events in an activity by explicitly representing sequence ordering in a Markov model. Another important source of temporal information is the start time and duration distributions of events of an activity. There have been earlier approaches to modeling durations of states in a Markov decision process such as the approach by Vaseghi [13] in which state transition probabilities are conditioned on how long the current stated has been occupied. In HAM, we model the start time of the first event of an activity and durations of all events of an activity using a normal distribution.

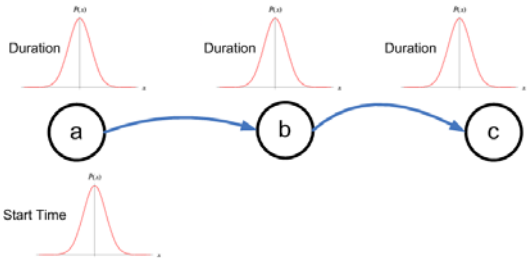

Figure 4: Start time and duration distribution in an MDP.

These distributions are stored in the Markov model nodes (see Figure 4). Both distributions are updated every time FPAM mines newly-generated data. Therefore 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. So, approximating these distributions is achieved not

through using a complicated global function, but rather by summing multiple simple local functions.

2.3 Dynamic Adaptation

Learning a model of resident's activities provides a basis for automating activities in a smart home. However, this is not a long-term solution because residents are likely to change their activity patterns over time depending on factors such as changing job or social relations, seasonal and weather conditions, and mental and emotional condition. As a result, we need to find a way to adapt to the changes that occur over time, in addition to incorporating explicit resident guidance of automation policies. CASAS achieves adaptation based on the resident's explicit feedback, provided through the CASAS user interface as well as implicit feedback via 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 feedback that should be detected by CASAS.

We employ four different adaptation mechanisms to consider the resident's explicit and implicit feedback: direct manipulation, explicit guidance, implicit guidance and smart detection (see Figure 5). CASAS uses all of these mechanisms to result in a flexible, user-centric solution to the dynamic adaptation problem that allows for various degrees of resident involvement.

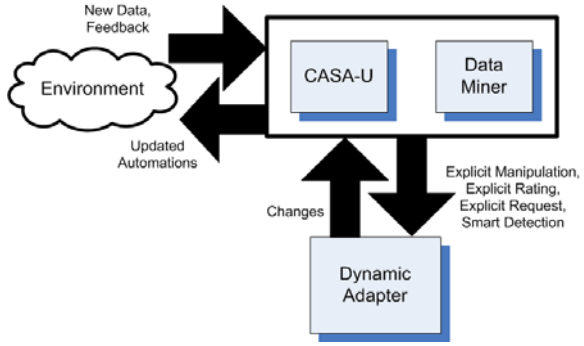

Figure 5 Adaptation Mechanisms.

In the direct manipulation approach, residents can directly manipulate HAM models by changing event start times, durations, or triggers, through CASA-U interface. They can also add activities, delete activities, or modify entire activities by adding, deleting, or reorder the events that comprise the activity.

In the explicit guidance method, users can guide CASAS by rating the automated activities based on their preferences on a scale of 1 to 5 where 5 represents that the resident likes the automated activity.

In the implicit guidance method, users can highlight an activity in CASA-U for observation, and allow CASAS to automatically detect changes and modify the model for that activity. This option removes the burden of explicit user manipulation. Whenever an activity is highlighted for monitoring, our adaptation algorithm collects recent event data and mines the sequences, looking for potentially-changed versions of a specific activity. These changes may consist of new activity start times, durations, triggers, periods, or structure. Structure change is detected by finding new patterns of activity that occur during the times that CASAS expects the old activity to occur. 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 AAM algorithm notifies the user of the changes by offering the list of changes that can be accepted or rejected.

Finally, in the smart detection method, CASAS passively monitors resident activities and mines collected data at periodic intervals (e.g., every three weeks). If a significant event change in events is perceived, CASAS will automatically update the corresponding HAM model. This approach adapts slower than implicit guidance method, in which the change detection starts immediately, such that users do not have to wait for the regular mining schedule.

To incorporate different types of implicit and explicit feedbacks provided by above adaptation mechanisms, we use a modified version of reinforcement learning to update the value functions of activities, and call it *feedback-based learning*. Feedback-based learning algorithm updates an activity's value function using two different learning rates, one learning rate applied to activities that their change has been detected automatically by CASAS (implicit guidance and smart detection); and a second, higher learning rate for activities that received explicit resident feedback or were explicitly introduced by the user (explicit guidance and direct manipulation).

Note that while feedback-based learning is based on reinforcement learning, there are substantial differences. Instead of modeling the entire world as an MDP, we model each individual activity as a separate MDP. In addition, unlike reinforcement learning which relies on a reward/punishment value in each state; feedback-based learning uses explicit feedback to force an immediate change in the model, and implicit feedbacks towards gradual stabilization.

2.4 CASA-U

The main obstacle to the design of smart home systems is the ease with which smart environment technology can be integrated into the lifestyle of its residents. To date, little effort has been devoted to this challenge. In current work, we try to address some of the key issues and challenges by applying a user-centric approach [5] to design of a smart home user interface. The user interface is designed as a simulation environment in which all previous and current activities can be visualized and residents are able to navigate through the map of the home, identify and modify automated events or their timings, and provide feedback to the smart home based on automation policies.

Several issues make design of smart home interfaces challenging. One such challenge is the choice of representation for smart home activities. Our objective is to present the smart home and its automation policies to the user in a clear manner, using a floor plan of the home as a primary means of communicating this information. We also need to represent the spatial relationship between elements in the home, the status of these elements, and the temporal nature of associated actions.

We consider our smart environment interface, CASA-U, as a discrete event simulator where each object is a self-descriptive, iconic representation of an item in the environment. The sensor layout and floor plan of the visualized simulation is based on the sensor layout and floor plan of a physical smart environment test-bed at Washington State University. The portion of the environment that we use for the experiments in this paper is modeled as shown in **Error! Reference source not found.** and is equipped with motion sensors, rug sensors, light sensors, and controllers for the lamps. To show the effect of motion sensors detecting someone walking around the room, we used footprints to imply the motion effect.

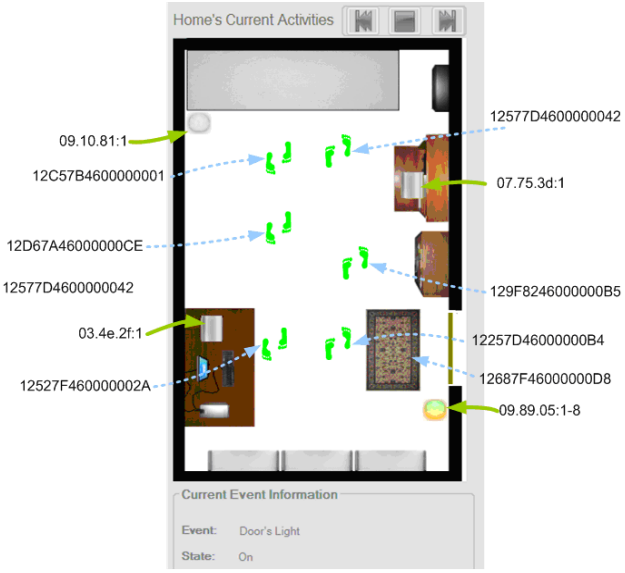

Figure 6: Sensor layout and floor-plan map.

CASA-U allows the resident to control events that are distributed across time as well as the resident's living space. To achieve this, CASU-U creates a temporal framework and spatial framework to allow the resident to perceive, comprehend, and ultimately modify events occurring in the physical world around the resident. In our schema, the floor map provides a spatial framework and the temporal constraints are displayed as an animation of event sequences where the direct mapping of the order of events in the physical world maps to the order of the displayed elements. In order to provide a clearer view of temporal relations, we label the objects with numbers that indicate their temporal ordering. Two spatial views are available to the user: one visualizes a live stream of sensed events (activity view) and the other animates past automated CASAS activities (automation history view) as in **Error! Reference source not found.**. In both views, the user has the option to go backward or forward in the stream using rewind/forward buttons to find a specific event.

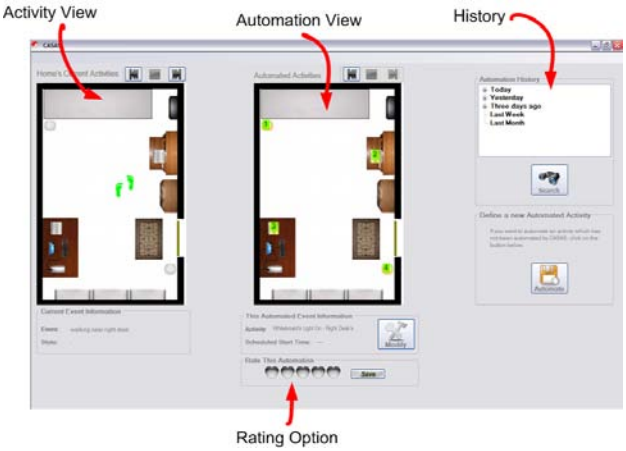

Figure 7: Main view of CASA-U.

Another challenge we faced in the design of the CASA-U user interface was how to provide users with the ability to change scheduled automated activities or request new automations. Each activity's model includes a definition of constituent events, the relative and absolute temporal relationships between these events, the duration of each event, triggering conditions and periods. Given this complexity, it is essential to provide the user with adequate guidance. In our design, 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.

Activity modification in CASA-U uses a direction manipulation approach. Users can click on an item in the map and change its attributes using either a context menu or drag and drop. For example, to define a new activity that includes the sequence "turn on the desk lamp then turn on the whiteboard light", the user may first right click on the desk lamp, select "turn on" from the context menu and repeat the step for the white board light, or alternatively drag them into the activity panel.

CASA-U also provides an option to search for activities of interest based on occurrence time or included events. A

separate map is used to visualize the results and the user view search results using a forward/rewind button.

CASA-U also provides an option for users to highlight an activity to be monitored. Once changes have been detected in the highlighted activity, a window is provided to the user that summarizes the nature of the detected change. The user then has the option to apply some of all of the detected changes to the model for future automation.

3 Experimental Results

Our goal is to design smart home software that adapts to the needs of the resident. Here we evaluate the adaptability and usability of the CASAS software using synthetic and real data collected in our smart environment.

3.1 Evaluation of FPAM

In order to evaluate FPAM's ability to find frequent and periodic patterns, we tested it on both synthetic and real data obtained from a smart home system. To test the algorithm on synthetic data, we implemented a synthetic data generator that simulates events corresponding to a set of specified activities. Timings for the activities can be varied and a specified percentage of random events are interjected to give the data realism. In addition to synthetic data, we evaluated FPAM on real-world collected data from the mentioned test-bed in WSU's School of Electrical Engineering and Computer Science.

At first, we generated one month of synthetic data that contained events for six activities. Five of the activities were event sequences of length two and the remaining activity contained an event sequence of length three. One activity included an "open the door" trigger at the end and another included an "open the door" trigger in the middle of the sequence. The periods for these activities ranged from two to five hours. The devices and sensors that were simulated in this scenario were the same as those installed in the physical test-bed. Details of the six simulated activities are listed here by start time, period, and events with the duration of 5 minutes for each single event.

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: Generated activities.

Our expectation was that FPAM would correct identify all six activities with their corresponding periods, triggers, start times, and durations. In fact, FPAM was able to find all of the activities with their correct periods. In addition, FPAM identified triggers where they existed and accurately divided the sequence into two subsequences, in case of triggers in the middle. Whenever it is faced with two events that are scheduled to occur at the same time, the synthetic data generator randomly picks just one of the events to simulate. Because of this design feature, the frequency of some of the detected activities was not as high as expected and thus the compression rates were lower than anticipated.

Next, we evaluated FPAM's ability to find variable-length sequences. For this experiment, we generated one month of synthetic data that included the following sequence of length ten and a period of two hours:

- DoorLight/1, LeftDeskLamp/1, LeftDeskLamp/0, LeftDeskLamp/1, LeftDeskLamp/0, LeftDeskLamp/1, LeftDeskLamp/0, LeftDeskLamp/1, LeftDeskLamp/0, LeftDeskLamp/1

FPAM was again able to discover the embedded activity and calculate its period as two hours. Interestingly, it also discovered another frequent pattern with a period of 28 hours. Because in this synthetic domain there are a limited number of devices and some randomness is interjected, there is always a chance that an undeclared frequent pattern will form, as was the case in this experiment. We traced this sequence as a frequent pattern of length 11, which provides additional evidence that FPAM can discover variable-length sequences.

In addition to these synthetic data experiments, we also tested FPAM on real smart environment data. Because we needed to validate that the discovered patterns aligned with what we knew existed in the data, we recruited a participant to execute a simple script in the AI Lab. The participant moved through the lab for about an hour, repeating the script ten times. In order to inject some randomness into the data, the participant was asked to perform random activities for about one minute in step three of the script. The script is defined as follows:

1. Turn on right desk light, wait 1 minute.
2. Turn off right desk light.
3. Perform random activities for 1 minute.

Because our test-bed environment is fairly small, the participant can inadvertently create patterns in the random actions themselves. In addition, the motion sensors picked up slight motions (such as hand movements) which resulted in a randomly-triggered pattern that occurred between steps 1 and 2 in the script, which FPAM then split into two subsequences. The light sensor also occasionally fired in the lab after the desk light was turned on. Despite these issues that occur in a real-world situation, FPAM was able to accurately discover the following patterns:

- Events: Right desk lamp on, Triggers: Walking in middle of room, Compression: 12
- Events: Right desk lamp off, Triggers: none, Compression: 9
- Events: Left desk lamp on, Triggers: none, Compression: 2
- Events: Right desk lamp on, Right desk lamp off, Triggers: none, Compression: 10
- Events: Whiteboard light on, Triggers: none, Compression: 2

As can be seen from these results, our expected patterns were discovered. The first and second patterns are the result of splitting the sequence "Right desk lamp on, Random event, Right desk lamp off" into two subsequences. The "walking" trigger is correct because after turning the light off, the participant performs a random action and heads back to the right desk to turn on the light, which involves walking across the room to reach the desk. The difference in compression values between the first and second sequences is due to multiple triggers from the light sensor for a single light on or light off action.The third sequence is the result of a random activity; the compression value is relatively small compared to the main script activities. The fourth sequence reflects the embedded activity, and the last sequence is a frequent activity associated with random events, again with a smaller compression value. While these results support our claim that FPAM can detect frequent and periodic activities, we see that precautions need to be taken to reduce interference due to sensor data.

3.2 Evaluation of the Dynamic Adapter

For our first experiment we created one month of synthetic data with six embedded scenarios, the same as in previous experiment with FPAM. After FPAM and HAM constructed models for these events, we highlighted Scenario 3 to be monitored for changes. We then changed the scenario description in the data generator so that all event durations were 7 minutes (instead of 5 minutes). CASAS detected the changes accordingly by finding 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 timings.

We next tested our adaptor on real world data using the smart environment of WSU AI Lab. A volunteer participant entered the room and executed two different scripts:

- Turn on the right desk lamp, wait 2 minutes, turn of lamp, perform random actions for 1 minute.
- Turn on right desk lamp, wait 1 minute, turn off lamp, perform random actions for 1 minute.

The first script was repeated for 2 hours with random events in between execution of the scenarios. CASAS generated the HAM model then the participant highlighted the scenario for monitoring and started to perform the second scenario. CASAS correctly detected the duration change. In addition, FPAM detected a new trigger, "walking near the left desk", which was inadvertently a common event that the participant performed before turning on the desk lamp.

3.3 CASA-U Usability Study

In order to test the usability of CASA-U interface, we performed a usability study. In the study, participants first completed a background questionnaire and then they explored the CASA-U interface. None of the participants had seen CASA-U before. We allotted 50 minutes for each study session. Participants were asked to complete a series of five tasks using the CASA-U software. These tasks included:

1. Rate a current automated activity.
2. Select an activity for monitoring.
3. View all automations in the automation history and navigate back to the beginning.
4. Search for an activity that includes the event "right desk lamp".
5. Add a new automated activity with the characteristics: it has not occurred before, it starts every hour and includes the events "Whiteboard light on" with duration of 4 minutes and "right desk lamp on" with duration of 3 minutes.

Upon completion of the tasks, participants filled out an exit questionnaire that solicited their impressions of the CASA-U software by asking them about their overall reaction, clarity of terminology, screen layout, ease of learning and the system's capabilities in a total of 40 questions based on a modified version of a standard Questionnaire for User Interaction Satisfaction (QUIS) test [8]. The QUIS was designed to assess users' subjective satisfaction with specific aspects of a human-computer interface. After analyzing the QUIS questionnaire, the software received QUIS ratings for overall user reaction, screen and layout, terminology and system information, learning, and system capabilities as shown in Figure 8 for average scores.

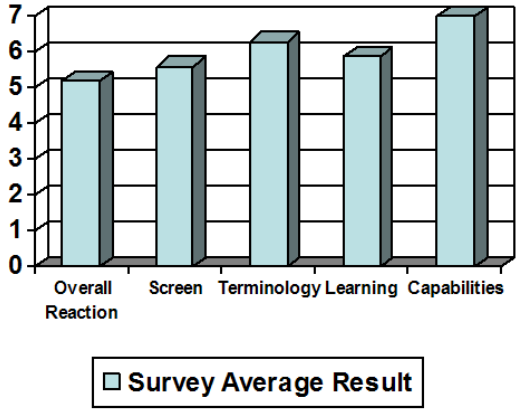

Figure 8 Average QUIS survey results.

To verify how well our usability and user experience criteria were satisfied, we analyzed tasks that were carried out by participants in terms of the elapsed time that each participant used to complete each step and in terms of user ratings of the system.

Table 2 shows the initial requirements and summarizes the empirical findings based on the usability test sessions (also see Figure 9). The average elapsed time was 156 seconds for all 5 tasks. The expected times were estimated based on the number of dialogs the users needed to navigate.

Task	1	2	3	4	5
Expected Time	20	20	45	60	90
Avg. Result	38.8	22	40.2	20.4	34.6

Table 2: Expected times and empirical results.

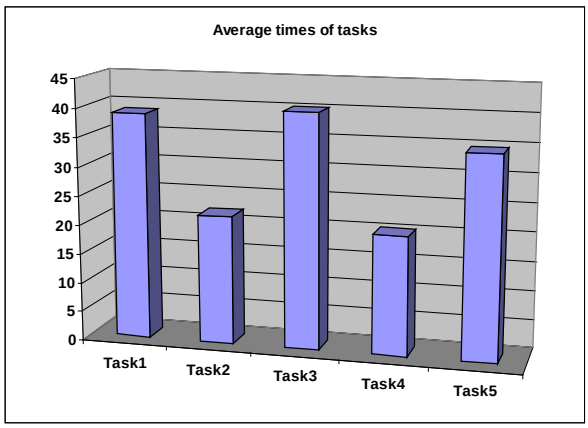

Figure 9 Completion times in empirical results.

Despite the fact that we expected task 5 to take the longest, most participants finished it easily with the use of dialogs that guided them through the process. Also, a few participants chose to "detect new automation automatically", indicating that considering "intelligent" options does make the process simpler.

Despite the fact that most users could achieve assigned tasks within the expected time, still they were struggling with learning the system and forming logical representations of activities. Some participants mentioned that the terminology was not clear. For example, some did now know what "monitoring an activity" means. Therefore, a revision in terminology is also necessary. Some participants also mentioned that they could not understand what should be expected of a smart home user interface and how they could ideally interact with smart home software. For example, two participants did not clearly understand what the numbered events mean and they could not understand what it means to "automate an activity", what aspects will be automated, and how they will be automated. In summary, the usability study revealed that the interface was relatively clear and easy to navigate, but additional training is needed for residents to make effective use of smart home technologies.

In addition to evaluating CASA-U, participants were asked whether they would be willing to use smart home technologies at home or not. Two of the participants mentioned that due to privacy issues they might not be willing to use such a technology while others mentioned that utilizing such a technology could be useful in everyday life and were willing to use such a technology at their own home. Some studies of smart home technology acceptance have been performed [3], but the results of this study highlight the need to perform usability tests in actual smart environments to determine the usefulness and acceptability of smart home technologies.

4 Conclusions

In this paper, we presented CASAS, an integrated set of components that aim toward applying machine learning and data mining techniques to a smart home environment. According to our experiments, we were able to detect, automation, and adapt to the changes in resident patterns, as hypothesized. We also performed a usability study of our user interface, CASA-U, to determine the usability of this interface and identify its shortcomings. In our ongoing work, we plan to perform more user studies in real world setting to better understand the strengths and weaknesses of the system. Ultimately, we anticipate adding additional features such as a voice recognition capability to the system to increase availability and ease of use.

References

[1] L. Borodulkin, H. Ruser, and H.-R. Tränkler. 3D virtual smart home user interface. *Proceedings of the IEEE International Symposium on Virtual and Intelligent Measurement Systems*, Girdwood, AK, 2002.

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

[3] G. Demiris, M. Rantz, M. Aud, K. Marek, H. Tyrer, M. Skubic, and A. Hassam. Older adults' attitudes towards and perceptions of "smart home" technologies. *Medical Informatics and the Internet in Medicine*, 29(2):87-94, 2004.

[4] A. Fox, B. Johanson, P. Hanrahan, and T. Winograd. Integrating information appliances into an interactive space. *IEEE Computer Graphics and Applications*, 20(3):54-65, 2000.

[5] D. Norman and S. Draper. User-Centered System Design: New Perspectives on Human-Computer Interaction. CRC Press, 1986.

[6] P. Rashidi, G.M. Youngblood, D. Cook, and S. Das. Resident guidance of smart environments. *Proceedings of the International Conference on Human-Computer Interaction*, Beijing, China, 2007.

[7] R. Sutton and A. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[8] J.P. Chin, V.A. Diehl, and K.L. Norman. Development of an instrument measuring user satisfaction of the human-computer interface. *Proceedings of SIGCHI*, pages 213-218, 1988.

[9] S. Laxman and P.S. Sastry. A survey of temporal data mining. *Sādhanā*, 13(2):173-198, 2006.

[10] J.F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods, *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750-767, 2002.

[11] R. Agrawal and R. Srikant. Mining sequential patterns. *Proceedings of the International Conference on Data Engineering*, pages 3-14, 1995.

[12] T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 53-62, 1999.

[13] S.V. Vaseghi. State duration modeling in hidden Markov models. *Signal Processing*, 41(1):31-41, 1995.