SMART HOME ADAPTATION BASED ON EXPLICIT AND IMPLICIT USER FEEDBACK

BY

Parisa Rashidi

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Washington State University

Electrical Engineering and Computer
Science Department

December 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of PARISA RASHIDI find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

ACKOWLEDGEMENTS

This research project would not have been possible without the support of many people. I wish to express my gratitude to my supervisor, Prof. Dr. Diane J. Cook who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude is also due to the members of the supervisory committee, Dr. Behrooz Shirazi and Dr. Christopher Hundhausen as without their knowledge and assistance this study would not have been successful. I also wish to express my love and gratitude to my beloved family, for their understanding & endless love, through the duration of my studies.

SMART HOME ADAPTAION

BASED ON EXPLICIT AND

IMPLICIT USER FEEDBACK

Abstract

by Parisa Rashidi, M.S.
Washington State University
December 2007

Chair: Diane J. Cook

In current work we introduce CASAS, an adaptive smart home system that utilizes machine learning techniques in order to dynamically adapt to user advice or changes in daily routine activities. The main components of CASAS include a frequent and periodic activity miner (FPAM), a hierarchal activity model (HAM), a dynamic adapter and CASAS's user interface and visualizer, CASA-U. The FPAM algorithm discovers arbitrary length periodic and frequent patterns from the resident's daily activities efficiently by utilizing the minimum description length principle. HAM, as a hybrid model of a decision tree combined with a Markov decision process, provides a hierarchal abstraction of patterns while utilizing temporal information such as temporal relations, temporal granules, start time and duration distribution. HAM is used to identify potential automations. The dynamic adapter component allows HAM to dynamically adapt to user's explicit feedback (advice) or implicit feedback (changes in daily routine activities) based on four techniques of explicit manipulation, explicit rating, explicit request and smart detection. It exploits guidance-based learning and observation-based learning along with the Activity Adaptation Miner (AAM) to adapt to these types of feedback. Finally, to allow users have a greater control over their personal environment and to provide a framework for explicit manipulation and rating of suggested automation policies, a user interface is provided that enables residents to navigate through a map of the home, view a history of events, modify the events and provide guidance to the smart home's automation policies. Integrating all these components together, the architecture of CASAS is provided that shows how resident interactions in a smart home can be automated and continually adapted to explicit or implicit changes in the resident's patterns. We also show the results of our successful experiments with CASAS on both synthetic and real world data, besides a usability test of CASA-U.

TABLE OF CONTENTS

List of TABLES

Number Page

Dedication

This thesis is dedicated to my mother who provided both emotional and financial support through the duration of my studies.

CHAPTER ONE

INTRODUCTION

"He is the happiest, be he king or peasant, who finds peace in his home"

Johann Wolfgang von Goethe

Since the beginning, people have lived in places that provide shelter, basic comfort, and support, but as society and technology advance there is a growing interest in improving the intelligence of the environments in which we live and work. Recently there has been extensive research toward developing smart environments by integrating various machine learning and artificial intelligence techniques into home environments that are equipped with sensors and actuators. Basically, a smart environment is an environment which acquires and applies knowledge about the physical setting and its residents, in order to improve their experience in the setting. A smart environment can be treated as an intelligent agent, which perceives the state of the environment using sensors and acts upon the environment using device controllers in a way that can optimize a number of different goals including maximizing comfort of the residents, minimizing the consumption of resources, and maintaining safety of the environment and its residents. Smart environments have the potential to aid people with cognitive and

physical limitations, to provide resource conservation, 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. Some researchers are targeting individual devices that provide a useful function, such as programmable appliances [89]. While these devices are novel and useful for limited tasks, they typically do not consider the bigger picture of interacting with the rest of the environment. Others have designed interactive conference rooms, offices, kiosks, and furniture with seamless integration between heterogeneous devices and multiple user applications in order to facilitate collaborative work environments. Ideas [90] for recognizing environment resident activities [1], for planning environment reminders [2], and for reacting to hazardous situations [3] have also been discussed.

However, a primary hindrance to realizing smart homes' potential is the ease with which smart environment technology can be integrated into the lifestyle of its residents. Our vision 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 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. We will validate our hypothesis by providing an

implementation of our approach and evaluating its ease of use as well as its ability to quickly converge on an accurate policy that incorporates the resident's explicit and implicit feedback.

In following sections, we will provide an overview of existing smart home technology, our initiatives and also related work. In the next chapter, we will describe the design and implementation of CASAS as a smart home environment which utilizes data mining and machine learning techniques to discover and automate frequent and periodic patterns of activity, besides adapting to dynamic changes or user advices. CASAS also includes a user interface and simulation environment, CASA-U, which can be used to visualize daily resident activities and capturing user feedback and advices. In the next chapter, we will discuss the experimental results to assess the effectiveness of our approach. In the last chapter we conclude with a summary of the research and a discussion of alternative and future research directions.

Overview and Related Works

With remarkable recent progress in computing power, networking equipments, sensors and various machine learning and artificial intelligence methods, we are steadily moving towards Mark Weiser's Ubiquitous Computing [31], beyond PCs and mobile phones, into a world entangled with sensors and actuators as he envisioned: ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people, then we moved to the personal computing era, in which person and machine stare uneasily at each other across the desktop. Next came ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives.

A research area that can highly benefit from advances in ubiquitous and pervasive computing is smart environment development. Since 1940, smart environments and especially smart homes have been a topic of interest for many researchers with the major goal of automating control of home appliances in order to achieve comfort, security, and energy efficiency, and recently for health monitoring of older adults or people with cognitive impairments [4]. An intelligent environment or smart space can be thought of as having many highly interactive and embedded devices and the ability to control these devices automatically in order to meet the demands of the environment or space. It is a technological concept

referring to a physical world that is interwoven with sensors, actuators, displays and computational elements embedded in the everyday objects found in the environment. This world is envisioned as the byproduct of pervasive computing and the availability of cheap computing power, making human interaction with the system a pleasant experience.

Cook and Sajal [5] define a smart home as a small smart environment where all kinds of smart devices are continuously working to make residents' lives more comfortable by acquiring and applying knowledge about residents and their surroundings to improve the resident's experience, life style and safety in those surroundings. Smart homes aim to satisfy the experience of individuals, by replacing physical labor and repetitive tasks with automated agents.

Recently there has been an increasing interest in smart environments and especially smart home technology, and different directions have been taken to approach the idea of integrating artificial intelligence techniques with pervasive computing. Some research labs and manufacturers are targeting individual devices that provide a useful function, such as programmable appliances [6, 7], hot pads that sense the heat of a pan [8], or an interactive tablecloth [9] that weaves a power circuit into a washable linen tablecloth, so that devices can be charged when they are placed anywhere on the tablecloth. While these devices are novel and useful for limited tasks, they typically do not consider the bigger picture of

interacting with the rest of the environment. As Rode points out [10], they also rarely consider difficulties encountered in cultures and markets other than the one for which they are designed. Rode observes that these devices would be much more useful if they could adapt themselves to new environments and tasks. Some researchers have considered automation of an entire space and have designed interactive conference rooms, offices, kiosks, and furniture with seamless integration between heterogeneous devices and multiple user applications in order to facilitate collaborative work environments [11-14].

A key objective in a smart home environment is to provide the resident with the ability to give feedback about the smart home automation policy, to modify predicated policies, and to add new policies. This area of research has not been explored so far, and this is the focus of our work. This is an area where machine learning techniques and artificial intelligence methods come into play an important role. AI and machine learning techniques can be used to contribute in many ways to the development of smart homes. For example, artificial intelligence methods for smart homes may provide both the flexibility to adapt to changing circumstances and the reasoning capabilities required to interpret events within the home and to make the right choices at the right times.

Augusto and Nugent [48] argue for the promise of smart homes as an AI domain, both because of the potential payoff for AI methods and because the constrained

task environment of smart homes facilitates application of AI solutions. Various computational intelligence techniques have been proposed and tried to support the needs of the smart homes, such as neural networks [49,50], Fuzzy logic [51], Hidden Markov Models [52,53], and Bayes Classifiers [54].

All of these techniques have their own merits and limitations when applied to a smart home premise. For example, before using a fuzzy logic smart home controller, appropriate rules and membership functions have to be defined based on prior expert knowledge about the solution of a problem. This might be a quite challenging task, especially when the prior expert knowledge is either limited or not available [55]. Although a Bayes Classifier model might seem simple and fast, its performance depends on the independence of the input features and the selection of the initial distribution for the model. Hence, the Bayes classifier might not be suitable for devices in smart homes that would require an automated crisp and perfect solution [56]. Hidden Markov Models (HMM) can result in very complex networks, and although a HMM can be quite useful for situations such as behavioral monitoring [57], its application would be limited in a smart home due to its inefficiency while dealing with a large number of sensors [58].

Ideas for developing smart environments and supporting technologies abound. Due to the difficulty of creating an automated physical environment, many of

these ideas are discussed in theory or are tested just on synthetic data. In those cases where physical environments have been designed [29, 16, 18, 30], the culmination of the project is an environment with sensing and automation. In none of these projects is the focus placed on creating a pleasing environment where the automation is beneficial and not annoying, and where the resident can guide the environment to behave in a customized manner.

As examples of smart home research, we can name Abowd and Mynatt's work [15] which focuses on ease of interaction with a smart space, and work such as the Gator Tech Smart House [16] that focuses on application of smart environments to elder care. Research on smart environments has become so popular that NIST has identified seamless integration of mobile components into smart spaces as a target area for identifying standardizations and performance measurements [17], although no performance metrics have yet been produced by the group. Mozer's Adaptive Home [18] uses neural network and reinforcement learning to control lighting, HVAC, and water temperature to reduce operating cost. In contrast, the approach taken by the iDorm project [19] is to use a fuzzy expert system to learn rules that replicate resident interactions with devices, but will not find an alternative control strategy that improves upon manual control for considerations such as energy expenditure. Many ideas have been presented to build and enhance smart environments and to benefit from smart environment assistance. Ideas for resident activity and state recognition [20-25], for planning of

environment reminders [26,27], and for reacting to hazardous situations [28] have been discussed.

One closely related project to ours is the MavHome project [32] which was developed at the University of Texas at Arlington. MavHome uses a method of automatically constructing HHMMs based on the output of a sequential data-mining algorithm and sequential prediction algorithm. The results obtained from these steps are applied to a real home setting in order to automate activities. However, there are a number of shortcomings in the MavHome project. For example MavHome is able to detect frequent patterns only in a fixed length window which causes it not to be able to discover activities of arbitrary length. In addition, it does not use context information such as start time, duration or triggers and its hierarchal model partitions activities based on the location which can not provide fine granular decomposition. Finally, and most importantly, its adaptation capabilities are fairly limited. In our work, offer a solution to these earlier shortcomings.

In our work, we will use data mining and machine learning techniques to discover frequent and periodic patterns of activity and to evolve and adapt discovered models over time. The patterns will be modeled by utilizing a hybrid model of Markov decision processes combined with decision trees. For discovering frequent and periodic patterns, we will use a new sequential data mining method

(FPAM) that has the ability to detect frequent and periodic patterns of arbitrary length efficiently and adapt to the changes over time while also considering temporal information such as start time and duration. The hierarchal model that represents discovered patterns is called a hierarchal activity model (HAM) which is used to predict and schedule automated activities and is a hybrid model of decision trees and Markov decision processes. HAM utilizes contextual information such as temporal relations, start time distributions, duration distributions and startup triggers in order to predict and schedule automated activities accordingly.

One important issue that still has not been explored by smart home researchers is how to adapt to the changing environment, which is a crucial issue in such systems as usually humans change their habits and activities over time. To achieve this goal, we use a mining algorithm called an Activity Adaptation Miner (AAM), along with guidance-based learning and observation-based learning as a form of reinforcement learning to discover any changes in previous automated activities and to adapt to user advices.

We also design and implement a visualizer and simulator user interface to address user experience and usability issues. The motivation for the design and implementation of such an interface is that 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 works that aim to provide a direct manipulative smart home user interface, such as the ResiSim residential simulator [59] and virtual 3D [60]. Both of them rely on 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 have tried to address these issues with respect to user experience and usability aspects.

Initiatives

Considering the rapid increase in percentage of aging population, a rising problem in the health care industry is the increased need for eldercare, especially for supporting a healthy, safe and most importantly independent life for senior citizens as they usually have a desire to remain independent. Therefore it seems that the segment of the population that may be mostly greatly affected by smart environment technologies is older adults and those with disabilities. Lanspery, et al. [33] note that older adults and people with disabilities want to remain in their homes even when their conditions worsen and the home cannot sustain their safety. As the population ages, this group is increasing [34] and the effects are expensive as well as unsatisfying. The potential benefits to automating daily activities give much hope for elder adults and people with disabilities for living independent lives at home. AARP reports [35, 36] strongly encourage increased funding for home modifications that can keep older adults independent in their own homes, and NSF has identified Technologies for Successful Aging as a 2007 priority topic [37].

We hypothesize that with the aid of smart environments many people with mental and physical disabilities can lead independent lives in their own homes with the aid of at-home automated assistance. The main hindrance to realize this

potential is the ease with which the smart environment technology can be integrated into the lifestyle of the residents. As an indication of interest in this area, a number of researchers are developing environment assistive technologies to support aging in place [38-43]. Allowing the performance of smart environments to be influenced by the resident is the theme of this proposed work.

Besides the potential benefit for senior citizens, industry is also very interested in the pervasive computing future that society is moving towards, and they recognize that the home is a fertile ground for offering products and services to improve the lives of people. To this end there are a number of industry initiatives that explore technologies and concepts in this arena. Some of ongoing industry projects include British Telecom's Telecare project [44], Intel Corporation's Proactive Health Lab [45], Siemens AG home appliance integration project [46] and Philips HomeLab Project [47]. In addition to all the above initiatives, another emerging direction of research is to apply machine learning algorithms and artificial intelligence techniques in a real world context to see how well these techniques can be applied to real life situations and assess their potential to enhance quality of life. Smart environments provide an impetus for developing new algorithms and techniques, such as sequential data mining and activity prediction, which in turn can be applied to challenging problems in other fields.

CHAPTER TWO

CASAS ARCHITECTURE

"If you hold a cat by the tail you learn things you cannot learn any other way."

Mark Twain

CASAS, as an integrated set of components, is composed of various parts that work together to accomplish an interwoven set of tasks, including finding frequent and periodic activity patterns, representing these patterns as automated activities, predicting and scheduling automated activities and finally adapting to explicit user feedback (advice) or observed changes in resident behavior. Figure 1 shows an overall picture of system, depicting the interaction between contributing components. To accommodate the above tasks, we exploit data mining and machine learning techniques.

We use a new sequential data mining method, called the Frequent and Periodic Activity Miner (FPAM), to detect frequent and periodic patterns and evaluate them based on the minimum description length principle. FPAM has the ability to detect frequent and periodic patterns of arbitrary length, which offers an

advantage over previously applied sequential data mining algorithms in smart home environments which identified event sequences only in windows of fixed length [32].

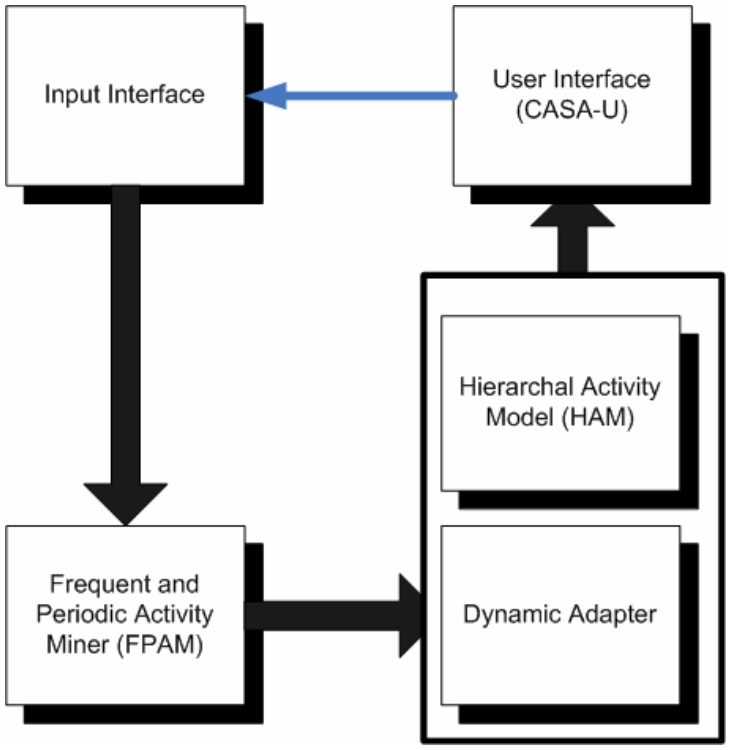

Figure 1 CASAS's overall architecture.

We also introduce a hierarchal activity model (HAM) to predict activities, which is a hybrid model of decision trees combined with Markov decision processes, capable of predicting and scheduling automated activities by utilizing temporal information such as temporal relations, temporal granules, start time distribution, duration distributions and startup triggers.

While many contributing smart home technologies are in place, one of the most important issues that has not been explored by smart home researchers, is the smart home's ability to adapt to the changing environment. Dynamic adaptation in smart homes is a crucial issue as automated activities in these environments are decided based on resident's patterns of activity which usually change over time or depending on various parameters such as weather condition, seasonal trends, changes in jobs, etc. To achieve this goal, we employ a combination of data mining and machine learning methods to discover any changes in previous automated activities, and to adapt to the changing environment based on user guidance and advice. We introduce four different approaches in order to accommodate dynamic adaptation, including: explicit manipulation, explicit guidance, explicit request and smart detection. Depending on the approach taken, we will be using a mining algorithm called Activity Adaptation Miner (AAM) to discover changes, learning methods called guidance-based learning and observation-based learning, or a combination of all of these techniques to adapt to user advice or changes in regular resident routines.

Also we would be exploring design and development of a visualizer and simulator user interface to address user experience and usability aspects. As already mentioned, there have been few works that address the issue of manipulating event sequences, collecting and responding to user feedback, and representing

temporal information with its relation to spatial elements. We have tried to address these issues with respect to user experience and usability aspects.

Frequent and Periodic Activity Mining: FPAM

The first step of automating activities in a smart home is discovering the frequent or periodic patterns of the resident's activity; these activities usually can be expressed as a set of time ordered sequences. Finding these frequent or periodic patterns can remove the burden of doing repetitive tasks from the inhabitant by automating them accordingly. Discovering frequent and periodic patterns among a set of time ordered sequence can be achieved by using a sequence mining algorithm adapted to a smart home scenario. However, finding patterns in sequences is a challenging problem and requires special attention when adapting the technique to a smart home scenario. There are a number of methods for finding frequent and periodic patterns in data, initiated from different fields such as bioinformatics, web mining, database mining, etc [91-93]. These methods, generally known as sequence mining, are concerned with finding statistically relevant patterns between time-ordered structures that form a sequence. By processing these time-ordered sequences sequence miners try to discover the interesting episodes that exist within the sequence as an unordered collection.

In our work, we define a new method, called Frequent and Periodic Activity Miner or for short FPAM. FPAM is based on a modified version of the Apriori algorithm [76], which is a classic algorithm for finding patterns in sequential data. FPAM tries to provide a more efficient version of Apriori algorithm that is also able to find periodic patterns, besides considering duration and start time distributions of events into account.

Literature Overview

To find frequent and periodic patterns of activity, we will be exploiting a subset of the temporal knowledge discovery field, usually referred to as "discovery of frequent sequences" [75, 80], it also sometimes is called "sequence mining" [76] or "activity monitoring" [77]. The pioneering work of Agrawal's Aprori algorithm [76] was the starting point in this area which was improved afterwards in the GPS algorithm by defining a sliding window [94]. Apriori uses a bottom up approach, where frequent subsets are extended one item at a time in a step known as candidate generation, and then groups of candidates are tested against the data; the algorithm terminates when no further successful extensions are found.

There have been a number of extensions and variations to the Apriori algorithm, for example, a group of researchers at the University of Helsinki have been focusing on discovery of episodes that occur frequently within sequences and they distinguish between serial and parallel episodes [78]. Bettini et al. propose a

complete framework for the discovery of frequent time sequences, placing particular emphasis on the support of temporal constraints on multiple time granularities where the mining process is modeled as a pattern matching process performed by a timed finite automaton [79].

An important issue in sequence mining algorithms is how to determine the window size that slides over data to discover patterns. Several potential solutions to this problem have been proposed such as a maximum allowed inter-node time constraint which dynamically alters window widths based on the lengths of episodes being discovered [82]. Similarly, episode inter-node and expiration time constraints may be incorporated in the non-overlapped and non-interleaved occurrence based counts [83]. However, in our scenario as we are dealing with irregular inter-node time constraints, the above solutions can not be applied. Instead, we apply a variable size window whose length is increased incrementally and symmetrically.

FPAM

In our work, we introduce a variant of the Apriori algorithm by addressing a number of issues that are known to exist in the Apriori algorithm. The Apriori algorithm attempts to find repeating subsets, as frequent as at least a minimum number *C* (the cutoff, or confidence threshold) of the total item sets. It uses a bottom up approach, where frequent subsets are extended one item at a time in a

step known as candidate generation, and then groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. Apriori uses a breadth-first search and a hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length $k - 1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the data to determine frequent item sets among the candidates [76].

Apriori, while historically significant, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets by attempting to load up the candidate set with as many candidate sequences as possible before each scan. To address this problem, we do not generate all the candidate lists, but rather slide a variable size window over the event data to discover variable-length sequences where the window size increases incrementally. This is an improvement over previous versions of smart home data mining techniques such as the ED algorithm [32] which were trying to find frequent sequences in a window of fixed maximum length.

In our method, we also use a multilayer hash table structure to find the candidate sequences efficiently. Another advantage of our algorithm is that it doesn't require multiple scanning through data to find the extended version of a

sequence, rather only in first iteration it has to go over the whole data. Besides, it finds periodic patterns simultaneously and revises the periods over time, resulting in a robust evolving model. In addition, FPAM is able to incorporate temporal information such as start time and duration distribution.

As already mentioned, in addition to finding frequent patterns in smart home event data, we should also be able to detect periodic patterns. In a smart home setting, not only it is important to find frequent sequences, but a large portion of target activities include those activities that are not the most frequent, but are rather the most regular ones (occurring at certain periods, such as weekly). If we ignore periodicity and only rely on frequency to discover patterns, we might discard many periodic events (like weekly events).

There has been some earlier work that tries to handle periodicity such as ED algorithm [32]; however the problem with ED and most other approaches is that they look for patterns with exact periodicity. Han et al [81] define a confidence for the pattern but they require the user to specify either one or a set of desired pattern time periods. In our approach, we define two different granules for periodicity, a fine grained granule of hourly period which can span several hours and coarse grained granule of daily period which can span any arbitrary number of days. Neither 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. This is another advantage of our method over previous methods.

Another issue that should be handled in smart home systems is processing of temporal information like duration and start time distributions. Most previous techniques treat events in the sequence as instantaneous. There are several exceptions to this rule such as the work by Laxman [83] which is a framework that would facilitate description of such patterns, by incorporating event dwelling time constraints into the episode description. A similar idea in the context of sequential pattern mining is proposed by Lee et al. [81], where each item in a transaction is associated with an exhibition time. In our work, we also provide the basis for calculating the duration and start time of events.

We assume that the input data is a long sequence of individual tuples. Each tuple is in the form of $<d_i, v_i, t_i>$ where d_i denotes a single data source like a motion sensor, light sensor, appliances, etc; v_i denotes the state of the source such as on, off, etc.; and the timestamp shows the occurrence time for this particular event. In our context we refer to every individual tuple as an "event" and we'll call a sequence of these events an "activity" or sometimes simply a "sequence". We also assume that data is not in stream format; rather it is read from a file and can be managed in main memory accordingly. Table 1 shows how the data might look:

Source	State	Timestamp
Light_1	ON	05/15/2007 12:00:00
Light_2	ON	05/15/2007 12:02:00
Motion_Sensor_1	ON	05/15/2007 12:03:00

Table 1 Example input file of FPAM.

The algorithm finds frequent and periodic patterns by visiting the data iteratively where the number of iterations depends on the length of the longest frequent sequence. In the first pass, the whole data is visited to calculate initial frequencies and periods, and in the next passes only a small portion of data is revisited.

In the first pass, a window ω of size 2, is slid through the entire input data and every sequence of length two, denoted by s_i^2 along with its frequency, $f_{s_i^2}$, is recorded (here *2* denotes the sequence length and *i* refers to a unique sequence identifier). Each sequence s_i^2 consists of consecutive events e_i and e_{i+1}. To have a more efficient access to sequences, the sequences are stored in a hash table where the key to each sequence s_i^2 is $e_i \circ e_{i+1}$ where $\circ$ denotes the concatenation operator. The frequency $f_{s_i^2}$, for each sequence s_i^2 is simply the number of times s_i^2 is encountered in the data. In order to mark a sequence s as a frequent sequence, its frequency should satisfy certain conditions. For finding frequent sequences, a simple fixed frequency threshold can not result in discriminating frequent from non-frequent sequences very well as the input's size is not fixed and might vary

between different scheduled mining processes. To avoid this problem, ED [32] and other algorithms use a formula similar to the the one shown in Equation 1 for determining if a sequence is frequent or not:

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

In Equation 1, f_a represents the frequency of sequence a, $|a|$ represents the size of sequence a, $|D|$ represents the input data size as the total number of present tuples and C represents the compression threshold. In our experimentations with the frequent-periodic activity miner (FPAM), we found that the above frequency criterion does not yield very good results because the compression ratio decreases as the input size grows. Increase in input size over a fixed duration (for example one week) can only happen if resident's activity level changes (more activities per day). For example, if resident has a very active day, $|D|$ will take on a high value and if resident is not active that day, it will have a low value. Therefore for the same activity with the same frequency (such as making coffee that happens twice a day), the compression value will depend on resident's activity level or input size, which is not a correct assumption.

According to the above argument, we need to sure that the compression ratio is independent of a resident's activity level. In our approach, we simply replace input size $|D|$ by the length of the input data in hours, i.e. if t_s is the data start

time and t_e is the data end time, we can replace $|D|$ by $|\Delta t_h| = t_s - t_e$ where $|\Delta t_h|$ is expressed in hours. This replacement makes the compression ratio independent of activity level and thus provides a more accurate definition of frequent patterns.

We consider two different aspects to determine if a sequence is frequent or not: compression rate combined with cutoff threshold, and above average frequency. Compression rate alone can not be a good indicator of frequency as it is usually very low and the comparison might not always return meaningful results. Therefore, we allow only for a percentage of sequences that have high compression values (cutoff percentage of sequences). The other case is when one sequence's frequency, f_a is above the average frequency of frequent sequences, f_μ. This latter case can help identify frequent sequences even in a home where the activity level of the resident is below the expected level such that even if the compression values of all sequences fall below the compression threshold, using this method, sequences with frequencies above average will be returned as relatively frequent activities.

Therefore, for a sequence to be considered as frequent, the following two conditions should hold:

$$\frac{|a| * f_a}{|\Delta t_h|} > C \qquad \textbf{(2)}$$

$$a \in F_{cutoff} \quad \textbf{(3)}$$

or

$$f_a \geq f_\mu$$

In Equations 2 and 3, F_{cutoff} shows the cutoff percentage for top frequent sequences and the rest of the parameters are the same as described before. In summary, for a sequence to be considered as frequent, its compression value should be above a compression threshold and it should be among the top frequent sequences (shown by cutoff percentage) or have an above average frequency. Applying all these conditions together helps to avoid marking transient patterns as frequent ones while finding patterns that are frequent independent of the resident's activity level.

In addition to finding frequent patterns, FPAM is also able to discover periodic patterns. Calculating periods is a more complicated process. To calculate the period, every time sequence s_i^2 is encountered, we will compute from the time that has elapsed since its last occurrence. More precisely, if we denote the current and previous occurrence of a sequence as s and s_p, and their corresponding timestamps as $t(s)$ and $t(s_p)$, then the distance between them is defined as:

$$d_s = \Delta t = t(s) - t(s_p) \quad \textbf{(4)}$$

This distance is an initial approximation of a candidate period. To determine periodicity, as mentioned before, two different periodicity granules are considered: coarse grained and fine grained periods. Coarse grained periods represent daily periods such as "every 3 days" and fine grained periods represent hourly periods such as "every 3 hours". One can claim that only a fine grained period can be sufficient to show periodicity of an activity. For example, every Sunday can be represented by a period of 7 × 24 hours. This claim is not substantiated in practice, however, as taking such an approach will require the activity to happen every 7 × 24 hours with a tolerance of just 1 hour. This is not a realistic assumption, as we want to allow for more tolerance in coarse grained periods. For example, consider the scenario when a resident might watch TV every Sunday, but at different times; in this case, a fine grained period is not able to catch periodicity as its tolerance is just one hour while a coarse grained period is easily able to catch such a periodicity as it allows for a tolerance of one day. The same claim can be made about other time granules, but for sake of simplicity and demonstrating the basic idea, we will just consider the two levels of temporal granules.

To construct periods, a lazy clustering method is used. As long as an activity's period can be matched with previous ones (with a tolerance of one hour for fine grained and one day for coarse grained), no new period is constructed. If the new activity has a period different other than previous periods, a new period is constructed and is added to the list of candidate fine grained or coarse grained periods. In order to make sure that candidate periods are not just some transient accidentally pattern, they are kept in a tentative list until they reach a confidence frequency value. When they reach this frequency threshold, they will be moved into the appropriate consolidated fine or coarse grained period list. In order for a period to be moved into a consolidated list, certain conditions should be met. One might assume that a single frequency threshold might work, but as we will discuss below, it can not catch the periodicity of all activities.

One reason that why we can not use a single frequency threshold is that the number of times that an activity occurs can vary for different periods. For example, consider the case where our input file contains two weeks data and there are two periodic activities: a_1 with a period of one hour and a_2 with a period of 4 days. In this scenario, the number of times we expect to see a_1 would be much more than a_2. Therefore, a single confidence value can not work for both of them. To work around this problem, we calculate expected number of occurrences, $E(f_a)$, for an activity a (until current point in time) and for each new occurrence of a we check it against the following equation where f_a is actual

number of occurrences so far and ζ is a pre defined threshold that determines what percentage of expected occurrences is sufficient to move a tentative period into a consolidated list (it can be different for coarse grained and fine grained periods as ζ_f and ζ_c):

$$\frac{E(f_a)}{f_a} > \zeta \qquad (5)$$

Note that 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. In this approach, whenever more data becomes available (the mining of daily activity data is scheduled regularly) the periods are revisited again, and if there is any period that does not meet periodicity criteria anymore, it will be moved from the consolidated list into the candidate list. Later if we again find more evidence that this period can be consolidated, it will be moved back into the consolidated list. This approach results in a more robust model that can evolve and adapt over time. Figure 2 shows a schematic diagram of consolidated and candidate lists. According to this model, a sequence that has either a fine or coarse grained consolidated period is called a periodic sequence.

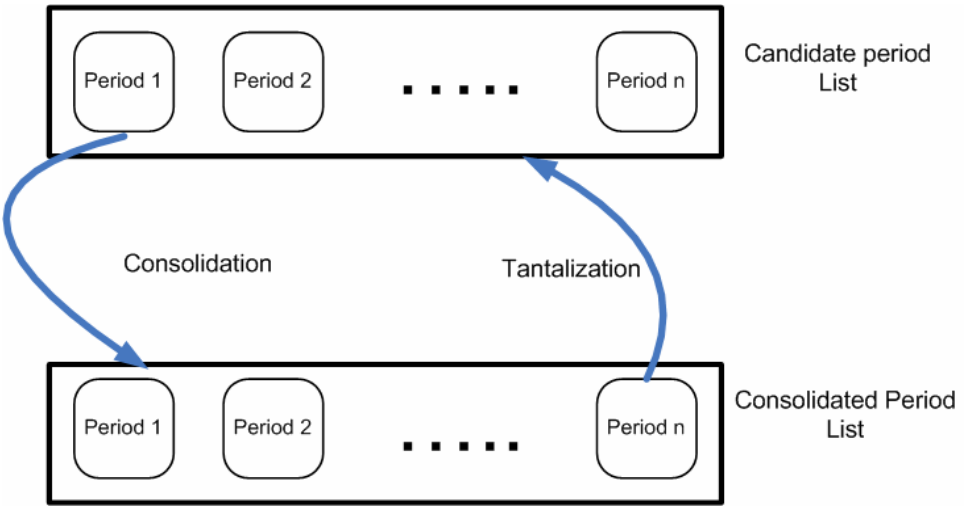

Figure 2 Consolidation and tantalization of periods dynamically.

After the frequent and periodic sequences have been identified in the first iteration, the next iteration begins with the sequences of length two in the hash table. However, we do not revisit all the data again; rather we try to extend the window size for frequent (or periodic) sequences to the left and right of the current sequence. Therefore we identify frequent and periodic sequences of length 3 and the sequences of length 3 will again be stored in a hash table for easy access. Continuing this process results in the generation of a multi hash-table structure where each hash table holds sequences of increased size (by a factor of one) compared to the previous hash table (see Figure 3).

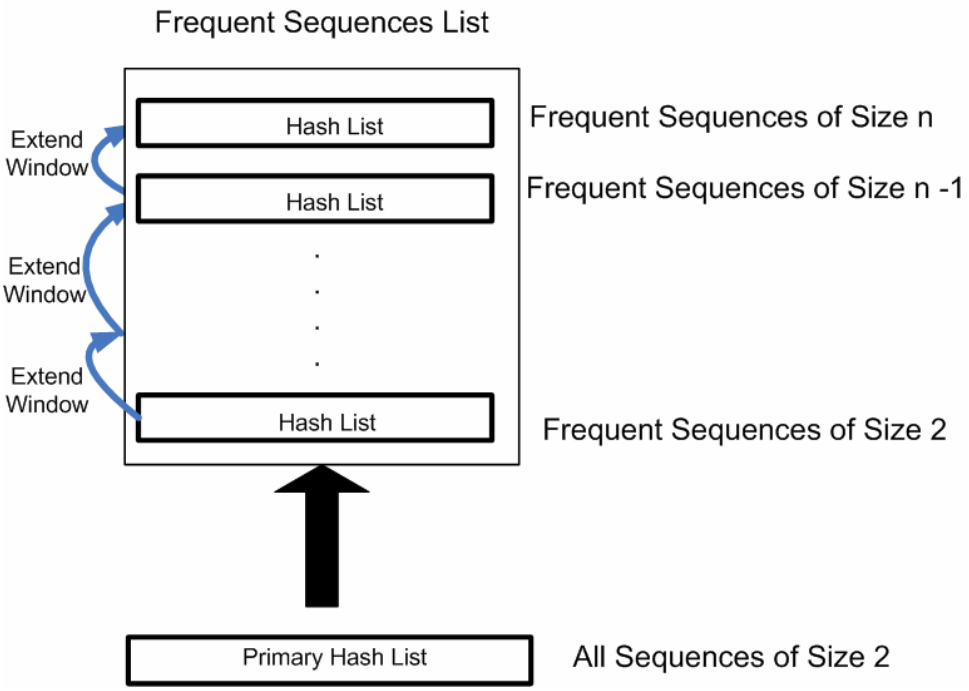

Figure 3 Constructing hash tables through window expansion.

Incrementing the window size will be repeated until no more frequent sequences within the new window size are found. At step *n*, with the window size of *n+1*, if all occurrences of a sequence *a* have length *n+1*, then the smaller size sequences which are subsequences of the patterns in *a*, will be discarded, resulting in discovering the maximal sequence correctly. Pseudo-code for the FPAM algorithm is shown in Figure 4.

In addition to finding frequent and periodic patterns, FPAM records duration and start times of events by processing their timestamps. This new source of

information is vital in smart home scenarios as it can be used to determine temporal relations between events or activities and also to calculate start time and duration distributions.

```
Data: Activity file, set of parameters
Result: Frequent and periodic sequences

// First generate the primary hash table that includes
// all 2 events sequences
Generate primary hash table

// Set the length to 2
L = 2

FP_L ={hash table of all frequent and periodic sequences of length L}

while FP_L ≠ ∅ do
    foreach s ∈ FP_L do
        Extend window ω to right
        Extend window ω to right

        // if satisfying frequent or periodic constraints
        if s frequent or periodic then
            FP_{L+1} = FP_{L+1} ∪ s
        end
    end

    // Increase length
    L++
end
```

Figure 4 FPAM pseudo-code

In summary, our current method for identifying periodic and frequent sequences has several advantages. First, it provides a framework that integrates finding frequent and periodic sequences simultaneously. Second, instead of generating an

exponential number of candidate sequences as in the Apriori algorithm, or instead of considering a fixed size sliding window as in most alternative approaches, FPAM discovers patterns by sliding a variable length window symmetrically and incrementally over data. The symmetric expansion of window allows for patterns to grow both forward and backward in time and incremental expansion allows for discovery of variable length patterns. In addition, this approach provides an efficient method by not iterating through the whole data every time. Instead, FPAM expands the window by one for each candidate to the left or right. To elaborate, consider stage *n-1* where candidates of length *n* will be generated. In the Apriori algorithm, this result in generating all possible candidates as different combinations of previous patterns which would be an exponential number of candidates generated at each step. Then input is searched to match those candidates. In our approach for each candidate sequence of length *n-1*, only two extensions will be generated (left and right extensions) which results in generating an overall number of *2m* extensions at each step (considering *m* as the number of frequent patterns found at stage *n-1*). At each iteration the whole dataset will not be revisited, rather only *m* items will be revisited (of course except for the first iteration). In order to allow for fast retrieval of sequences when extending them, for each sequence pattern, an index table is maintained that points to the locations of all instances of that particular pattern.

The other advantage of our model is evolving model of periods over time which is achieved by maintaining candidate and consolidated lists of patterns. Finally, FPAM keeps track of duration and start times for periodic and frequent patterns which later can be used to determine temporal relations or distributions. We will be using this information in HAM structure accordingly to calculate start time and duration distribution and also to determine the temporal relations between different events and activities.

Triggers

An important notion that can be used to improve activity prediction in smart homes is the notion of triggers. Basically, a trigger is an event which causes an activity to start every time the event occurs. It can be thought of as rule composed of a condition and a consequence, such that: trigger → activity. So far, we assumed that each activity can happen either at repeatable periods or based on patterns obtained for a frequent activity. However, an activity also can be started whenever it is triggered by other events. One example is if someone opens the door, this event may trigger the lights to be turned on. In our work, we incorporate the notion of triggers into our system and the triggers consist primarily of motion sensors.

To accommodate the notion of triggers, each activity in HAM will include a dynamic set of triggers, and the HAM model can be queried to return triggered

activities whenever a trigger event is executed. However, to define triggers based on results obtained from FPAM, several changes should be made to these results.

Ignoring the triggers concept, frequent patterns will be mined regardless of whether the events in these patterns are triggers (motion sensors) or automated/manual device interaction events (actuators). However, from a practical point of view, a motion sensor can not be part of a scheduled activity as we can not make somebody to walk into the room at certain scheduled times! Therefore it is necessary to process activities that include a trigger as one of their events. We take the following policy in post processing the mined sequences:

- ✓ If a trigger happens at the end of an activity, just ignore it by appropriately trimming the activity and deleting the trigger node.

- ✓ If a trigger happens at the beginning of an activity, it is considered as the firing condition for that activity and its corresponding node is deleted (it is no longer considered part of the sequence, but rather is a condition).

- ✓ If several triggers happen consecutively, we will just consider the last one.

- ✓ If a trigger happens in the middle of a sequence, we will split the sequence into two sequences where the trigger becomes the firing condition of the second sequence (see Figure 5).

- ✓ If a sequence contains more than one trigger, the above steps are repeated recursively.

Note that we assume that frequency and period would be the same for split sequences as the original sequence. On the other hand, the compression value may change as it depends on a sequence's length. Therefore, the compression value is computed for new sequences and if it doesn't satisfy the frequency criteria, it will be removed from the frequent patterns' list. 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). However, for sake of simplicity and also efficiency, we will not mine the data again; rather we will choose the sequence with the highest frequency and will discard the rest.

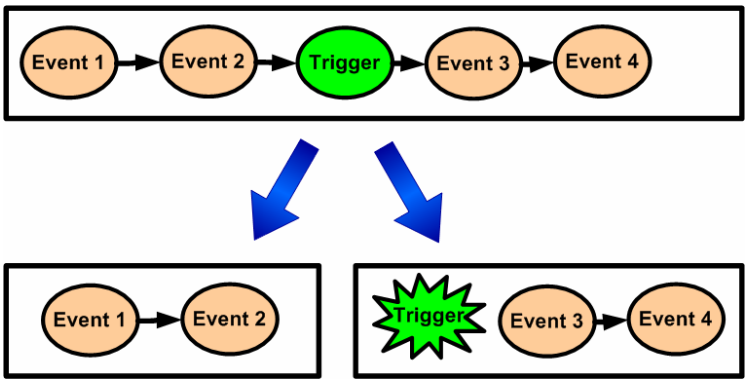

Figure 5 Trigger splitting

Hierarchal Activity Model

The Hierarchal Activity Model, which is called HAM for short, represents a hierarchal abstract representation of smart home activities and provides a basis for demonstrating temporal information among various activities. By utilizing structural and temporal information in a hierarchal structure, HAM can reveal the underlying structure in activities that hard to capture just based on the raw results of FPAM. Therefore, by gaining a better insight into daily activities' structural and temporal patterns, it can better predict potential activities that need to be automated. HAM is constructed from the results of running FPAM on daily activity data and its input includes frequent and periodic patterns of activities. To be clear about our vocabulary, we call each atomic change in the state of a device an "event" and a sequence of these events an "activity". For example, "TV: On" and "Coffee maker: Off" are examples of events and "TV: On - Coffee maker: Off" is an activity of length two, containing two events.

The HAM structure can be considered as a hybrid model of a simple decision tree combined with Markov decision processes. According to the machine learning literature, a decision tree is a predictive model that provides a mapping from observations about an item to conclusions about its target value [85]. In a

decision tree, leaves represent classifications and branches represent conjunctions of features that lead to those classifications [73]. To construct a decision tree, there are various approaches to select the best attribute closer to the root. In our case, we will employ a fixed structure for our decision tree; therefore there is no need to look for the best attribute assignment.

On the other hand, a Markov decision process, which is sometimes called a Markov chain, is a discrete-time stochastic process, consisting of a number of states with transitions between them. At any moment in time, the system might transit to a new state or stay in the same state. A Markov decision process has the Markovian property, which means that the next state solely depends on the present state and does not directly depend on the previous states [74]. This assumption holds in our model, as none of the transitions in an activity depend on previous states or transitions.

In our HAM, each frequent activity is a leaf of a decision tree that corresponds to a Markov decision process. More precisely, in our model, we define each leaf of the HAM structure as a Markov process, p_i, where p_i contains n states s_{ij}, each corresponding to one of the n individual events of a frequent activity. There are also $n-1$ transitions between states, representing the sequential nature of the activity. For example, Figure 6 shows the Markov process for activity "TV: On - Coffee maker:: Off".

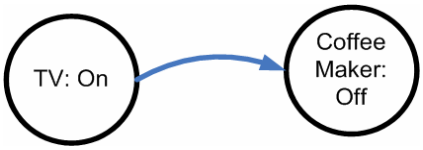

Figure 6 A simple Markov decision process.

As we will define in the "Temporal Information" section, we consider two temporal granules as the classification attributes of our decision tree which also determine the structure of HAM. These two temporal granules include "hour" and "day". The first level of HAM is based on the day granule which classifies frequent activities based on the day of week they have occurred and the second level classifies each day's activities into the finer grained granule of hour (see Figure 7). In our model, the number of intervals in the second level is a tunable parameter and can be changed to represent a desired time interval in hours. We usually set the number of intervals to 24 to allow for the finest level of accuracy.

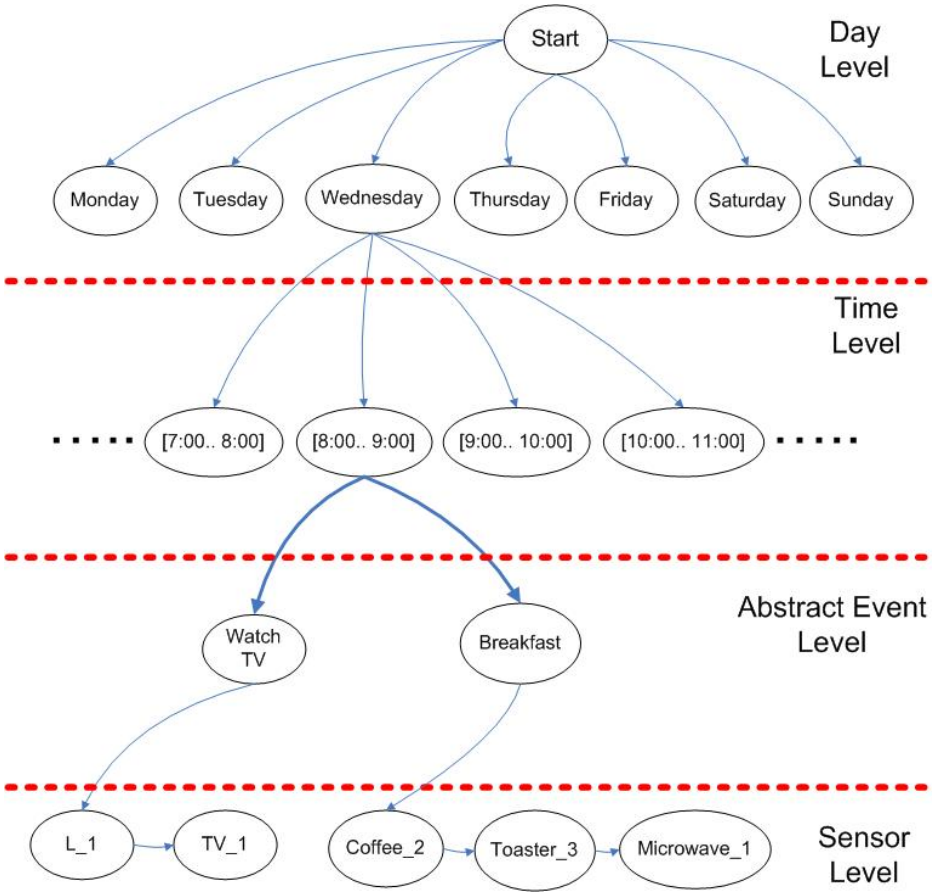

Figure 7 HAM Model

As mentioned already, the HAM model is automatically constructed from the results of the FPAM algorithm. Whenever a new set of results, *R,* is available, it is passed to the root of tree. The root filters out frequent activities according to their day of occurrence (also computing frequencies for each day) and then passes them down to the appropriate day node. Each day node filters out activities based on whether their start time falls within a time interval that is represented by the

corresponding time node and computes the frequencies for each time node. Each time node can contain several abstract nodes where each abstract node refers to an episode (activity) represented as a sequence of events (states). States of the abstract node are represented in the lowest level of the model, called the sensor level. The states along with their transitions form a Markov decision process for each activity.

The HAM structure can be used to find activities that should be automated at a given time. For example, consider that it is Wednesday morning between [8:00 ...9:00], then according to in Figure 7, CASAS can easily find out which activity should be automated. In this case the selected activity is the breakfast activity as it satisfies the given time constraints. However, as there is usually more than one activity in each time node, a more complicated approach is taken which we will describe later.

Decomposing activities in such a manner has several advantages. First, it provides an efficient method for decision making. CASAS can easily find out which activities should be scheduled to execute automatically. Second, incorporating two different levels of granules allows differentiating between similar activities that happen in different time contexts. For example, it is reasonable to assume that the resident's habit is different on Sunday between 8 - 9 AM than on Monday between 8 - 9 AM. Another advantage of decomposing

activities by their occurrence time is that it provides a better approximation of start time and duration distributions, as it can approximate start times and durations using a set of Gaussian functions rather than a single one. Figure 8 shows how this type of more complex function can be generated by combining a number of Gaussian distributions.

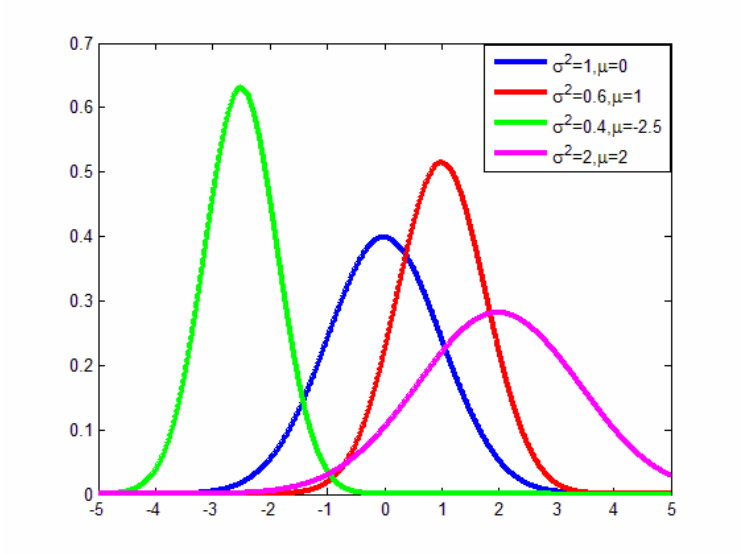

Figure 8 a complex function that is generated by combination of Gaussian distributions.

Some other models such as ProPHeT [155] have tried to achieve a hierarchal decomposition of activities by using only a location level and decomposing activities into bedroom, kitchen, bathroom, and living room activities. However, it does not provide a practical method of hierarchal decomposition as a number of locations might be different from one residential home to another. In

addition, ProPHeT ignores the whole available temporal context information. For example, it is not able to distinguish an activity that happens only on Sundays and Saturdays.

In addition to the HAM structure for modeling frequent sequences, we also consider an accompanying model that keeps all periodic activities and is basically responsible for scheduling the next periodic activity.

Predicting Next Automated Activity

As mentioned in the previous section, each activity corresponds to a Markov decision process where each model state denotes the "state of a device" and transitions between states in the model denote transitions between different states of the world. More precisely, a Markov Decision Process is a tuple <*S, A, P, R*> where *S* is the set of states (all events of an activity on our case), *A* is the set of transitions, *P* is the probability that action *a* in state *s* at time *t* will lead to state *s'* at time $t + 1$ and is defined as:

$$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a) \quad (6)$$

In our model the above probability is 1 for each pair of connected states as there is only one transition from one state to the next in each activity. Also, in a Markov decision process, *R*(*s*) is the immediate reward (or expected immediate reward) received in state *s*. However, in the HAM model reward can not be

determined in advance and depends on users' feedback and advice. The rewards are based on scale of 1..5 where user can rate an activity as extremely disliked (1), disliked (2), neutral (3), liked (4), or extremely liked (5). Each of these ratings correspond to a reward value ranging from [-0.9 ...+0.9] where extremely disliked activities receive a reward of -0.9, disliked activities -0.4, neutral activities 0.0, liked activities +0.4 and extremely liked activities a reward of +0.9. The same reward will be assigned to all states of an activity as they are considered as a unit responsible for current feedback. The rewards are used to update the function value of a state, Q_s, during guidance-based learning and observation-based learning which will be described later in the dynamic adaptation section.

After the hierarchal model is constructed and updated, there are a number of different activities in every "time node". A crucial decision-making problem regarding automating these activities is how to schedule activities in such a way that: first, the most expected activities are given a higher chance of being automated; second, other activities (especially recently added ones) also have a chance of being explored, though it might be lower; third, the temporal relations between activities are preserved (activities are scheduled correctly as a maximal non-conflicting set of activities); and finally, exploration takes its place in relation to exploitation over time.

The reason why we need to schedule selectively in a time node and can not simply return all the activities inside a time node, is that there might be time conflicts between different activities. For example, consider the following figure, Figure 9, which depicts that activities "A" and "B" and also activities "C" and "D" have time conflicts, i.e. one activity starts while another activity has not been finished yet. Note that one implied assumption behind our current system is that we are employing a serial sequential mining algorithm, not a parallel one. This in turn implies that task automation will be done sequentially, not in parallel, because we are performing a sequential data mining that can not provide enough evidence of how to generalize for doing tasks in parallel.

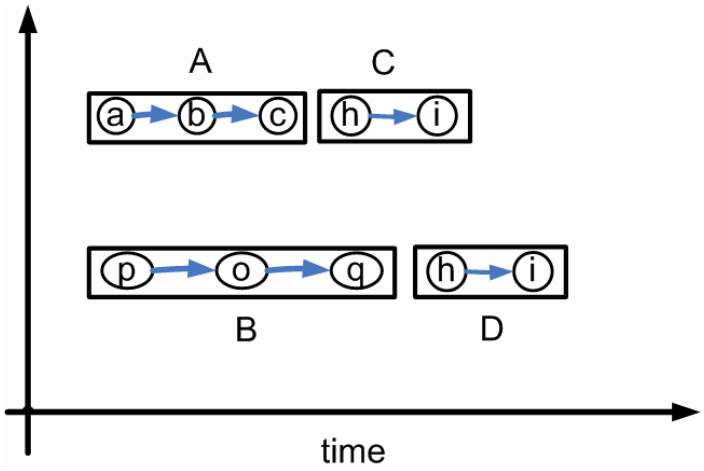

Figure 9 Time conflicts.

One might ask how time conflicts might appear in the data if we are employing a serial sequential data mining method. One possible source of time conflict arises due to different possibilities even for a given time in a specific time node of a specific day. For example, suppose we mine data from 4 weeks of residential activity and are considering the specific time interval of [7:00 ... 8:00] PM on Sundays. It is possible that the resident always does not do the same thing in this time interval. For example, the resident may spend half of all these Sundays watching TV and the other half studying[1]. Our algorithm labels both activities as "weekly periodic" activities (however not having a 100% periodicity confidence). We do not discard either of these two found patterns, rather we will leave it to CASAS to decide over time which one is more likely by applying guidance-based and observation-based learning. Obviously just one of the patterns should appear at every scheduling decision point and not both of them, which highlights the necessity for resolving time conflicts.

This time conflict might also arise from a smart detection step. As we will point out in the smart detection section, a smart detection algorithm detects changes in previous patterns by regularly mining activity data. However, it does not modify previous activities if it finds changes in them, rather it adds a new activity with

[1] Actually this might arise due to the fact that we are not considering all the possible contextual information, for example, inhabitant might be prefer to study on sunny days and watch TV on rainy days. As weather contextual information is missing from our model, it can not differentiate between mentioned cases. To alleviate this, we do not discard either of the mentioned cases; rather will leave it to system to determine which one is more likely over time.

modified properties and again tries to resolve over time which version is more likely by applying observation-based learning and guidance-based learning. From the above description, we clearly see that we need to select between overlapping activities. It should be noted that time conflicts might not be harmful in all cases. However, considering the above explanation and also considering the fact that we are not applying a parallel mining schema, we avoid such conflicts in our algorithm.

One might say it possible to resolve conflicts by temporally sorting all the activities and then resolving their conflicts accordingly. However, it turns out that the problem is not as simple as starting from the first activity and resolving conflicts between every pairs of activities, as sometimes the result contradicts the other constraint we mentioned earlier. The other constraint, as we mentioned before, is to give a higher chance of being selected to activities with higher expected utilities (we will define expected utility shortly, but for now consider it as usefulness of an activity). For example, consider the case shown in Figure 10 where $E(x)$ denotes the expected utility for activity x.

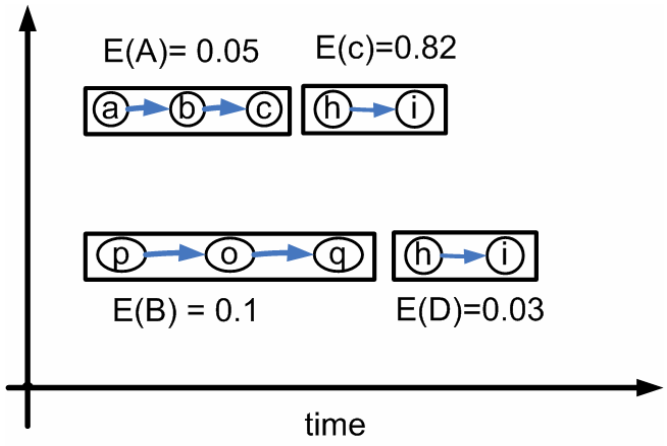

Figure 10 Expected utilities and time conflict resolution.

Considering this proposed simple time conflict resolution approach, if we start from the first activities, we have to give a higher chance to activity *B* as it has a higher expected utility. However by doing so, we simply will be eliminating the chance for activity *C* to be selected in next step as it would be eliminated before next step due to time conflicts with activity *B*.

Therefore, we take a different approach for time conflict resolution which is a greedy-stochastic method. In this approach, we do not start from the first activity, and try to resolve the time conflicts; rather at each step, we choose a random activity (a stochastic decision) according to the expected utility distributions (a greedy decision) and will remove from scheduled list any activity that has a time conflict with the selected activity. Applying this approach to the above example, activity *C* will have the highest chance of being selected at first step. In the next

step, we eliminate all the activities having a time conflict with selected activity (*D* and *B*, assuming activity *C* has been selected) and then again will choose another activity according to the expected utility distributions (*A* in above example). As we can see this procedure results in appropriately giving higher chances to more expected activities while also preserving temporal constraints.

Scheduling activities in just one time node at each step decreases the number of candidate activities considerably, thus decreasing the exponential number of temporal relations between pairs. In addition, it implicitly facilitates incorporating contextual information that are missing form our model by allowing for different activities with the same start time-day to have a chance for being selected (for example, inhabitant might do different activities on the same day-time based on weather condition. We do not model this, but at least we give it a chance).

In order to be able to choose among candidate activities that can be automated at any given moment in time, we need to define a metric for comparing different activities. The metric that we choose in our work, is the expected utility of an activity. According to Russell and Norvig [84], the expected utility of an action *A* given evidence *E*, can be computed as the following:

$$EU(A \mid E) = \sum_i P(\mathrm{Result}_i(A) \mid Do(A), E) U(\mathrm{Result}_i) \quad (7)$$

The principle of maximum expected utility (MEU) says that a rational agent should choose an action that maximizes the agent's expected utility. In our model, as we consider an activity to be a unit of execution, we assign *A* to be an activity, not a single action (event). Therefore, the above formula is translated into the following formula where $\overline{Q}(A)$ is total value function for desired activity *A* and $P_T(A)$ is total probability of transition into *A* or better said, selecting *A*.

$$EU(A) = P_T(A)\overline{Q}(A) \tag{8}$$

The total value function for the desired activity *A,* $\overline{Q}(A)$*,* is defined to be the average of all of its events' value functions, as defined in Equation 9, where *Q(s)* denotes value function of a single event *s*.

$$\overline{Q}(A) = \frac{\sum_{i=1}^{n} Q(s)}{n} \tag{9}$$

The total transitional probability for a given activity is calculated as the product of several other probabilities. In the HAM structure, there are three different types of transitional probabilities for every given activity: daily transitional probability, time transitional probability and relative transitional probability. The total transitional probability is a product of these three transitional probabilities. The daily transitional probability shows the probability of occurrence for a specific activity in each day of the week. For example, the daily transition probability of

"watching TV – Making Coffee" may be 0.8 on Sundays and 0.03 for all the other days of the week. The time transitional probability for a specific activity shows the probability of occurrence in any time interval. For example, "watching TV – Making Coffee" might happen in time interval [7:00...8:00] AM with a probability of 0.75 and might happen in the rest of time intervals with a probability of 0.25. Relative transitional probability shows how probable a specific activity is to happen with reference to the other activities in the same time interval. The sum of relative transitional probabilities for all activities in a given time interval adds up to 1 and is calculated as the following:

$$P_{ri} = \frac{P_{ti}}{\sum_{j=1}^{n} P_{tj}} \qquad (10)$$

Where P_{ti} shows time transitional probability for an activity I (will be defined shortly) and m is total number of activities in the given time node.

The time and daily transitional probabilities are calculated from frequencies found by FPAM where frequency for an activity i, n_i, is simply number of times it's observed by FPAM. As already mentioned, when HAM is constructed, sequences are distributed into different nodes, first based on their occurrence day and then based on their occurrence time. At each step, corresponding frequencies are computed as a portion of original frequency according to distribution in a given day or time node. For any activity, first the occurrence days are found and then

based on the daily distribution, the daily transitional probability, P_{di}, is calculated according to following formula where n_d denotes frequency on day d:

$$P_{di} = \frac{n_{di}}{n_i} \tag{11}$$

After filtering down the activities through the proper day nodes, the same process is followed for assigning an activity to the proper time nodes and computing time transitional probabilities in each time node. Time transitional probability, P_{ti}, is calculated according to the following formula where n_t shows frequency in a given time interval t (time node):

$$P_{ti} = \frac{n_{ti}}{n_{di}} \tag{12}$$

Figure 11 shows a schematic view of calculating probabilities by distributing activities into appropriate day and then time nodes.

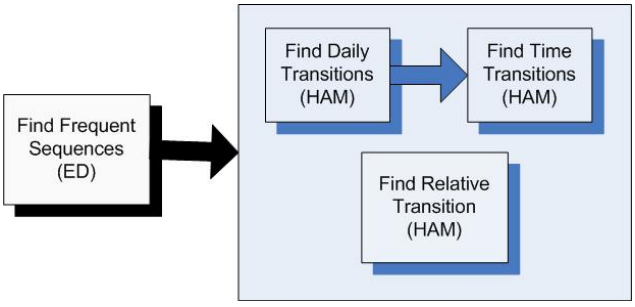

Figure 11 Process of finding transitional probabilities

If we show the total transition probability for a specific activity i as P_{Ti}, its daily transitional probability as P_{di}, its relative transitional probability as P_{ri} , time transitional probability as P_{ti}, and its compression value as C_i, then the following formula shows what is the probability of occurrence for a specific activity i on a specific day and in a specific time interval:

$$P_{Ti} = P_{di} \times P_{ti} \times P_{ri} \times C_i \qquad (13)$$

Note that in above formula we are accounting for various factors, P_{di} and P_{ti}, show daily and time distribution of an activity on a certain day and time, P_{di}, shows the probability of this activity among all other activities on the same day and time and C_i, allows for incorporating a global perspective of an activity's chance into current time and day node. The following figure shows a schematic diagram of transition probabilities in this scenario. As in our implementation total probability is multiplication of several small numbers (< 1), in order to make it easier to compare various total probabilities, we multiply it by a constant factor such as 100.

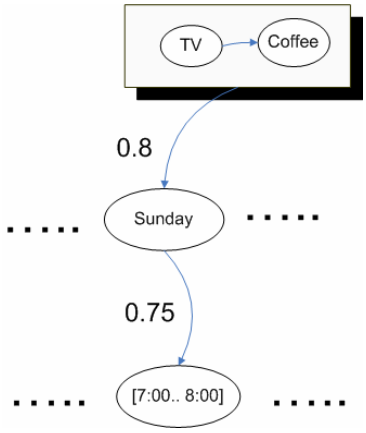

Figure 12 Transitional probability for "TV-Coffee".

By calculating total transitional probability and also value functions, we are able to determine the expected utility for a given activity. However, as we said before, besides giving higher chances to activities with higher expected utilities, we also want some other criteria to be met including: balancing between exploration and exploitation over time and also a maximal schedule of activities in given time node. To achieve this, we take an iterative approach over a set of valid activities, V where at the beginning V includes all the activities in current time node plus periodic activities that their next occurrence falls into current time interval. At each step an activity a is selected (as described below) from V and then all the activities that have a time conflict with a, are removed from V. This process is continued until no more activities can be scheduled in current time interval.

To select an activity, we take an approach that allows for a balance between exploration and exploitation. One important aspect of learning in a smart home setting is to allow for exploration of potential activities that can be automated while also avoiding user frustration due to too much exploration and randomness of decision making. Therefore it's necessary to balance between exploitation and exploration in smart home's automation policy. To achieve this, we adopt the experimentation strategy formula from [85] and apply some modification in order to fit it into our problem setting. The original formula is:

$$P(A) = \frac{k^{Q(A)}}{\sum_j k^{Q(j)}} \qquad (14)$$

In above formula, *P(A)* is the probability of selecting action A and $k > 0$ is a constant that determines how strongly the selection favors actions with high Q value functions. Larger values of k will assign higher probabilities to actions with above average potential, causing it to exploit what it has learned and seek actions that are believed to maximize the reward. In contrast, small values of k will allow higher probabilities for other actions, leading the agent to explore actions that do not currently have high potential values. k is usually varied with the number of iterations so that exploration is favored during early stages but then it's gradually shifted into an exploitation strategy. Regarding our problem setting, we alter

above formula as and again as before, we will consider A to be an activity, not a single event:

$$P(A) = \frac{k^{E(A)+\beta*D(A)}}{\sum_j k^{E(j)+\beta*D(j)}} \quad (15)$$

In above formula, we have replaced value function Q with expected utility E in order to consider various transitional probabilities besides value function. We also have added an extra term $\beta*D(A)$ which stands for considering recently added sequences. Previous models usually favor only activities with high value functions when exploring, but in our model we also try to consider recently added activities by exploring. The relative importance of recently added sequences can be changed by tweaking β accordingly. Also, in our setting k is considered to be the reciprocal of exploration rate, and in order to allow for k to change, we set it to obey the exponential decay formula to decrease effect of exploration over time. A quantity is said to be subject to exponential decay if it decreases at a rate proportional to its value. Symbolically, this can be expressed as the following differential equation, where k is the quantity and ξ is a positive number called the decay constant.

$$\frac{dk}{dt} = -\xi k \quad (16)$$

The solution to this equation is:

$$k(t) = k_0 e^{-\xi t} \qquad (17)$$

In our design, the unit of time for t in above equation is considered to be one month, setting it to a unit less than one month leads to quick degradation in exploration value. For example if t's unit is set to be one day, after a month which is usually considered as a regular time span between two consecutive data mining sessions, exploration value will be decreased by a factor of e^{-30}, thus not allowing for much exploration after the second mining session. Besides time unit, a decision should be made about decay rate as it implies the speed at which exploration would be ultimately stopped. A plot of possible values is depicted in Figure 13. Large decay constants make the quantity vanish almost immediately; smaller decay constants lead to almost-imperceptible decrease[2]. This plot shows decay for decay constants of 25, 5, 1, 1/5, and 1/25. We set decay rate to 1 to have a moderate effect (yellow plot line). Besides, we choose k_0 to be 1 at the beginning.

[2] Plot originally used in http://en.wikipedia.org/wiki/Image:Plot-exponential-decay.png. Under the terms of the GNU Free Documentation License, Version 1.2

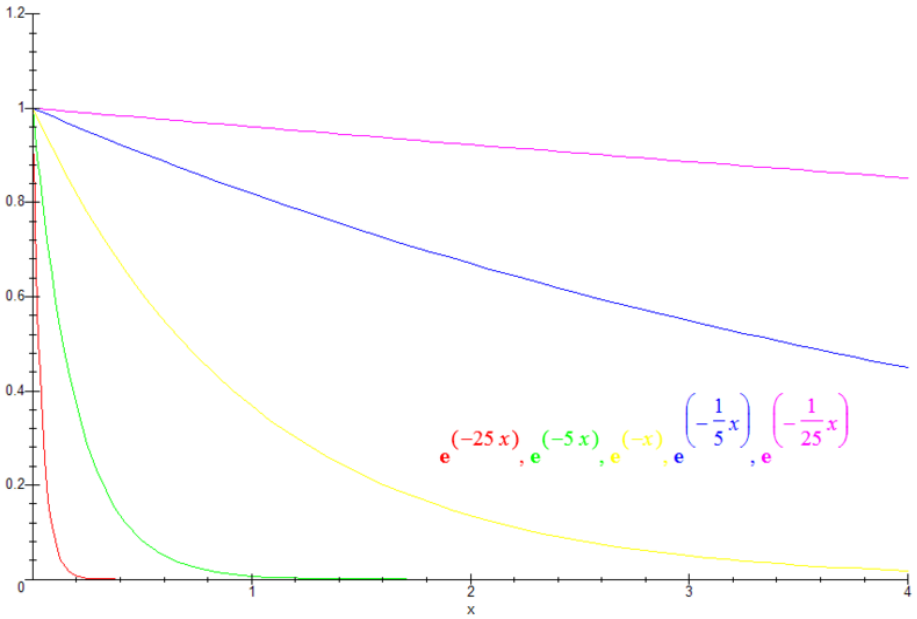

Figure 13 decay for decay constants of 25, 5, 1, 1/5, and 1/25.

The following simple pseudo-code (Figure 14) shows how events for a single time node can be scheduled accordingly.

Data: Current Time Node, T_c
Result: Scheduled activities S in T_c

// First set the valid list, V, to all frequent
// activities, F in T_c plus all periodic
// activities P with next occurrence inside
// interval T_c
$V = F + P$

while $V \neq \oslash$ **do**

  // compute chance of being selected for each $v \in V$
  foreach $v \in V$ **do**

$$p_v = \left(\frac{k^{E(v)+\beta*D(v)}}{\sum_{\forall a \in V} k^{E(a)+\beta*D(a)}}\right)$$

  end

  // if in exploration mode, randomly select a $v \in V$
  // according to distribution p_v
  if *exploration* $\geq$ *threshold* **then**
    v' = randomly select $v \in V$ according to distribution p_v
  else
    $v' = argmax_{\forall v}(p_v)$
  end

  // Insert v' considering temporal relation
  $S = v' \cup_{temporal} S$

  // remove any $v \in V$ that has a time conflict with v'
  $C_{v'} = \{$ all $v \in V$ having a time conflict with $v' \}$
  $V = V - C_{v'}$
end

Figure 14 Schedule pseudo code.

Temporal Information

To incorporate temporal information into our model, we will be considering several different temporal aspects of activities, such as the relative order of events in an activity (known as temporal relationship), temporal granules, event startup

time distributions and event duration distributions. We represent temporal relationships between events explicitly as a Markov decision process and as we mentioned before the temporal relations between activities will be preserved during activity scheduling. Simialrly, the HAM structure is used to represent temporal granules and their relations. By incorporating event start time and duration into a Markov decision process, we allow for explicit representation of the corresponding distributions.

In order to reason about temporal relations between a series of event, a number of alternative methods have already been discussed by researchers such as: Allen's temporal logic [62], point algebra [63] and fuzzy representations [64]. Allen's temporal logic, which is based on a set of thirteen atomic temporal relations between time intervals, provides very good expressiveness power but its drawback is that it has been proven to be computationally intractable; the same also can be said about fuzzy relations. In turn, point algebra, known as the less expressive tractable version of Allen's temporal logic uses time points rather than time intervals for comparison. Given two points in time, P_1 and P_2, one of the three relations depicted in Figure 20 can hold between the points [65].

Relation	*Illustration*	*Interpretation*
$P_1 < P_2$		P_1 precedes P_2
$P_1 = P_2$		P_1 same as P_2
$P_1 > P_2$		P_1 follows P_2

Figure 15 Point Algebra Relations.

Regarding the way we model an activity as a Markov decision process of several events, we can see that point algebra is implied implicitly in a Markov decision process. More precisely, if a there is a transition from state *a* to state *b*, it corresponds to the first relation point algebra such that $T(a) < T(b)$ and if there is a transition from state *b* to state *a*, it corresponds to the third relation of point algebra such that $T(a) > T(b)$ where $T(x)$ refers to occurrence time for state *x*. For example, according to the above definition, in Figure 16 we can see that $a < b < c$ where *a, b* and *c* are states (events) of given Markov decision process (activity).

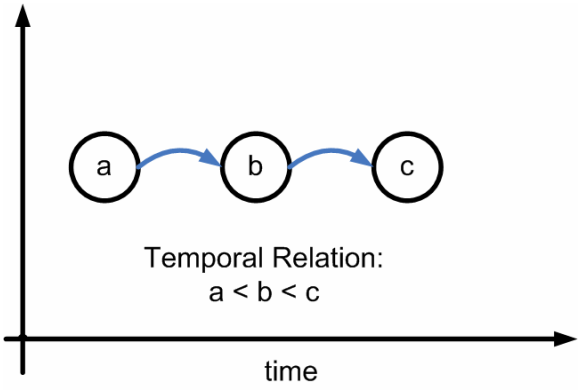

Figure 16 Temporal relation of events.

As we mentioned earlier, there is an implicit temporal relationship between different activities in a time node which would be reflected explicitly during activity scheduling for that time node, as depicted in Figure 17.

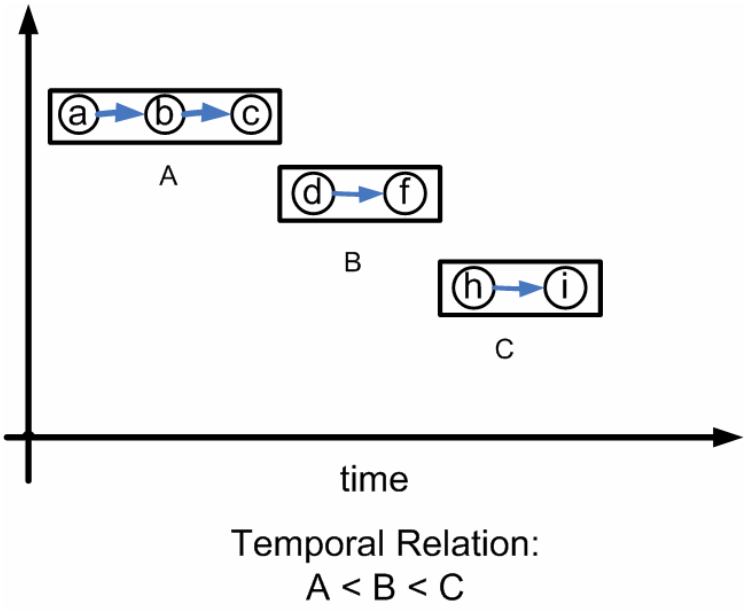

Figure 17 Temporal relation of activities.

The above discussion illustrates how temporal relations can be represented in two different levels of atomic events and activities as a set of atomic events.

Besides temporal relationships, another important aspect for temporal reasoning in smart homes is time granules, which can be viewed as a partition of a time domain in groups of indivisible elements (granules). For example, by considering the time interval [12:00, 12:45] and the event "the cooker is ON", we can note that it is defined with respect to the bottom granularity of minute. A bottom granularity represents the minimal granularity for a particular application. Some examples of granules include hour, day, month, etc. The ability to provide and relate the temporal representation of facts at different levels of granularity is an important research theme in computer science and, in particular, can be used in smart home applications [66].

A number of meaningful relationships [67] can be established between pairs of time granularities *G1* and *G2*, one such relation is "finer than" relationship. A granularity *G1* is "finer than" a granularity *G2* if each granule of *G1* is contained within a granule of *G2*. In our model, we define two different granules, hour and day, where day is finer than hour. The relation between these two granules is reflected in the hierarchal structure of HAM model where the first level of the hierarchy (see Figure 18) refers to days of week and the next level refers to hours of that day. Considering two different granules in a hierarchal manner allows us

to differentiate between activities that happen at different times and days of the week. For example, the resident might like to watch TV on Saturday at 7:00 PM while on the other days of week at the same time, s/he might prefer to study. Therefore, using two levels of granules leads to a clear separation of similar activities in different temporal contexts.

In addition to temporal relations and temporal granules, another important temporal aspect is duration and start time distributions for each event. There have been a number of approaches for modeling durations of states in a Markov decision process such as the one in [69] in which state transition probabilities are expressed conditionally on how long the current state has been occupied. In this model, the conventional fixed-state transition probabilities a_{ij} are replaced by duration-dependent variables $a_{ij}(d)$ that depend on the time d already spent in state i. In this way, state transition and state duration probabilities are combined to form duration-dependent transition probabilities.

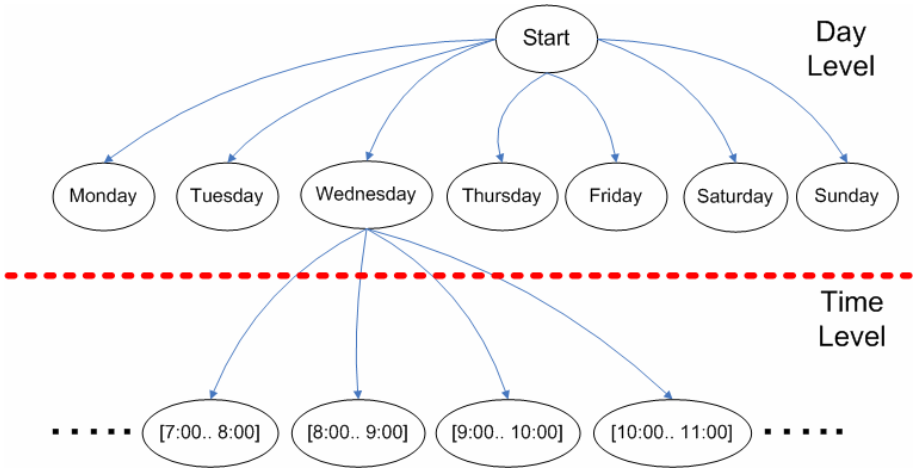

Figure 18 Relation between time granules.

In our model, we define the duration of each state *i*, as a normal distribution d_i:

$$d_i = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (17)$$

For the first state of each activity, we also consider a start time distribution by computing its mean and standard deviation. It should be noted we do not need to compute start time distributions for other states of an activity as their start time can be easily computed by adding duration amounts to the earlier events in the activity. Both start time and duration distributions are computed and updated again every time new data arrives (for example, after a regular mining session). Figure 19 demonstrates start time and duration distributions of such a Markov process.

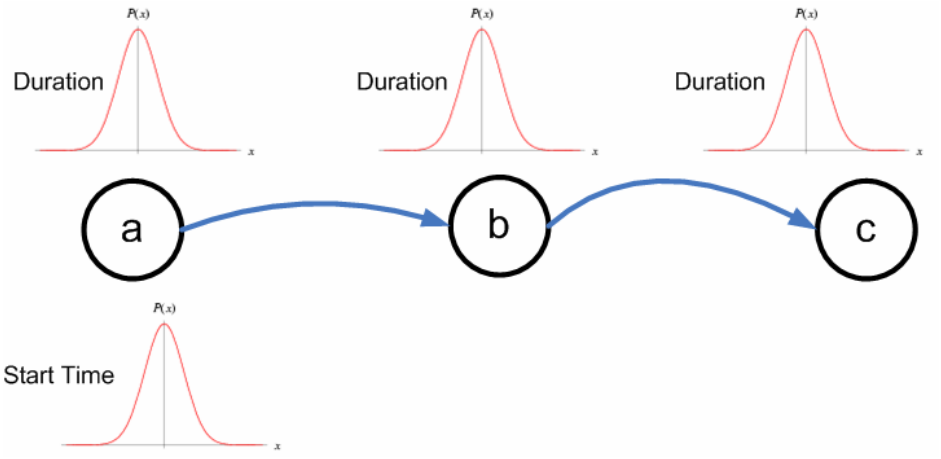

Figure 19 Start time and duration distributions.

As we mentioned before, computing the temporal distribution of an activity for each day and time node leads to a combination of multiple Gaussian functions which obviously is more complicated than a single Gaussian function and thus can approximate the start time and durations much more accurately.

Dynamic Adaptation

In most smart home environments that have been developed so far, it is assumed that the learned model is static, i.e., once we have learned the resident's frequent and periodic activities, no changes are necessary to maintain the learned model over time. However, this is not a realistic assumption as we know humans are likely to change their habits and activity patterns over time depending on many various factors such as changing job or social relations, seasonal and weather conditions and even regardless of any external factor, just based on their mental and emotional condition. Therefore, a static learned model can not be considered as a long term solution for a smart home. Instead, we need to find a way to adapt to the changes that occur over time, in addition to incorporating resident guidance and advices about automation policies.

In our solution, we take four different approaches for dynamic adaptation to changes:

1. *Explicit Manipulation:* We provide residents with an option in the CASA-U interface which allows for direct manipulation of any automated activity by changing different aspects of the activity such as structure, start time,

durations, period and startup triggers. Residents can even introduce totally new activities or remove previous ones through CASA-U.

2. *Explicit Guidance:* Residents can express their preferences using a rating system in CASA-U. CASA-U uses these ratings to automate activities more selectively based on user preferences.

3. *Explicit Request:* Residents can also use CASA-U to declare that an automated activity needs to be changed. However, in contrast to the first approach, the resident does not manipulate the automated activity directly. Instead, the detection of changes is left to CASAS. CASAS accomplishes this goal by closely monitoring daily activities and their changes and AAM mining technique which will be described later.

4. *Smart Detection:* Finally, to take away any burden of identifying changes by the resident, CASAS will automatically look for any changes in activities over time by closely monitoring all daily activities of the resident and applying regular data mining sessions.

Figure 20 shows the relationship between the dynamic adapters and the other components within CASAS. Integrating all the above approaches together provides a flexible, user centric solution to the dynamic adaptation problem in a smart home. Such a solution allows for various flexible degrees of resident

involvement: at one extreme the resident can leave all the work to CASAS and at the other extreme s/he can explicitly manipulate automation policies. At the same time it is possible to guide the smart home with different types of guidance (by rating or change request) as a moderate level of involvement.

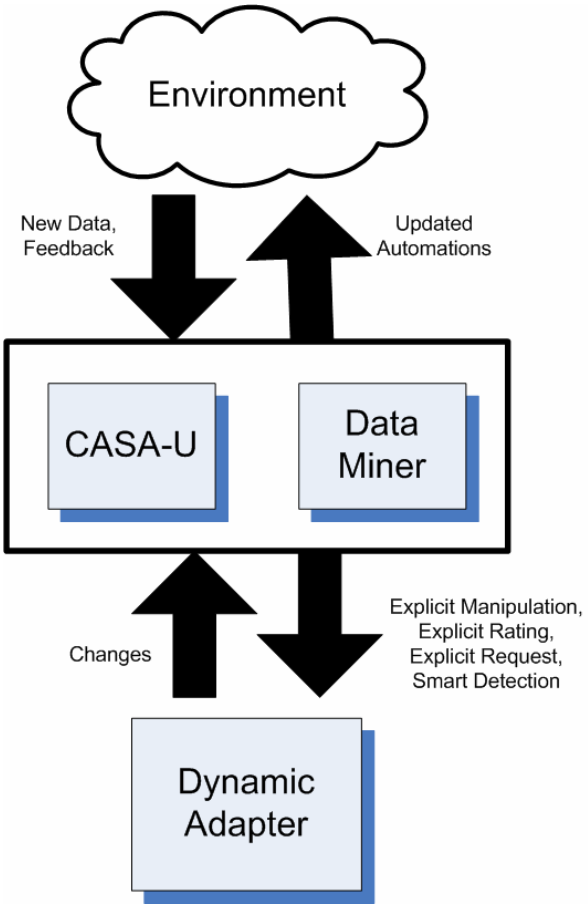

Figure 20 Interaction between dynamic adapter and other components.

The advantage of guiding CASAS over simply leaving all the work to CASAS is that it will take less time to find changes in automated activities depending on the

guidance level. In addition, resident guidance of the smart home will allow the automation policy to reflect the desired type of level of automation, rather than merely mimic all of the activities the resident would typically perform manually.

Types of Feedback in Smart Homes

In a smart home scenario, dynamic adaptation requires acquiring appropriate feedback in order to adjust the system accordingly. Feedback can be expressed by the resident in different forms, either explicitly or implicitly. In the case of explicit feedback, or as we will call it, advice, a resident should use the smart home interface to provide the algorithm with guidance and feedbacks; while implicit feedback can be expressed as immediate reversal of an action, speech commands, facial expressions or simply changing the way an activity was manually performed in the past.

In our work, we consider both implicit and explicit forms of feedback. Explicit feedback comes from rating an automated activity or manipulating an automated activity through the user interface, CASA-U. The implicit feedback is considered as changing the way an activity was performed before. For example, consider the case where the resident turns on the coffee maker at 7:30 AM every day, but later changes his habit by turning it on at 6:30 AM everyday. This is an example of implicit feedback and should be detected by the smart home accordingly.

Explicit Manipulation

As mentioned earlier, residents can use CASA-U to manipulate desired activities in order to guide automation policies of CASAS. An activity can be manipulated in many different ways. For example it is possible to change the starting time, period, and duration of each individual event as well as the startup triggers. It is also possible to add or remove an event from the activity's set of events. In addition, users can remove an entire activity. Users are even able to easily define a new automated activity through the wizard-like dialogs of CASA-U. Whenever the resident manipulates an automated activity, the changes are reflected in the HAM model by updating the corresponding sequence. In addition, adding or removing an activity results in adding or removing the corresponding sequence from HAM. To reflect the fact that the modifications are the resident's preferences over previous versions, in the feedback-based learning method described below, we will set the learning rate for the modification-version of the learning algorithm to an *explicit learning rate* (which is higher than the *implicit learning rate*) and the reward value is set to the highest possible value. The detailed explanation of the learning model and the reward values will be given in the next section.

Explicit Guidance and Guidance based Learning

Humans rarely approach a new task without presumptions on what type of behaviors are likely to be effective. This bias is a necessary component to how we

quickly learn effective behavior across various domains. Without such presumptions, it would take a very long time to stumble upon effective solutions [70]. This is necessary in large domains, where reinforcement signals may be few and far between, as in a smart home scenario. Therefore, getting advice from users can be very useful to realize a more efficient learning mechanism.

Incorporating bias or advice into reinforcement learning takes many forms, for example, in Sophie's World [71], a method is suggested to incorporate user guidance into reinforcement learning where the guided task would be taken immediately. This method is not useful in our case as we want the advice to be incorporated into the model and to be exploited over time. In another similar approach designed by Moncrieff [3], a structured method is suggested to incorporate advice into reinforcement learning and which also suggests a reward function with a lower and upper bound to avoid formation of loops among states. Again, this model is not fitting into our setting as the number of states for each sequence (corresponding to the Markov chain) is finite and we are not employing a traditional reinforcement learning method.

In our approach, we introduce a new advice taking method in order to guide CASAS's automation policies. To give the resident the opportunity to guide CASAS, we provide a rating system in which the resident can guide and give advice to CASAS by indicating if s/he likes an automated activity or not and to

what extent. As mentioned before, the ratings are based on a scale of 1..5 where user can rate an activity as extremely disliked, disliked, neutral, liked or extremely liked.

We use a modified version of reinforcement learning [61] to update the value functions based on given ratings or implicit changes in order to learn the resident's new preferences. We will call this learning method *feedback-based learning,* because it actually unifies two variations of feedback learning: guidance-based learning and observation-based learning. The feedback-based learning algorithm updates an activity's value function Q_l^{π}, according to the formula in Equation 19:

$$Q_l^{\pi} = (Q_l^{\pi} + \alpha_g^{G} \alpha_o^{1-G} r) \qquad (19)$$

In Equation 19, r denotes the reward value and $\alpha_x \in [0,1]$ denotes the learning rate. G denotes the learning type and can be either 0 or 1. If guidance-based learning is employed (for explicit feedback) G will be set to 1, otherwise if observation-based learning is employed G will be set to 0 (for implicit feedback). In order to avoid infinite growth of potential, we allow value functions to grow only in the range [0...1] by applying the formula in Equation 19, where $\overline{Q_l^{\pi}}$ denotes normalized potential.

$$\overline{Q_l^\pi} = \begin{cases} 0 & Q_i^\pi \le 0 \\ Q_i^\pi & 0 < Q_i^\pi < 1 \\ 1 & Q_i^\pi \ge 1 \end{cases} \qquad (20)$$

Guidance-based learning is used to apply user advice and guidance about smart home automation policy and it has a higher learning rate due to the explicit emphasis provided by the resident for introducing the activity. In addition to guidance-based learning, activities can also be learned implicitly. One form of implicit learning that we consider is based on monitoring changes in activities during each regular mining session; we will refer to this form of learning as *observation-based learning*. In observation-based learning, daily activity data is mined regularly (e.g., every week), so there is a good chance that most previously detected activities will be observed again (unless there is a dramatic change in activities, like new residents moving in which we do not take into consideration).

Whenever an activity is observed again, its value function will be updated according to the formula in Equation 19 by setting *G* to 0 and therefore using the observation learning rate α_o which is usually set to a value lower than the guidance-learning rate, α_g. In our setting, we set α_o to 0.2 and α_g to 0.9. Similarly, the observation reward is set to 0.1 and the guidance reward (in cases where user is creating or modifying an activity) is set to 0.7. The reason for selecting these

particular values will be described shortly. Also for ratings, the rewards will be determined based on user's feedback as already mentioned.

According to the above definitions, there is a basic difference between guidance - based learning and observation-based learning; in guidance-based learning the user expects the system to learn new preferences very quickly and possibly override previous values as s/he has explicitly expressed the preferences and changes. In observation-based learning, there is no such strong evidence for quickly changing previous models, therefore learning should be performed gradually and should preserve history. That is why we have considered two different learning rates in Equation 19, α_g^G and α_o^{1-G}, where α_g represents the learning rate for guidance-based learning and α_o represents the learning rate for observation-based learning. To simulate the fast overriding nature of guidance-based learning and gradual history-preserving nature of observation-based learning, we set α_g to a value close or equal to 1 while we will set α_o to a relatively small value such as 0.2.

The above procedure shows how the value function for a specified activity can be reinforced and updated. This might raise a question regarding similar activities. Suppose that an activity *a* happens at three different times during a day, for example in the morning, at noon and at night; this corresponds to three different versions of this activity in HAM as a_1, a_2, and a_3, each one corresponding to

occurrence of *a* in one time node. Now, suppose that user rates the "in the morning" version of activity, a_1,, as highly positive, then naturally we might ask what should be done about the other two activities at noon and at night, a_2, and a_3? Considering CASAS as a smart environment, it is a realistic assumption that users would expect it to be able to generalize beyond merely user advice; however, there is always a chance of over-generalizing, or generalizing the policy to the point where it is applied when it is not necessary.

To solve the above problem, we need an inductive bias to generalize more discriminately. To achieve this, we consider the degree of similarity between two activities as a measure of generalization. To be more accurate, we denote each attribute *i* of activity *j* as a_{ij} and the distance between attribute *i* of two activities *j* and *k* is defined as $\left|a_{ij} - a_{ik}\right|$. The total distance between activities *j* and *k* can be defined as following by summing up distances over all attributes:

$$d_{jk} = \sum_{i=0}^{n} \left|a_{ij} - a_{ik}\right| \tag{21}$$

Similarly, s_{jk} , is defined as the reciprocal of distance:

$$s_{jk} = \frac{1}{d_{jk}} \tag{22}$$

We consider generalizing between two activities if they have the same structure (i.e. same device sources and states). However, their other attributes such as durations and start times can be different. This seems to be a rational choice, as other identical attributes such as periods or durations are not good discriminating attributes. For example, there might be lots of activities that occur within the same period, but they are not necessarily related to each other. The attributes selected to determine distance in our problem setting are duration *D*, start time *t* and period *P*. Therefore we can rewrite the formula in Equation 21 as the one shown in Equation 22:

$$s_{jk} = \frac{1}{\mu_{\Delta D} + \Delta t + \Delta P} \tag{23}$$

where $\mu_{\Delta D}$ is defined to be the average difference between duration of all corresponding events in two given activities, as:

$$\mu_{\Delta D} = \frac{\sum_{i=1}^{m} \left| D_{ij} - D_{ik} \right|}{m} \tag{24}$$

In Equation 23, *m* is the number of events in activities *j* or *k* which obviously should be equal for both of them as we required the structure of the two activities to be the same in order to be able to generalize between them.

The above equations give us a way to calculate the similarity between two activities. We will thus use similarity degree to determine how to generalize the reinforcement. To reinforce similar activities, we will use the general feedback based-learning equation where the learning type is considered to be the same as original activity and the reward is computed according to the following formula:

$$r_j = \beta * s_{jo} * r_o \qquad (25)$$

In Equation 24, activity *o* is the original activity that is supposed to be reinforced and activity *j* is a similar activity that will be reinforced as a result of generalizing the reinforcement for activity *o*. β is a tuning parameter that can tweak the effect of the similar activity's reinforcement or better said the generalization degree; if it is set to 0, similar activities will not be reinforced; if it is set to 1, similar activities will receive the same reward as the original activity. In our setting, β is set to 0.5 to introduce a mild generalization effect.

In addition to activity reinforcement, we are also considering a decaying effect which basically subtracts a small value ε from all activities' value functions at each

step, for a step size of θ. By applying the decay function, the value of any activity during an arbitrary time interval Δt_d would be decreased as:

$$Q_l^{\pi} = Q_l^{\pi} - \frac{\varepsilon * \Delta t_d}{\theta} \qquad (26)$$

The decay effect allows for those activities that have not been perceived over a long period of time to descend toward a vanishing value function over time, or in an intuitive sense to be forgotten. The effect of the decay function is compensated through reinforcement. In order to avoid a large decay effect that can not be compensated by reinforcement, it is necessary to apply some constraints to the decay function. We require the constraint expressed in Equation 26 to hold, where the time span between two consecutive mining sessions is denoted by ΔT:

$$\frac{\varepsilon * \Delta T}{\theta} < \alpha_o r_o \qquad (27)$$

In Equation 26, α_o and r_o denote the learning rate and the reward value for observation-based learning, respectively. Using Equation 26, we guarantee that in the absence of any explicit negative rating, a frequently-observed activity can compensate for the decay effect.

It should be noted that though we use reinforcement terminology in our learning method, it has substantial differences from the traditional reinforcement learning

method. In traditional reinforcement learning, the whole environment is typically formulated as a single finite-state Markov decision process (MDP) while in our approach, each individual activity is considered as a finite state Markov decision process and therefore the whole environment can be considered as a set of all finite state Markov decision processes instead of a single Markov decision process. In traditional models, it is also assumed that a reward/punishment value will be provided to guide the agent at each step while this can not be assumed in smart home environments because it depends on user feedbacks (both explicit and implicit).

In addition, reward/punishment in guidance-based and observation-based learning have different meanings. In guidance-based learning, it should be considered as forcing an immediate change while in observation-based learning it is considered as evidence for gradual stabilization. As another difference, in traditional models, a discount factor is applied to rewards in future states, while in our model reward is applied to all the events (states) in the sequence (activity) uniformly as we consider the activity as a representative unit. In addition, in traditional models, no decay is applied to a value function over time while we our mode applies a decay function in order to forget inappropriate solutions. And finally, in our reinforcement formula, we integrate both guidance-based learning and observation-based learning into a single framework, resulting in a system that can stabilize over time while adapting to feedback from users.

Explicit Request and Observation Based Learning

As we already noted, CASAS provides an option to detect the changes in a specific desired activity. This option has been mainly provided to remove the burden of explicit manipulation from users such that all the work that the user has to do is simply to declare the activity that should be monitored by CASAS in the CASA-U interface.

When an activity is highlighted for monitoring, it will be added to the list of monitored activities which is then passed to the dynamic adapter component based on pre-defined scheduled times (for example, once a week). The mining schedule for monitored activities is more frequent than the normal mining schedule for updating the model and the rationale behind it is that users probably are not willing to wait a long time before a change is detected, considering the fact that they already have explicitly declared it to be monitored.

In addition to the difference in schedules, the mining process for detecting changes is different than when we use FPAM as the mining algorithm. We call the mining procedure for detecting changes as Activity Adaptation Miner or for short AAM. In the FPAM, basically we were looking for all frequent or periodic sequences, while in AAM we would be looking after potentially-changed versions of a specific activity. However, this is not a trivial search task and there are some subtleties that should considered accordingly. First of all, we should define what

do we mean by a changed version of an activity? How do we detect that it is the same activity with some changes and not a totally new activity?

To avoid confusion, we will refer to the first activity as the "original activity" and the changed versions as "changed activities". A changed activity is basically the original activity with some changes in either structure, start time, period or startup triggers. Without loss of generality, we refer to changes in two different categories and will perform detection based on the following categorization (however we do not know in advance which case is the target case):

- ✓ Changes that preserve structure.
- ✓ Changes that do not preserve structure.

The pseudo-code in Figure 21 shows how changes can be detected based on the general categories we have defined.

Detecting changes in both of these cases is complicated and requires a sequential data mining approach that is modified to detect changes in different parameters of an activity such as start time, period, structure, triggers and durations. To be able to keep track of all these different changes, a "change history" structure for the original sequence is created where each single change to original sequence is saved as a new changed version. These results will be shown to the user later in

order to get confirmation for applying correct changes. The user can apply a single change or a combination of different changes.

Data: Activity a,Activity file, set of parameters
Result: detected changes in a

```
// two types of changes will be considered:
// structure preserved
// structure not preserved
```
$X = \{$ all changed versions of $a\}$

```
// At first step, detect changes that preserve structure
```
$X_1 = \{$ all detected changed versions of a with the same structure$\}$

```
// Then, detect changes that do not preserve structure,
// but have the same start time distribution.
```
$X_2 = \{$all detected changed versions of a with the same start time distribution$\}$

$X = X_1 \cup X_2$

return X

Figure 21 Detecting changes.

Just like in FPAM where we were searching for frequent patterns, in AAM we will be looking for frequent "changes of patterns". As we pointed out before, different changes are possible, such as changes in structure, start time, period or startup triggers. According to the above defined categorization, the structure can be changed or it can be preserved. If the structure has changed, we will find the changed version of an activity by looking for any frequent patterns that happen in the same start time range, considering the appropriate deviation (assuming that

no two activities can happen at the same time). In the latter case (structure preserved), we will be looking for frequent patterns that match the given structure to discover any changes in start time, period, durations or startup triggers. All patterns that have a similar structure but show differences in any of above attributes above a given threshold will be considered as a different versions of the pattern. Figure 22 shows pseudo-code for the above procedure.

Data: Activity a,Activity file, set of parameters
Result: detected changes in a

`// structure preserved`
$X_1 = \{$ all changed versions of a that preserve structure $\}$

$A_x = \{ period_x, duration_x, starttime_x, triggers_x \}$
$\theta_y =$ difference confidence for $attribute_y \in A_x$

`// detect changes that preserve structure`
slide a window ω of length $|a|$ over all data
$X' = \{$ all found sequences with the same structure as $a\}$

foreach $a' \in X'$ **do**
  compute all $y \in A_x$
end
foreach $a' \in X'$ **do**
  foreach *attribute* $y'_a in A_a$ **do**
    if $|y'_a - y_a| > \theta_y$ **then**
      $X_1 = X_1 \cup a'$
    end
  end
end

return X_1

Figure 22 Detect changes that preserve structure.

In the former case where the structure has been changed, we need to look for a frequent pattern, *a*, such that its start time, s_a, is contained within the interval $\Delta\delta = \mu_o \pm \sigma_o$, where μ_o and σ_o denote the mean and standard deviation the original activity's start time distribution. We mark all potential $\Delta\delta_i$ values and then move a sliding window ω of increasing size over all such $\Delta\delta_i$,. Just like in FPAM, the increasing window size is only terminated when no more frequent patterns of length $|\omega|$ can be found. The detailed method is similar to our FPAM method - the only difference is that we will not looking over all available data, rather just over $\bigcup_{i=1}^{m} \Delta\delta_i$, where *m* is total number of all marked start points. Also note that a frequent pattern can easily be extended beyond $\Delta\delta_i$,. We only require its start time (i.e. the start time of its first event) to be contained within the $\Delta\delta_i$, interval. The newly-discovered frequent pattern might also have other different characteristics such as period, duration, or startup trigger. This process results in finding a totally new sequence which may be longer, shorter, or of equal size, having different properties except for the start time range.

In the latter case where we assumed that the structure would be preserved, we first mark all the occurrences of the original activity in the data, and then according to these occurrences we can calculate new properties such as new durations, new start times, new periods or new startup triggers. The detection of the original activity can be done effectively using finite state automata (FSM) and whenever the FSM enters the final state, the start position *l* will be recorded by

the algorithm. At the end of the process we will have a collection of such start points as $\bigcup l_i$. AAM accordingly uses this data to calculate new properties such a mean and standard deviation for start times, durations, periods, etc. Figure 23 shows the pseudo-code for the above procedure.

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.

Smart Detection

In addition to the above methods for detecting changes, CASAS also automatically mines data regularly (for example, every three weeks) to update the model. This approach is slower that other three previous approaches 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. However, for previous activities, the situation is more complicated, as there might be changes in start times, durations or startup triggers, or in the absence of any changes, the activity can be the same as before. If the activity shows no changes, then the only thing that needs to be done is to apply the observation-based learning algorithm by setting $G=0$ and reward as r_p as we described before.

Data: Activity a,Activity file, set of parameters
Result: detected changes in a

```
// structure not preserved
```
$X_2 = \{$ all changed versions of a that do not preserve structure $\}$

$A_x = \{ period_x, duration_x, starttime_x, triggers_x \}$
$\theta_y =$ difference confidence for $attribute_y \in A_x$

```
// detect changes that do not preserve structure

// mark all the time points δ_i withing μ_a ± σ_a
// where μ_a and σ_a are mean and deviation of
// a's start time distribution
```
$\Delta = \cup_i \delta_i$

```
// Set the length to 2
```
L = 2

$FP_L =$ {hash table of all frequent and periodic sequences of length L}

while $FP_L \neq \emptyset$ **do**
  foreach δ_i *point* $\in \Delta$ **do**
    Extend window ω to right
    Extend window ω to right

    `// if satisfying frequent or periodic constraints`
    if *s frequent or periodic* **then**
      $FP_{L+1} = FP_{L+1} \cup s$
    end
  end

  `// Increase length`
  L++
end

$FP = \bigcup_{\forall L} FP_L$

foreach $a' \in FP$ **do**
  foreach *attribute* $y'_a in A_a$ **do**
    if $|y'_a - y_a| > \theta_y$ **then**
      $X_2 = X_2 \cup a'$
    end
  end
end

return X_2

Figure 23 detecting changes that do not preserve structure.

If the activity shows some changes, we will add the changed activity to the model (no removal of original activity is necessary) and will leave it to the reinforcement and potential decay functions to decide over time which version is more likely. This leads to correct solution because reinforcement increases the value function of a frequently perceived activity while decay function decrease value function of all activities at each step and therefore for those activities that have not been perceived for a long time, it leads toward a vanished potential.

CASA-U: The User Interface

As has already been mentioned, the long-term goal of our research is to automate resident interactions with the environment that are repetitive or, in the case of individuals with physical limitations, difficult to perform. The main obstacle to the design of such 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. The contribution of our work is a user centric approach to design of the user interface for a smart home environment that can adapt to its residents, allowing the residents to play a critical role in guiding the environment's automation policy.

In our work, we try to address some of the key issues and challenges in a smart home user interface by applying a user-centered design process [86]. 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, view previous modifications through an appropriate search interface and provide feedback to the smart home based on a simulation of the home's predicted automation activities.

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

There are a few related approaches to simulating smart environments, including the ResiSim residential simulator [60] and virtual 3D [87]. While both of them rely on direct manipulation, neither addresses the issue of manipulating event sequences, collecting and responding to user feedback, and representing spatio-temporal relationships.

CASA-U design

One of the most important tools for performing evaluation and research in many areas is simulation. This turns out, not surprisingly, to be particularly useful in smart environments by playing many key roles such as visualizing resident activities and providing insights on patterns that have been automated.

In our current design, we consider the 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. In this floor-map paradigm, the

physical dwelling of the resident is directly mapped to its digital representation and sensors and controlled elements (like lights) are displayed on the map as well. Using data collected from motion sensors, the map displays the resident's location, visualized as animated footprints. We can identify several types of objects in the simulation environment: static, dynamic and interface. While static object states do not change, dynamic objects can change state. Interface objects allow either users or other external entities to interact with the simulation. Each object possesses attributes, a number of possible states, and a specific functionality. To run a simulation, we just start all of the objects; the updating will take place based on an event-driven paradigm. In our context, we call each atomic change in the state of a device an "event" and a sequence of these events will be called an "activity".

The sensor layout and floor plan of the visualized simulation is based on the sensor layout and floor plan of the AI lab at the School of EECS at Washington State University. This lab has been equipped with a number of motion and light sensors to test smart home ideas. The basic floor plan of the AI lab can be seen in Figure 24. We turned the basic floor plan and sensor layout into a visualized realistic 2D model where various lights and lamps can be turned on or off (light sensors) or door can be opened or closed (motion sensor). To show the effect of motion sensors detecting someone walking around the room, we used footprints

to imply the motion effect. Figure 25 shows the 2D simulation environment based on the floor plan and layout of the AI lab.

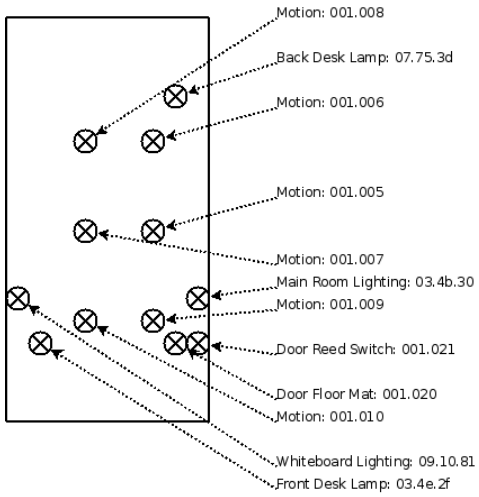

Figure 24 AI lab floor plan and sensor layout.

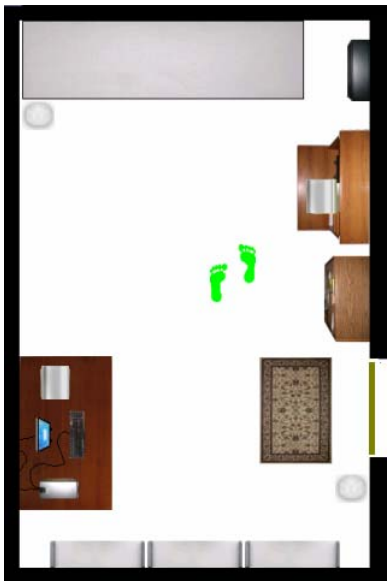

Figure 25 2D visualized model of AI lab floor plan.

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 relation (see Figure 26).

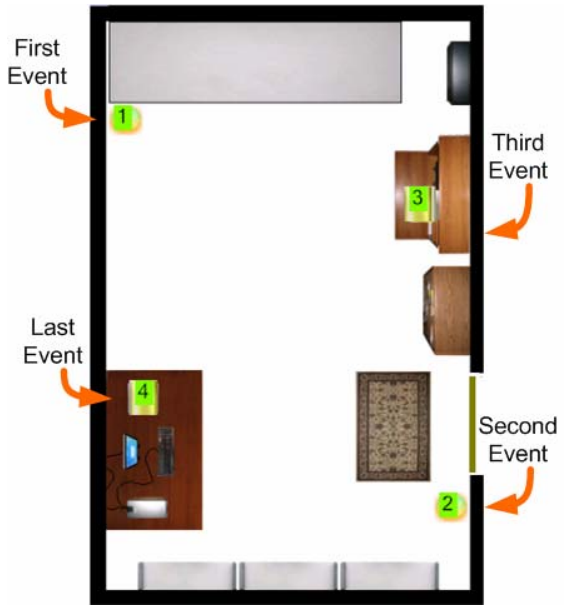

Figure 26 Displaying temporal relations using labeled events.

To avoid confusion between resident activities and past automated activities performed by CASAS, we provide two separate spatial views: one for visualizing a live stream of the resident's sensed activities (activity view) and another showing past automated CASAS activities (automation history view), as depicted in Figure 27. In both views, the user has the option to go back or forth in the stream using rewind/forward buttons in order to find a specific event.

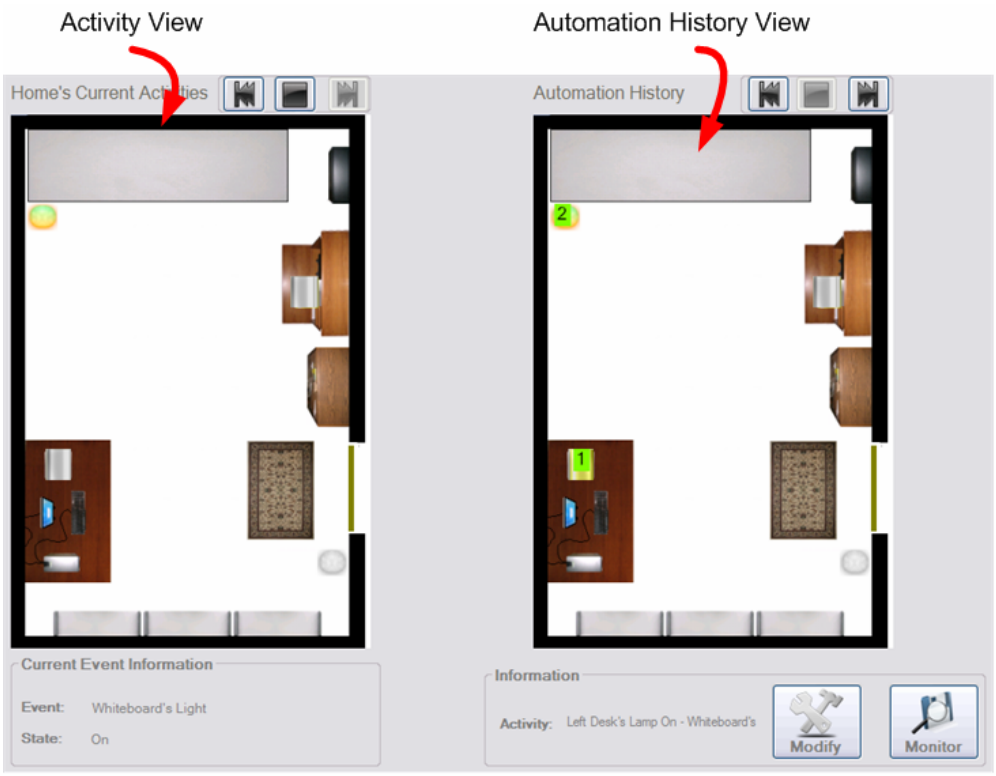

Figure 27. Dual view in CASA-U.

A serious challenge we faced in the design of the CASA-U user interface was how to efficiently provide users with the ability to change scheduled automated activities or request new automations. Each activity's model includes a definition of included events, the relative and absolute temporal relationships between these events, the duration of each event, triggering conditions and different scheduling 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.

For example, for defining a new automation activity, the user is guided through the following steps:

- ✓ Indicate whether it is a totally new activity or it exists in the history. User also can select the "don't remember" option (see Figure 28).

- ✓ A dialog box appears asking user if s/he wants to skip all the steps and leave it to CASAS to find new patterns that can be automated.

- ✓ Indicate how the activity should be started: being triggered or on scheduled times or based on both (see Figure 29).

- ✓ Define triggering events (see Figure 30).
- ✓ Define the activity's events, using either the context menu or the drag-and-drop method. It is also possible to record a set of previous events as new activity using forward, rewind and record buttons (see Figure 31).
- ✓ Set duration for each event and also period of this activity (see
- ✓ Figure 32).

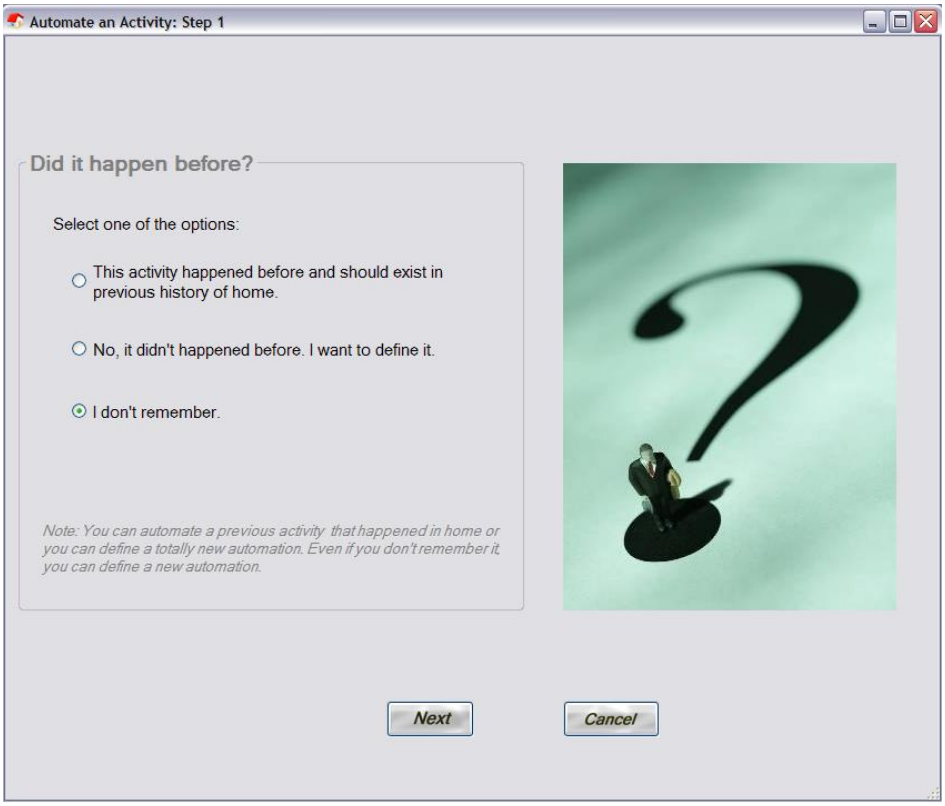

Figure 28 define a new automated activity, step 1.

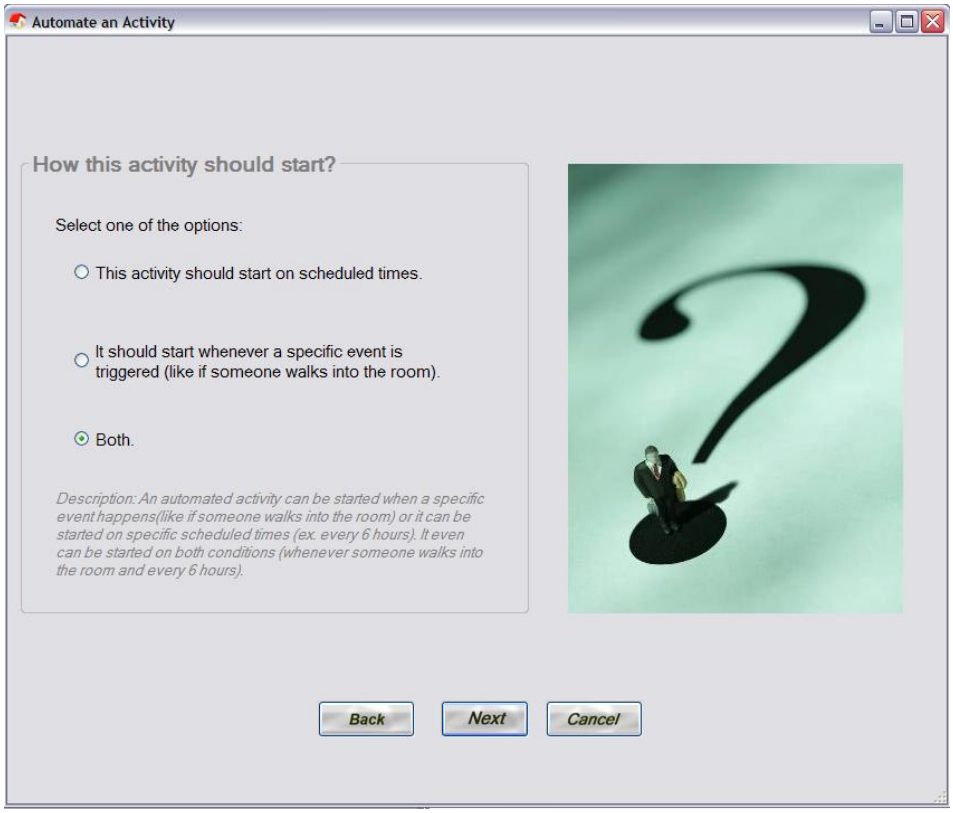

Figure 29 define a new automated activity, step 3.

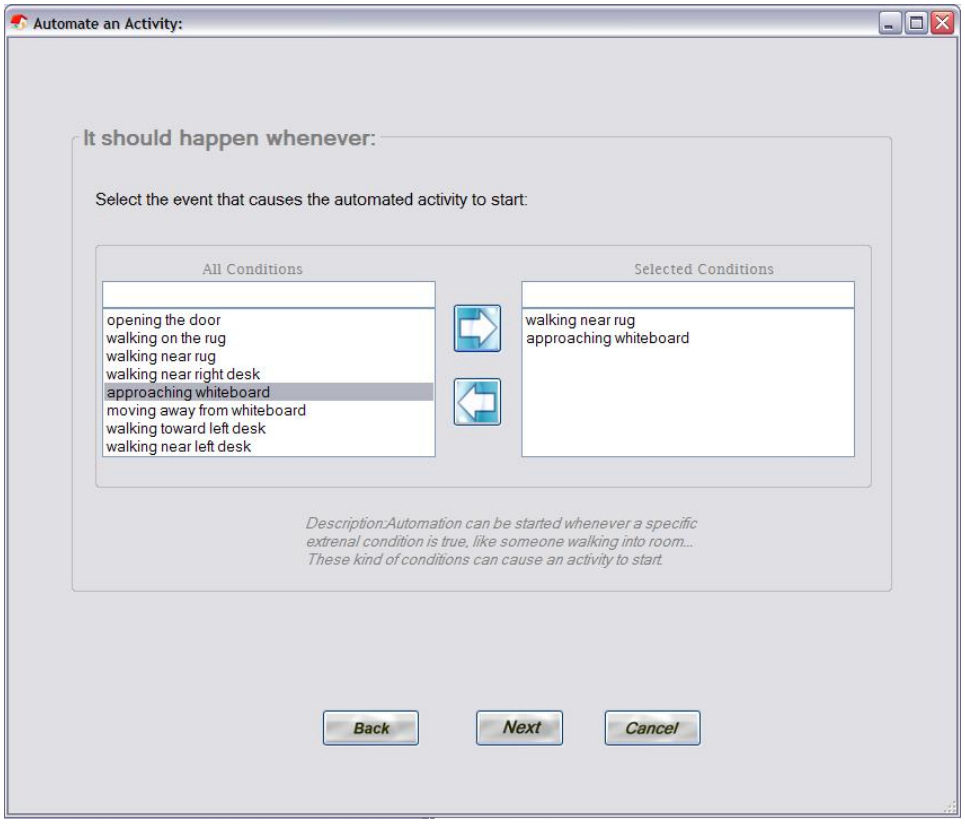

Figure 30 define a new automated activity, step 4.

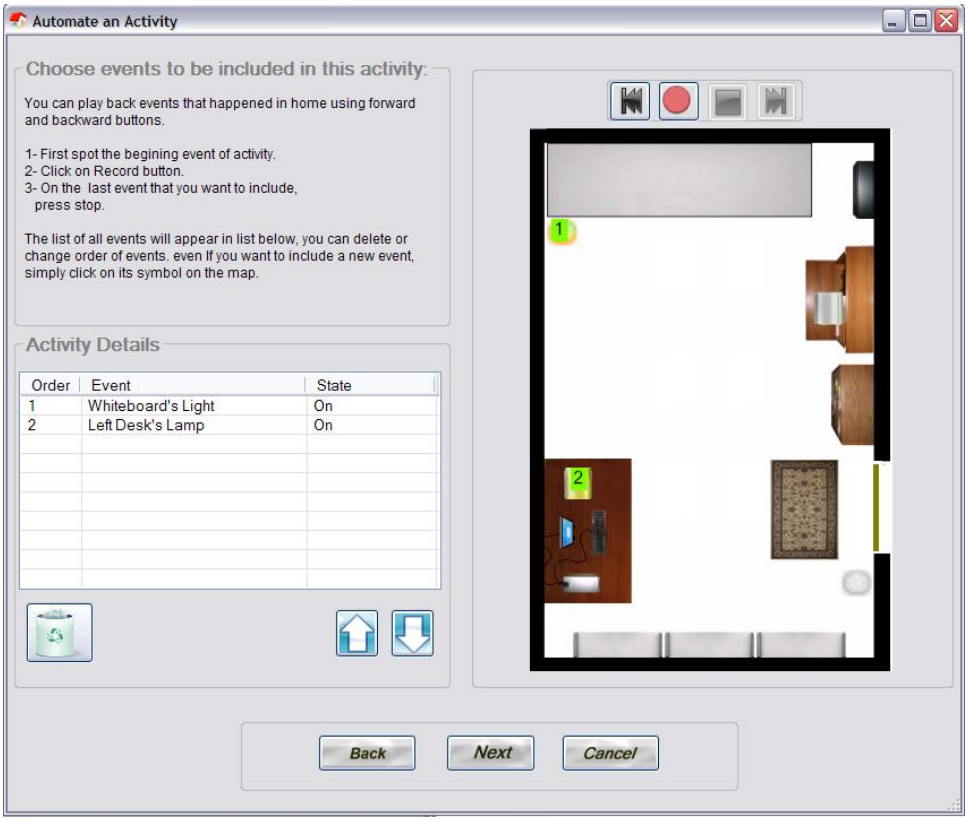

Figure 31 define a new automated activity, step 5.

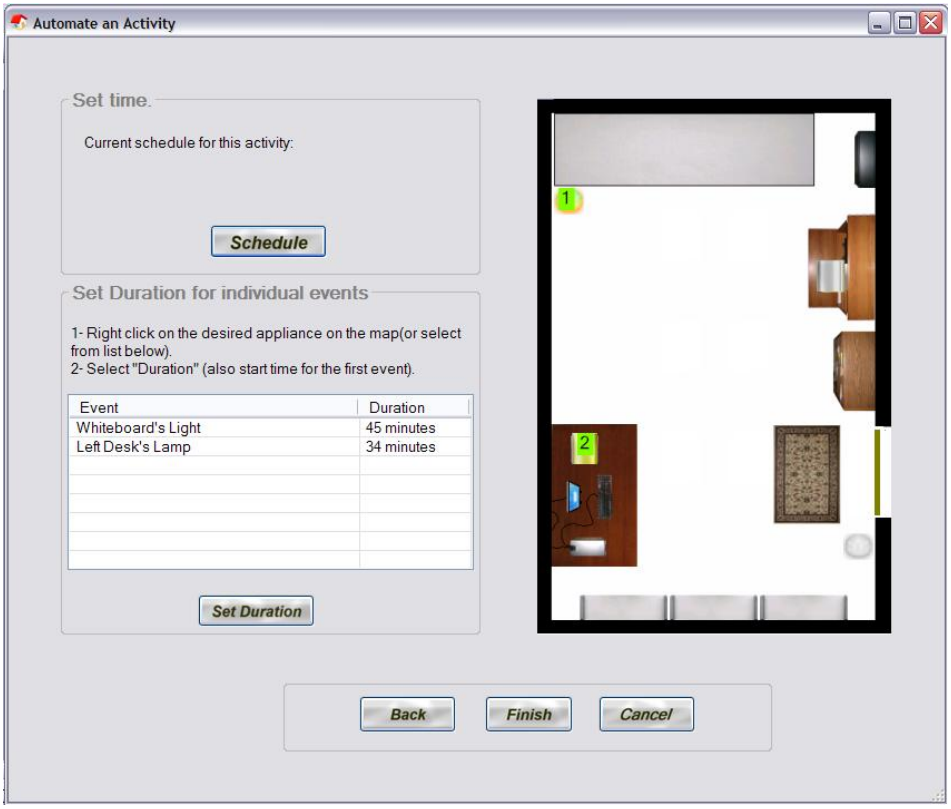

Figure 32 define a new automated activity, step 6.

Modifying an activity consists of the following steps:

- ✓ Indicate which features should be modified. The user can also select an option for removing an activity totally (see
- ✓ Figure 33).
- ✓ A dialog box appears asking if the user wants to skip all of the steps and leave it to the system to detect changes in daily routines.

- ✓ Indicate triggering events (see Figure 34).

1. Modify events included in this activity, using either the context menu or the drag-and-drop method (see Figure 35).

2. Set duration of each event included in this activity, besides provide a period (weekly, daily, and hourly) for scheduling this activity (see Figure 36).

CASA-U also provides a search option to search for desired activities based on occurrence time or included events. A crucial issue is how to display search results in a clear manner, considering spatial and temporal relations. In our current design, a separate map is used to visualize the results and the user can go back and forth between the results using a forward/rewind button. The results appear in the same order in which they have occurred. Users can also select a desired activity to modify.

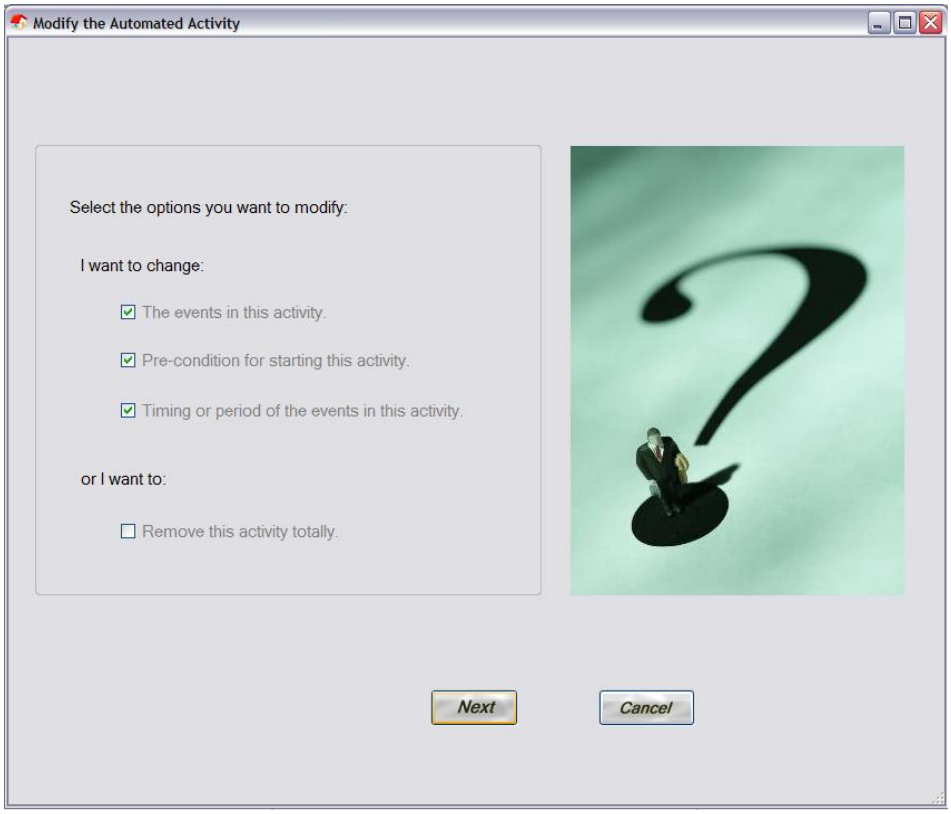

Figure 33 modify an automated activity, step 1.

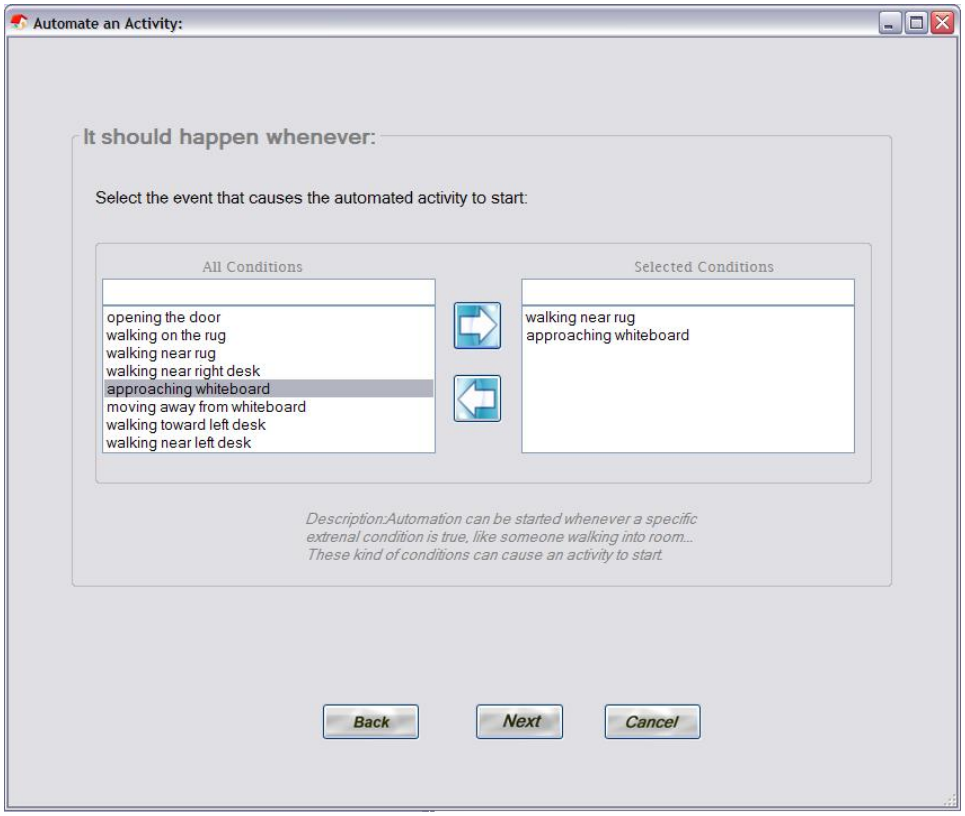

Figure 34 modify an automated activity, step 3.

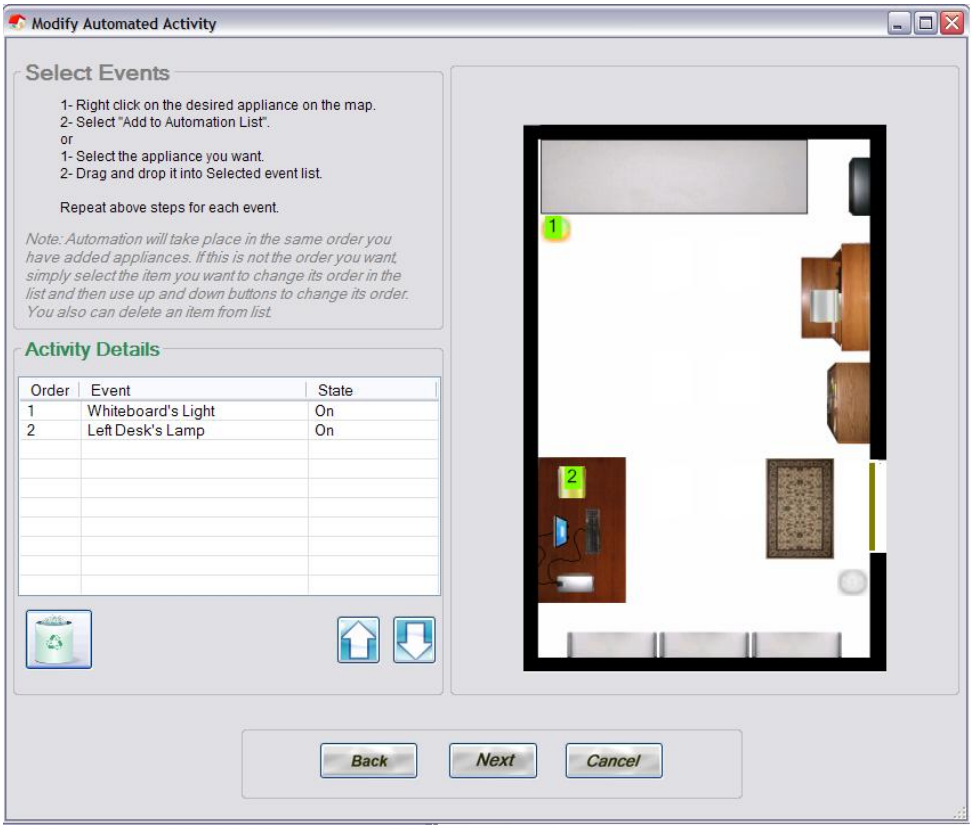

Figure 35 modify an automated activity, step 4.

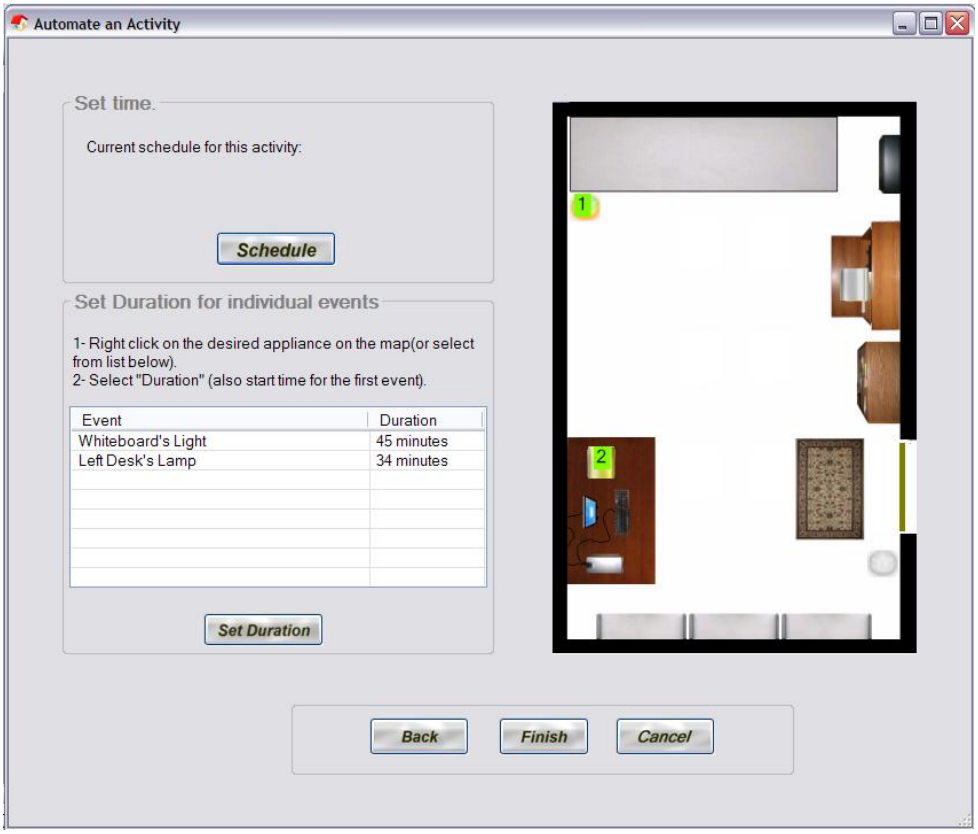

Figure 36 modify an automated activity, step 5.

The modification procedure uses a direct manipulation approach; users can click on an item in the map and change its attributes using either context menus or drag and drop (

Figure 37). For example, to define a new activity that includes "turn on the desk's lamp, then turn on the whiteboard's light", the user can 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 definition panel).

Users also can remove events or change the temporal order, schedule, duration, and trigger conditions for each event in the activity.

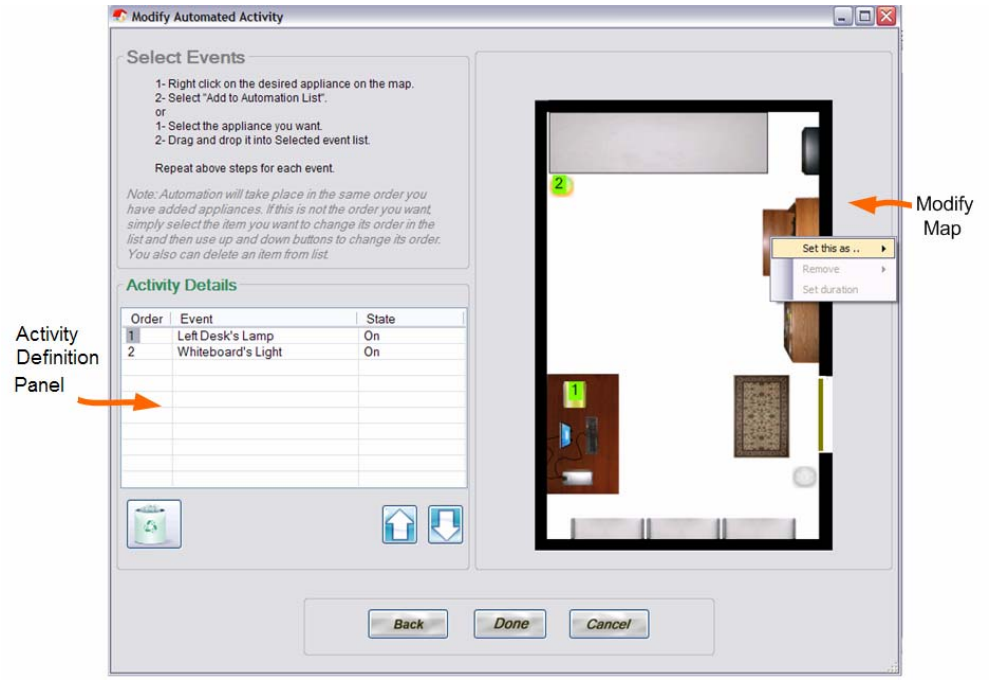

Figure 37. Modification dialog, changing events.

As we mentioned in the above steps, CASA-U additionally offers a suggested automation policy which allows users to skip all the steps, leaving it to CASAS to detect new frequent patterns or changes in previous activities. We referred to this option as explicit request and smart detection in previous sections.

Another interesting feature of CASAS, as mentioned before, is the ability to receive guidance and advice from the resident and adapt to changes in resident

lifestyle. Resident feedback includes explicit requests in the user interface or implicit feedbacks. To give the resident the opportunity to guide CASAS, we provide a rating system in which the resident can indicate if they like an automation activity or not. This corresponds to our second approach of dynamic adaptation as explicit guidance. The learning algorithm updates its model, based on the guidance provided by user as the ratings of desired activity which is be based on a scale of 1..5 (see Figure 38).

Figure 38 Rating.

CASA-U also provides an option in the main window that can be used to inform CASAS that a particular activity should be monitored and its potential changes should be detected (which we referred to as an explicit request in the dynamic adaptation section). The reason for considering such this option is that it would be more convenient for users if they could declare an activity to be monitored directly through the interface, rather than going through the "modify" option and waiting to be asked by CASAS if they want the current activity to be monitored or not. We also decided not to show the detection results at the moment they are detected, as it can be disturbing for user. Rather we show a small tool-tip and inform the user that the results are ready and s/he he can view the results by

choosing the "Edit" menu and then the "Show Detected Changes" sub-menu. After selecting this option, the user will be offered the list of detected changes and can choose some or all of the changes to be applied to the monitored activity (see Figure 39).

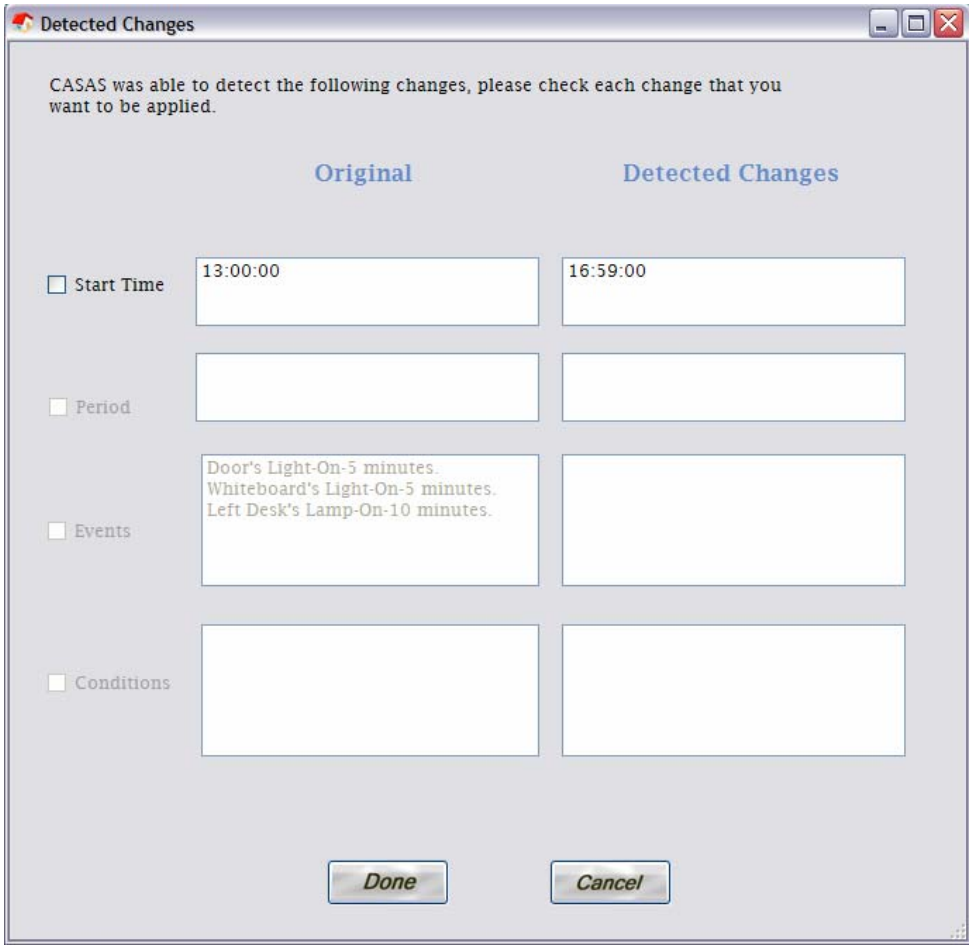

Figure 39 detected changes.

As we also mentioned in the dynamic adaptation section, the user also will be offered a list of similar activities (with the same structure and other characteristics except for the time or the day of week attribute). S/he can select among those activities and apply the detected changes to those similar activities too or reject the changes (see Figure 40).

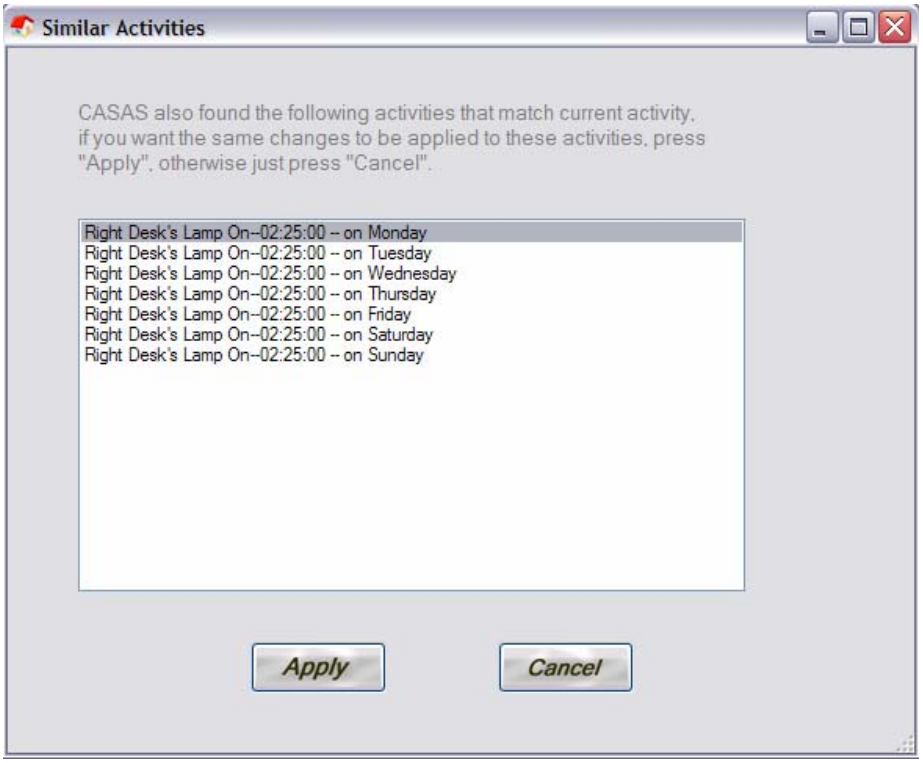

Figure 40 Similar activities detected by CASAS.

CHAPTER THREE

DISCUSSION AND EXPERIMENT FINDINGS

"I find life truer, more desirable and mysterious every year, the idea that life can be an experiment of the seeker for knowledge, and not a duty, not a calamity, not trickery."

Friedrich Nietzsche

In the following sections, we will summarize the results of our experimentation and implementation of FPAM, HAM, AAM and CASA-U, all as part of an integrated model, CASAS. In addition to implementing core modules, we implemented a few utility tools for generating synthetic data and also for viewing HAM model. All The modules have been implemented in C#, using Microsoft ® .net framework, resulting in 133 different core classes and a total of 309 classes, including help generating classes (see Figure 41).

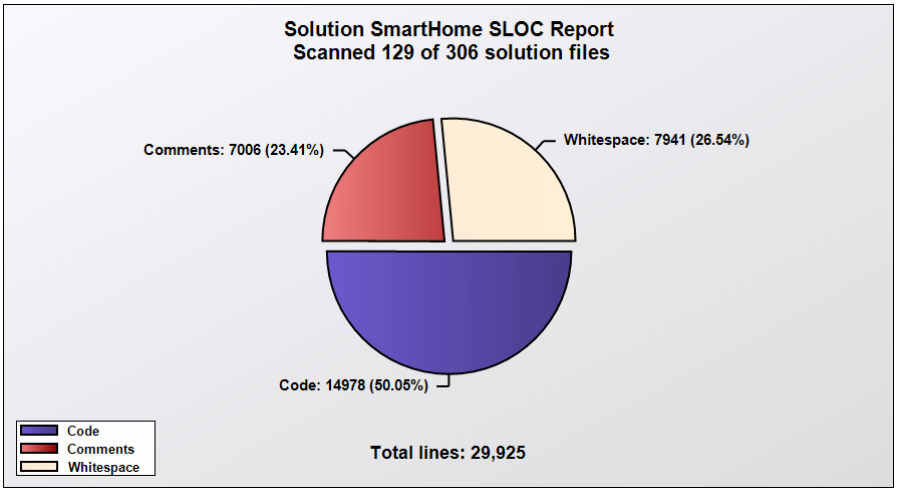

Figure 41 Number of classes in CASAS solution.

Synthetic Activity Generator (SAG)

Testing CASAS in a real world setting requires a great amount of time for gathering data and experimenting. In our work, in order to speed up the testing process, we decided to write a tool for generating activities based on given scenarios. The Synthetic Activity Generator (SAG) is a tool for generating synthetic activities based on information provided through a description file. SAG has the ability to generate activities based on given scenarios. In the case of periodic activities it generates activities with given periods and tries to fill in the gaps with other random activities. If there is a collision with the period of another activity at a certain moment, it resolves the conflict by choosing one of the activities randomly. For generating random activities and in order to avoid generating unwanted frequent or periodic patterns by random generator (the case seen in most ordinary random generators), a cryptography random generator was used which comparably generates more random results than ordinary random generator.

SAG also is able to consider other information such as duration values and their distribution (as absolute, uniform distribution or normal distribution), order of event in an activity, start time of activity and length of generated data. To generate desired activities, first a descriptor file should be provided that describes present devices, their states and desired type of period and timings for activities. The structure of a simple descriptor file is depicted in Figure 42.

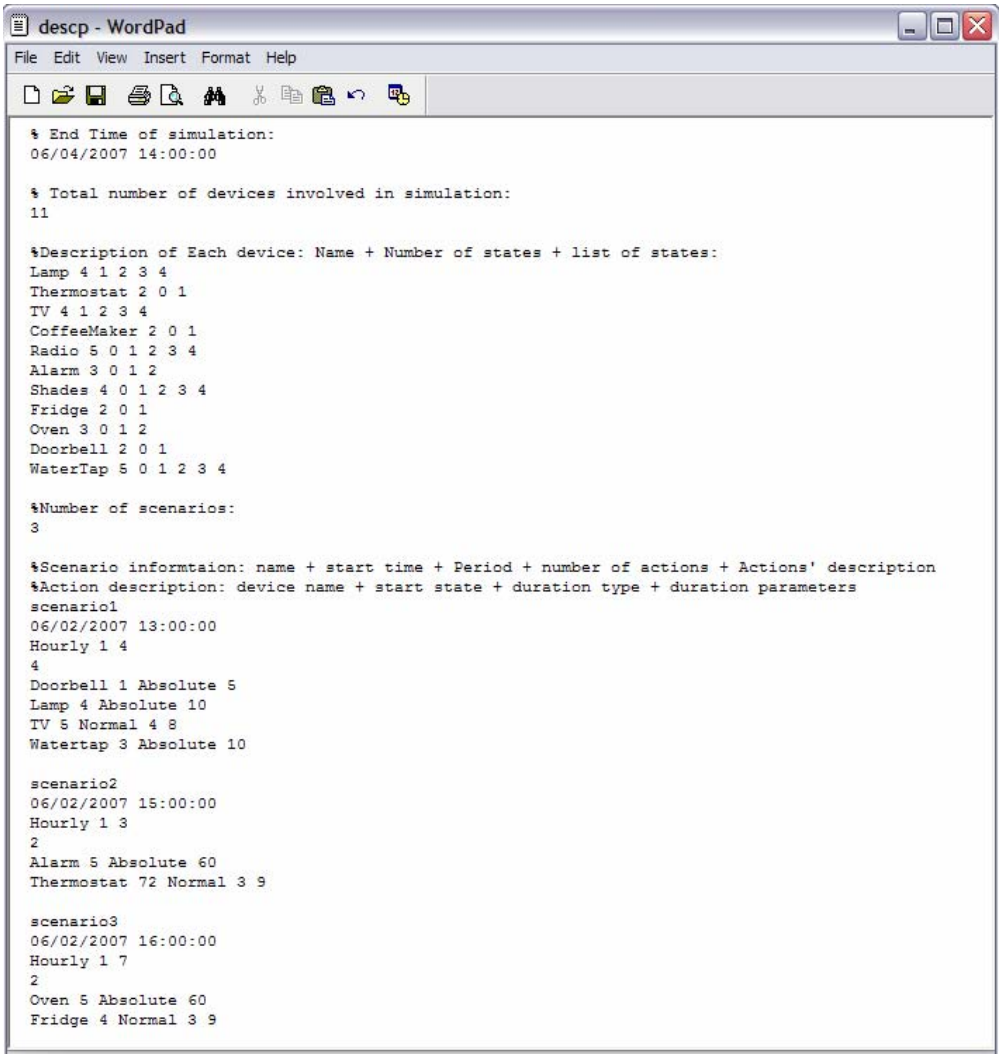

Figure 42 Description file.

In this example, scenario 1 fires every 4 hours, scenario 2 fires every 3 hours, and scenario 3 fires every 7 hours. Each scenario then specifies the number of associated events and provides the events descriptions. An event is described by

the name of the device it affects and the state to which it changes the device, followed by duration information.

Duration values can be specified as Absolute, Uniform, or Normal. Absolute t means that the duration of this action lasts for t minutes. Uniform means that duration of this action will be of a uniform probability distribution (between t_1 and t_2) and normal means that duration of this action will be of a normal probability distribution (between t_1 and t_2). In this example, scenario "scenario1" has 4 events. The first event indicates doorbell ringing which is followed by lamp being turned on which lasts for 10 minutes, and so on. Figure 43 shows a small part of a generated activity file for this descriptor:

It is worth mentioning that in addition to acting as a stand alone application, SAG was integrated into the CASAS model to generate a virtual event stream that is dispatched to the core components at regular time intervals (every 30 seconds) to simulate the effects of a real resident interacting with devices. These interactions are visualized by CASA-U accordingly and will be mined on a regular basis.

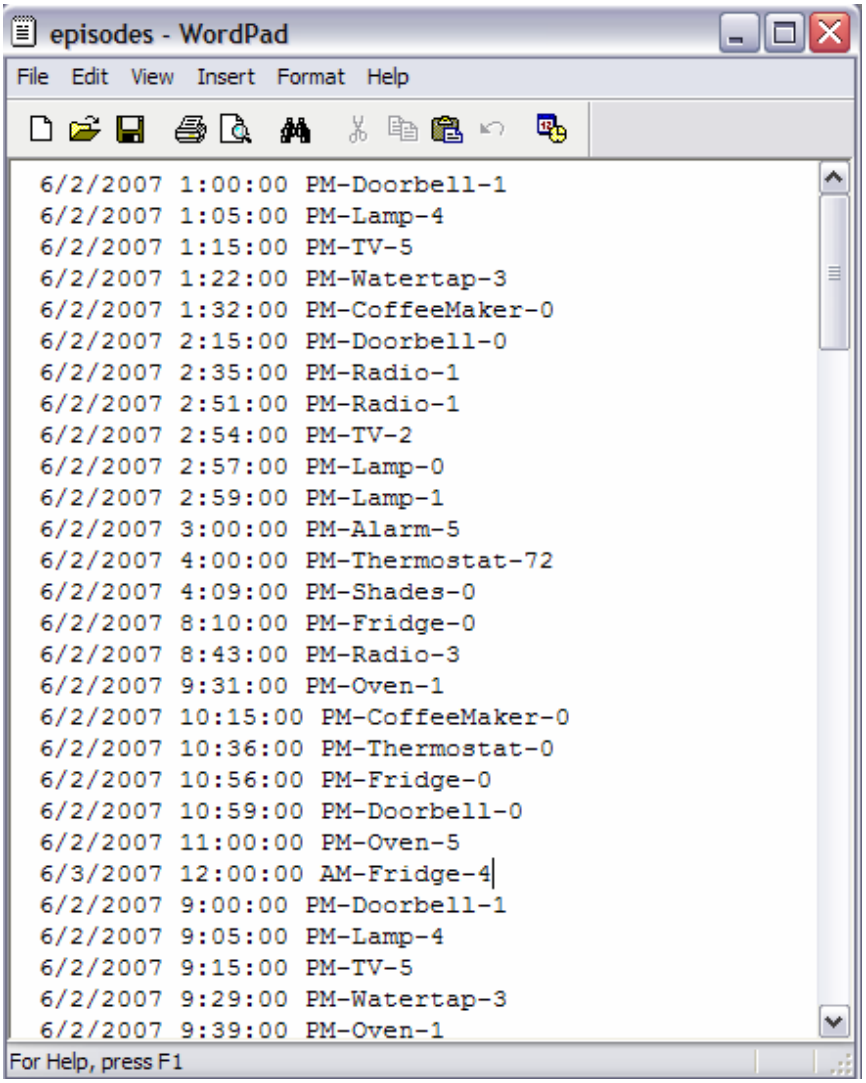

Figure 43 Generated activity file.

HAM Viewer Tool

In order to get a visual view of the HAM dynamic structure that could help smart home researchers test different scenarios efficiently, a user interface was designed to provide a tree view on HAM and a list view on periodic activities. The tree

view structure complies with the HAM structure completely by providing "day", "time", "activity" and "event" levels. At the activity level, different information such as transitional probabilities, triggers, etc., is shown about each activity. Users can construct a model by specifying a description file as described in previous section and also set various parameters such as compression threshold, period confidence values, etc. After the model is constructed, users can perform various operations such as reinforcing an activity or predicting events. Figure 44 shows a snapshot of the HAM Viewer.

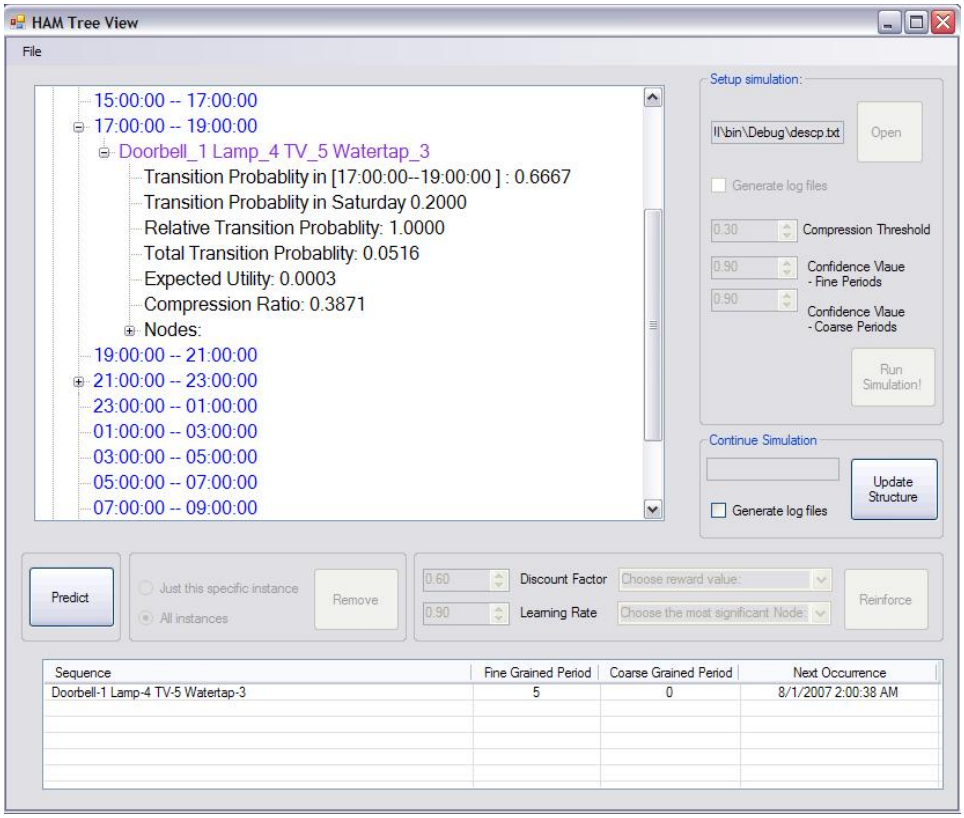

Figure 44 HAM Viewer.

Another capability of the HAM Viewer is providing a visual graph structure of the HAM model. It was developed using the GLEE engine (a .NET tool for graph layout and viewing). This graph can be viewed via choosing the "View" menu and then selecting "Graph View". It also offers zooming and panning capabilities. A "zoomed in" part of this graph is depicted in Figure 45.

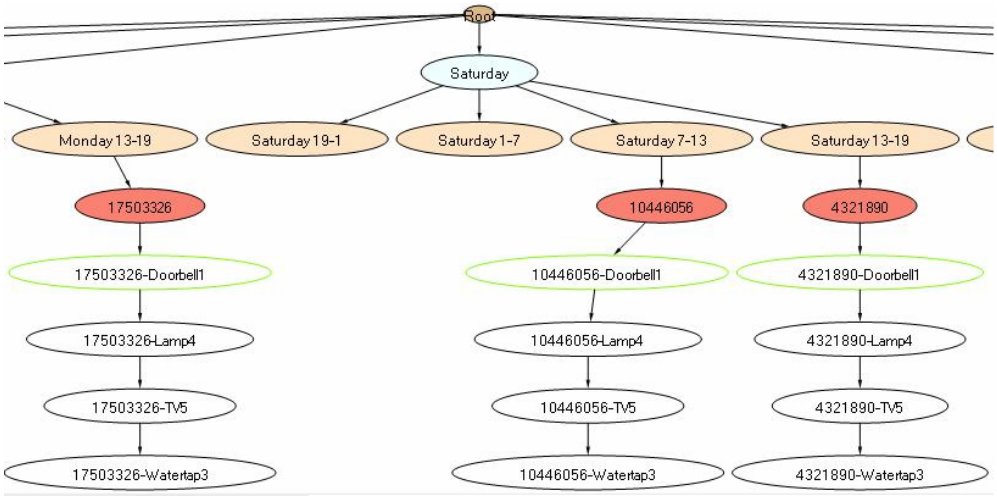

Figure 45 Graph structure of HAM.

Experiment Results

In this section, we present experimental results that validate the performance of FPAM, HAM and CASA-U using both real world and synthetically generated data.

FPAM

To evaluate the ability of FPAM to find frequent and periodic patterns, first it was tested on one month of synthetic data generated by SAG, containing 6 main scenarios with different settings such as different length sequences, triggers at different positions, different periods and events. Five scenarios included sequences of length 2 and the other one was a sequence of length 3. Two of the scenarios also included triggers, one with an "opening the door" trigger at the end and another one with an "opening the door" trigger in the middle (as its second event). The periods for these scenarios ranged from 2 to 5 hours. The devices and sensors considered in this test scenario were the same as are installed in the AI lab at the Washington State University School of EECS. As we described in SAG descriptions, in addition to main scenarios provided by user, the gaps between provided scenarios will be filled by random activities. Depending on the number of devices and scenarios, there is a chance that provided scenarios will be accidentally generated as random activities. In addition, there might be some activities that overlap due to period overlap, where SAG will decide to randomly

choose one; this might reduce the expected number of instances of a specific scenario. The following list shows all tested scenarios:

a. scenario1: start on 09/02/2007 13:00:00, Hourly 2, 2 events:
    a. Door's Light- 1 Absolute- 5
    b. Left Desk's Lamp- 1 -Absolute- 5
b. scenario2: start 09/02/2007 13:30:00, Hourly 3, 2 events
    a. Whiteboard's Light- 1 -Absolute- 5
    b. opening the door- 1 -Absolute- 5
c. scenario3, start on 09/02/2007 13:45:00, Hourly 5, 2 events:
    a. Right Desk's Lamp- 1 -Absolute- 3
    b. Whiteboard's Light- 1 -Absolute- 3
d. scenario4, start on 09/02/2007 14:15:00, Hourly 2, 3 events:
    a. Left Desk's Lamp- 1 -Absolute- 5
    b. opening the door- 0 -Absolute- 5
    c. Right Desk's Lamp- 1 -Absolute- 5

e. scenario5, start on 09/02/2007 14:35:00, Hourly 3, 2 events:

    a. Left Desk's Lamp- 1 -Absolute- 5

    b. Whiteboard's Light- 1 -Absolute- 5

f. scenario6, start on 09/02/2007 14:55:00, Hourly 3, 2 events:

    a. Right Desk's Lamp- 1 -Absolute- 5

    b. Door's Light- 1 -Absolute- 5

FPAM was able to find periodic and frequent activities accordingly, such that all activities were detected with correct periods. It also correctly handled activities that included triggers such that the sequence of length 3 with a trigger in the middle was broken down into two corresponding activities of length 1 and the sequence of length 2 with a trigger at the end was reduced to an activity of length 1. However, due to the period collision avoidance employed by SAG, some activities were not present as frequent as expected in the generated synthetic data, which resulted in lower compression rates.

In addition, to test the capability of FPAM to find variable-length sequences, it was tested on one month's worth of synthetic data that included the following long sequence of length 10 with a period of 2 hours:

Door's Light- 1, Left Desk's Lamp- 1, Left Desk's Lamp- 0, Left Desk's Lamp- 1, Left Desk's Lamp- 0, Left Desk's Lamp- 1, Left Desk's Lamp- 0, Left Desk's Lamp- 1, Left Desk's Lamp- 0, Left Desk's Lamp- 1

FPAM was again able to find the given sequence (see Figure 46) with a given period of two hours. It was interesting to note that it also discovered another frequent pattern of length 11 with a period of 28 hours along with the original given pattern. Because in the current setting there are only a limited number of devices and also because the random generators are not perfect, there is always a chance that an undeclared frequent pattern will be formed, as is the case in current problem. We were able to trace this accidentally-generated sequence in the generated episode files as a frequent pattern of length 11 which provides additional evidence for the ability of FPAM to find variable-length sequences.

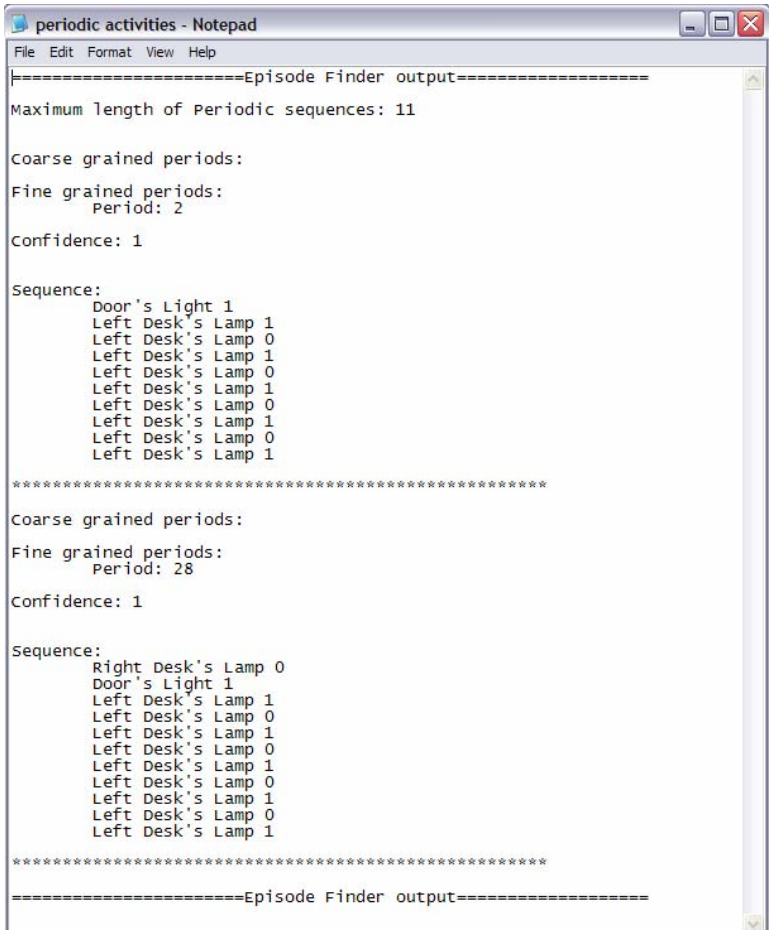

Figure 46 FPAM results.

In addition to the above tests applied to synthetic data, we evaluated FPAM on real-world collected data. To evaluate FPAM on real world data, we tested it on the data obtained through sensors located in alpha room of the AI lab in WSU's School of Electrical Engineering and Computer Science. The sensors layout is exactly the same as the user interface's layout, as can be seen in Figure 47 where numbers on the map correspond to motion (light) sensor numbers.

Figure 47 Sensor layout.

We prepared a simple script that included a series of activities to interact with sensors and then based on the script, a participant interacted with sensors and lights for about an hour, repeating the script for 10 times. In order to inject some randomness into data, participant was asked to do something random in step (3) for about a minute. The activity in this script was defined as follows:

1. Turn on right desk's light for 2 minutes

2. Turn off right desk's light.

3. Do something random for 1 minute.

It should be noted that due to the small number of devices and sensors in the room, it is not possible to act quite randomly and therefore some patterns will still be repeated, forming a frequent activity in addition to above mentioned predefined patterns. Besides between turning on and off the light, the motion sensors detected slight motions (such as hand movement) which resulted in a randomly-triggered pattern occurring between step 1 and 2 in above script. As we described before, if a trigger is detected in the middle of a sequence (the motion patterns for example), our algorithm will split apart the detected sequences into two parts to separate trigger. Another issue is that when lights are turned on, it's possible for the light sensor to be triggered several times very quickly (like 0.1 seconds for a single turn on).

Despite above issues, our algorithm was able to find the following frequent patterns properly:

- ✓ Right Desk's Lamp ON, Triggers: walking in the middle of the room, compression: 12
- ✓ Right Desk's Lamp OFF, no triggers, compression: 9
- ✓ Left Desk's Lamp ON, no triggers, compression: 2

- ✓ Right Desk's Lamp ON - Right Desk's Lamp OFF, no triggers, compression: 10
- ✓ Whiteboard's Light 1, no triggers, compression: 2

As can be seen from above obtained results, our desired patterns have been discovered. The first pattern has been obtained by splitting a sequence of "Right Desk's Lamp ON – x - Right Desk's Lamp OFF" where "x" shows a random motion pattern as described before. It also shows that the pattern has a trigger of "walking in the middle of the room", which again is a correct finding because after turning the light off, the participant performs a random action and then heads back to the right desk to turn on the light; however, to reach the right desk again, the participant has to cross the room.

The second sequence is again obtained by splitting the desired sequence in the script to extract the trigger in the middle. Note that compression values have been multiplied by 100 to make it easier to compare different compression values. The difference in compression values between the two sequences "Right Desk's Lamp ON" and "Right Desk's Lamp OFF" is due to multiple ON-OFF triggers of switches for one a single ON-OFF action (when lights are turned on or off, it is possible for the light sensor to be triggered several times very quickly) and it seems that random multiple triggers happen more in the case of "Right Desk's Lamp ON".

The third sequence is a result of a random activity (step 3) and it can be seen that its compression value is relatively small compared to the main script activities (steps 1 and 2). The fourth found sequence is again a result of multiple switched and the last found sequence is a frequent activity generated by a random step (step 3) where again its compression value is relatively small compared to the values for the main script activities (steps 1 and 2). The above results show that FPAM is able to find frequent patterns in real world data as well. However, some precautions should be taken to reduce interference of sensor data such as in the above "Right Desk's Lamp ON - xxx- Right Desk's Lamp OFF" pattern. Also, some real world situations such as multiple ON-OFF triggers of switches for a single ON-OFF action should be discriminated from the actual desired patterns.

AAM

In the next step, we evaluated and tested the AAM model (and as a result the HAM model as well) to see how well it can detect changes in a given activity. AAM was first run on the synthetic data for one month which contained 6 main scenarios. One of the scenarios (the following) was chosen to be the activity for which we will detect changes:

3. scenario3, Hourly 5, 2 events,
    a. Right Desk's Lamp- 1 -Absolute- 5
    b. Whiteboard's Light- 1 -Absolute- 5

After generating and mining data for these 6 scenarios, the HAM and CASA-U models were generated. We identified the above activity through the CASA-U interface. Note that the above scenario will be reduced to a one event sequence as the second event is a trigger which basically has no effect on the activity and therefore is eliminated. Therefore, the activity we were looking for in the interface was "Right Desk's Lamp - On". We then used the "monitor" option to tell CASAS to monitor and detect changes for this activity. In order to facilitate the process of finding changes, we generated another one month of data using SAG (the synthetic data generator). We modified the description of the scenarios in the first dataset, replacing durations for all to 7 or 10 minutes (instead of 3 or 5 minutes). This change can be seen in the following description of our desired activity:

4. scenario3, Hourly 5, 2 events,

    c. Right Desk's Lamp- 1 -Absolute- 7

    d. Whiteboard's Light- 1 -Absolute- 7

To accelerate the testing process, the AAM regular schedule was changed to be started in a few seconds (otherwise, with a normal setting we have to wait a whole week!). AAM was able to find the changes accordingly, as can be seen in Figure 48. According to the results, we can see that AAM has detected the changes

accordingly by finding a duration of 6.44 minutes which is quite close to the actual changes of 7.0 minutes. The reason that AAM did not detect a change of precisely 7.0 minutes is due to the fact that there are not many devices in the current description (12 devices, including motion sensors and lights). In addition, despite the fact that SAG tries to generate random episodes, it does not use a perfect random generator. As a result, SAG might generate the same episode accidentally but with a different duration, which in turn causes a deviation from the designated duration.

We can also see that in addition to changes in duration, AAM has detected some changes in start time. This is another correct finding by AAM, as in the second dataset, we changed the duration of all events in all scenarios which resulted in a shifted start time for all scenarios, in our case 14:55 instead of original 14:25. We can also see another start time, 8:55. This later start time results from another activity that had similar structure in our description. Users can easily see the results and apply the changes they want. For example, they are able to select a new duration and discard the new start times. The changes are recorded for every activity in order to keep a history of changes.

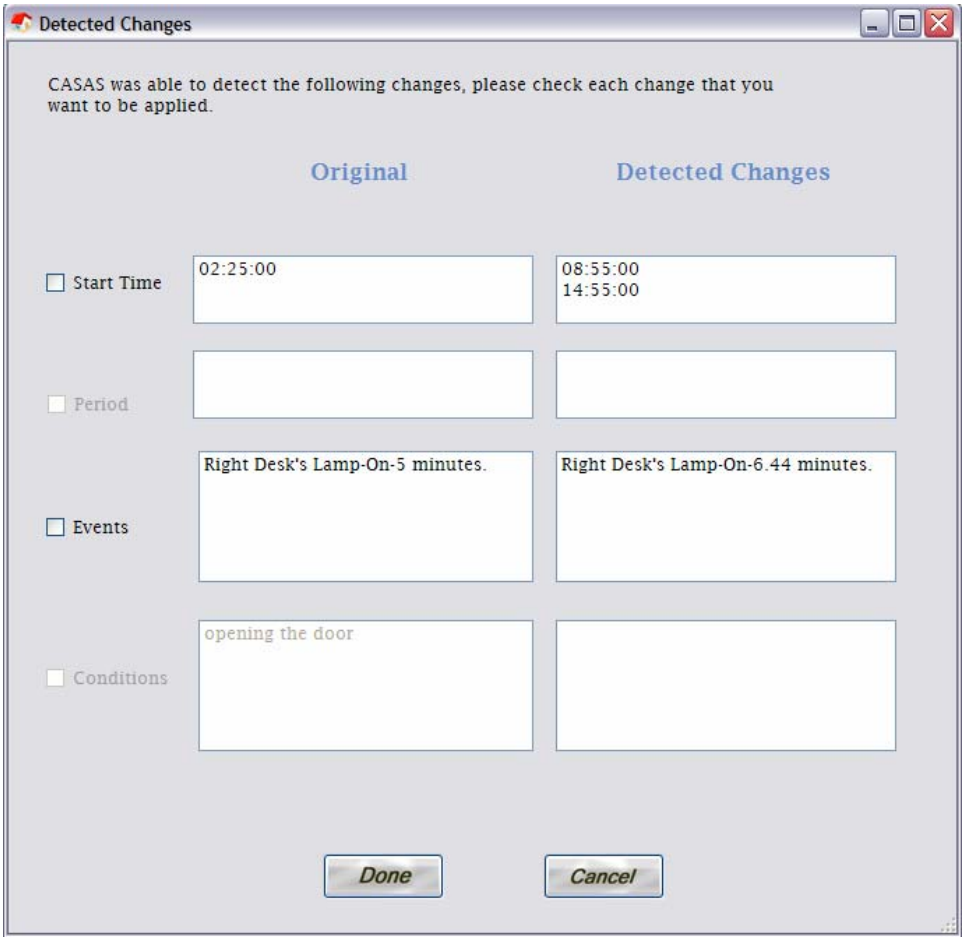

Figure 48 AAM detection results.

After changes were applied to the original activity, CASAS also offered a number of activities which again are similar but non-identical versions of our desired activity (see Figure 49).

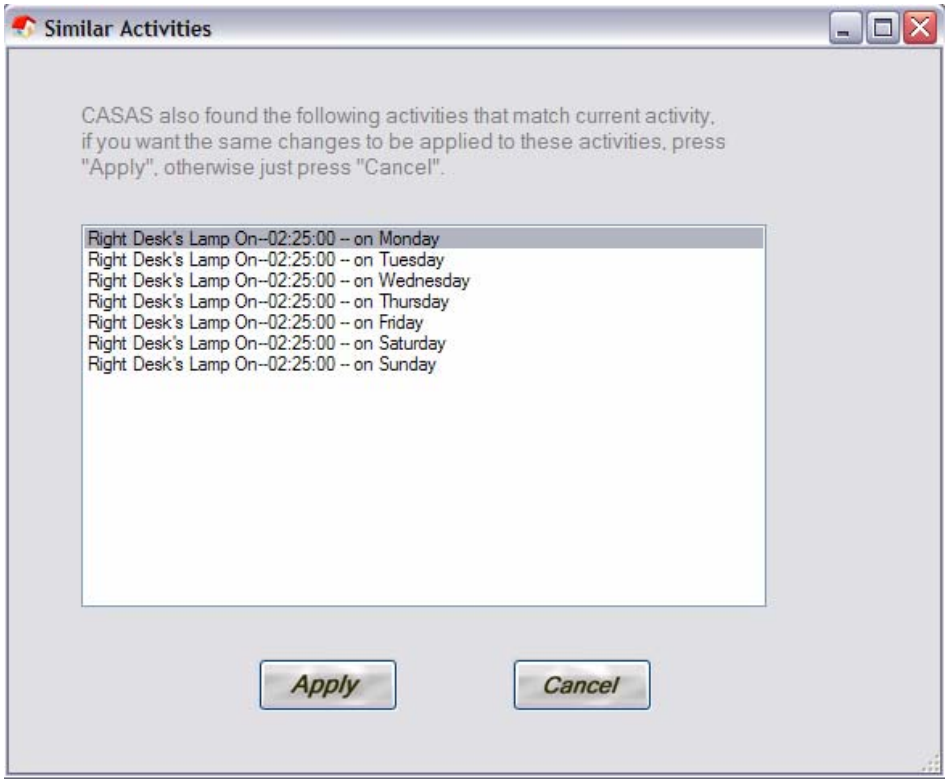

Figure 49 Similar Activities.

Just like in case of FPAM, we decided to test AAM on real world data by testing it on data gathered from sensors located in the alpha room of the AI lab. This time, we prepared two different scripts, as follows:

1. Turn on the right desk's light for 2 minutes
2. Turn off the right desk's light.
3. Perform some random actions for 1 minute.

And the second one as:

1. Turn on the right desk's light for 1 minutes
2. Turn off the right desk's light.
3. Perform some random actions for 1 minute.

We can see they are almost the same, except that in the second script, the durations are different for "Turn on the right desk's light". The above scripts were repeated for about 2 hours by interacting with sensors and lights in the alpha room while the data was collected by a database. As we said in the FPAM section, due to the motion detector sensing movement between turning on and off the right light, some random motion patterns would be introduced. To make it easier for duration calculation and to count the whole activity (turn on, then off) as a single unit, we decided to eliminate those random motion patterns between on-off pattern as they have no effect on decision making.

After generating and mining data for the first script, the HAM and CASA-U models were generated. We identified the "Right Desk's Lamp ON - Right Desk's Lamp OFF" activity through the CASA-U interface. The duration estimated by CASAS for this activity was 1.66 minutes (less than 2 minutes due to some accidental similar patterns and also due to the switch problem that we described before). We then used the "monitor" option to tell CASAS to monitor and detect changes for this activity. To find the changes, AAM used a second dataset based on the second script. AAM was able to find the changes, as can be seen in Figure

50. We can see that the duration change has been detected successfully, as well as a new trigger ("walking near the left desk"). By closely examining the data, we found out that "walking near the left desk" is a frequent event happening before turning on the light (reaching from the left desk to the right desk), therefore it has been detected as a frequent pattern. This experiment shows how AAM can successfully work on real world data.

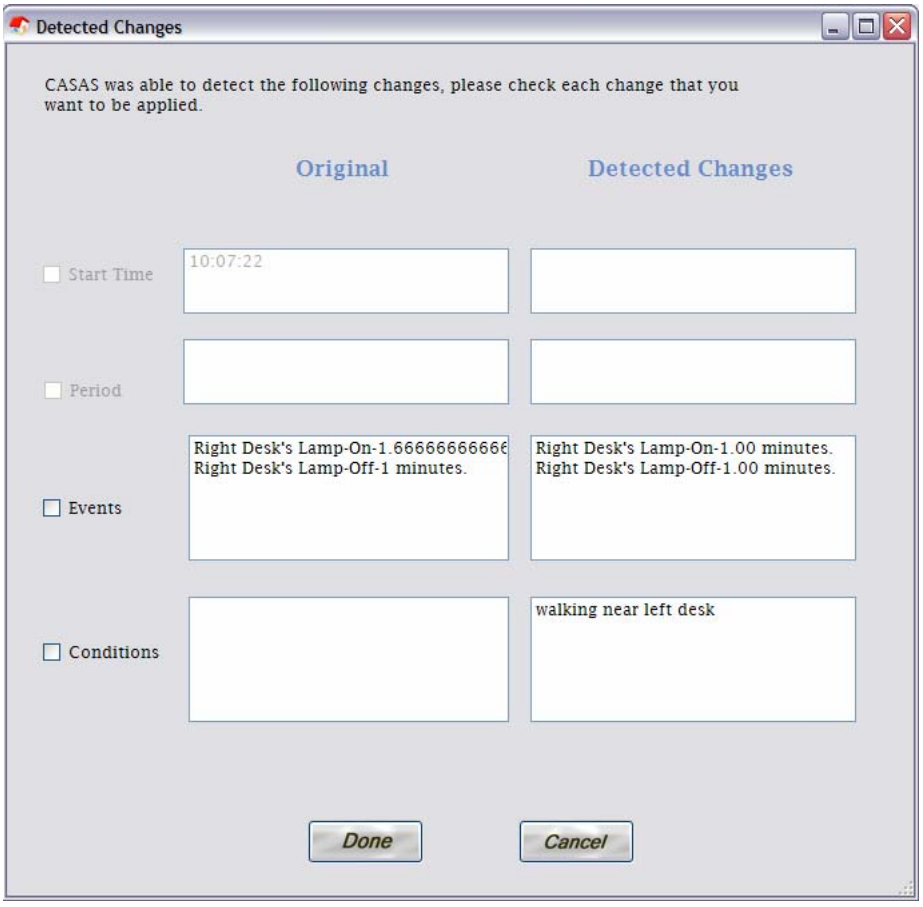

Figure 50 Detected changes.

CASA-U Preliminary Test

To evaluate the usability of design, a paper-based prototype and a high fidelity computer-based prototype were tested in three different sessions with a total of 10 participants who had different levels of computer skill (on a scale of 1-10). The paper-based prototype was tested using a "wizard of oz" [88] method, with each participant working through three tasks. We found several key results based on this study:

Mapping: In our early design, the temporal relationships between dynamic objects were demonstrated using connectors to emphasize sequential relationships; however the additional information, and in some cases the cluttered visualization, confused users. In addition, some participants were not sure about the meaning of relationships between two events. This problem was attributed to the fact that the representation of temporal relationships between events was not completely a natural mapping and did not provide the users with instant clues. Therefore, in our revised design, the connectors were eliminated and instead labeled events were used to emphasize order and temporal relation in a clearer manner. Another possible approach would have been to represent the events along a timeline and group related events together as episodes; however, such an approach would not be able to show spatial relations.

Tree view history panel: In the early design model, users could only find desired activities by using the forward/backward button or through a search option.

However, several participants claimed that they expected a tree view panel of activity history to provide an overall picture of the system's operations. This option has been added to the current implementation and is able to show a history of up to three months.

Dual view instead of a single view: Another change to the initial interface was to construct a dual map view instead of a single map to clarify which events actually occurred in the environment and which were scheduled for automation. To achieve this, we divided the screen into two different but symmetric maps of the home. One of the dual maps represents what is currently happening in the home based on sensor data (activity view), while the other map provides information about the next scheduled automated activity (automation view). We also decided to show only a single event on the activity map in order to reduce user confusion. In contrast, data from sensors in the initial design were visualized as a fixed length sequence of events.

Conceptual Model: In our user study, most users indicated on the questionnaire that they were not able to understand the underlying model very well. Because smart home design and application is currently a research topic rather than a well-known established technology, most users do not have a clear idea of the abilities and potential uses of such a tool. In addition, the lack of established paradigms for smart home interfaces and many complicated related issues such as

simultaneous representation of spatial-temporal constraints can be a source of confusion for users. However, some improvements can be made by performing more comprehensive empirical studies of potential usages of such a technology at the data gathering phase including involving more participants, trying out multiple alternative designs, testing for each alternative design, and providing different interfaces for different groups of users (power users might desire a different interface from elder adults in a nursing home).

Enabling/Disabling automation: We also found that residents prefer to be able to disable or enable automation in certain parts of the home, based on location or time. This will be a useful control feature in smart home interfaces; however we do not include it in our current implementation of CASA-U.

CASA-U Usability Study

We also performed a usability study in order to test the final version of CASA-U. In the study, first participants completed a background questionnaire and then they explored the CASA-U interface in order to get a feel for the software's functionality. None of the participants chosen for this study had seen CASA-U before or taken part in the preliminary study. After completing the core tasks of the usability study, exit questionnaires were completed by participants to elicit data and give insight into the extent to which CASA-U supported what the participants wanted to accomplish.

We recruited 5 participants for this usability study with different levels of computer interaction skill; the familiarity and skill of participants in working with common computer applications ranged from 1 to 6 on a scale of 6 (two of them with a skill level of 1-2 and the other three with a level of 4-6). One of the participants had no familiarity with smart homes while the other 4 participants had from basic to advanced knowledge about smart homes. We ran the study in the visualization and end user programming lab (VEUPL), located in EECS department of WSU. The videos were recorded and then analyzed using Morae® software. We allotted 50 minutes for each study session. At the beginning of the session, participants filled out a paper-based background questionnaire, asking them about their familiarity with computer applications (on a scale of 1-6) and also their familiarity with smart home concepts. Next, they completed a warm-up exercise, in which they were given a brief description of the CASA-U software, the rationale behind CASA-U and then asked to think aloud while they explored the CASA-U interface for 5 minutes.

After the warm-up exercise, participants were asked to complete a series of five tasks in which they had to use CASA-U software to complete the study tasks. These tasks included:

- ✓ Rate a particular activity: user should rate a current automated activity.

- ✓ Monitor a particular activity: user should select the monitor option for a current automated activity.

- ✓ View past automations: user should view all past automations in the automation history and then again navigate back to the beginning.

- ✓ Search for a particular activity: user should search for an automated activity that includes "Right Desk's Lamp".

- ✓ Adding a new automated activity: user should automate an activity with the following characteristics: it did not happen before, it should start at scheduled times every 1 hour, and it includes two events: "Whiteboard' light - On" with duration of 4 minutes and "Right Desk's Lamp - On" with a duration of 3 minutes

As they worked through these tasks, participants were instructed to think aloud by verbalizing their thoughts and actions. 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 finally the system's capabilities in a total of 40 questions based on a modified version of a standard QUIS test [95] adapted to our setting. The Questionnaire for User Interaction Satisfaction (QUIS) is a tool developed by a multi-disciplinary team of researchers in the

Human-Computer Interaction Lab (HCIL) at the University of Maryland at College Park [95]. The QUIS was designed to assess users' subjective satisfaction with specific aspects of the human-computer interface. The QUIS team successfully addressed the reliability and validity problems found in other satisfaction measures, creating a measure that is highly reliable across many types of interfaces. After analyzing the QUIS questionnaire, the software received the following QUIS ratings (on a scale of 1-9):

- ✓ Average Overall User Reactions : 5.28
- ✓ Average Screen & layout: 5.6
- ✓ Average Terminology and System Information: 6.48
- ✓ Average Learning: 5.86
- ✓ Average System Capabilities: 7.0

Figure 51 and Figure 52 show the overall and average results of the survey.

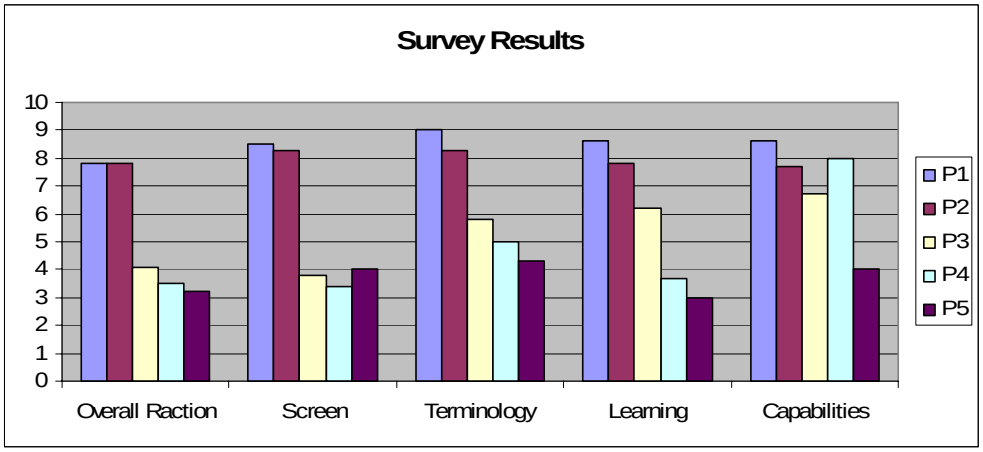

Figure 51 Survey Results.

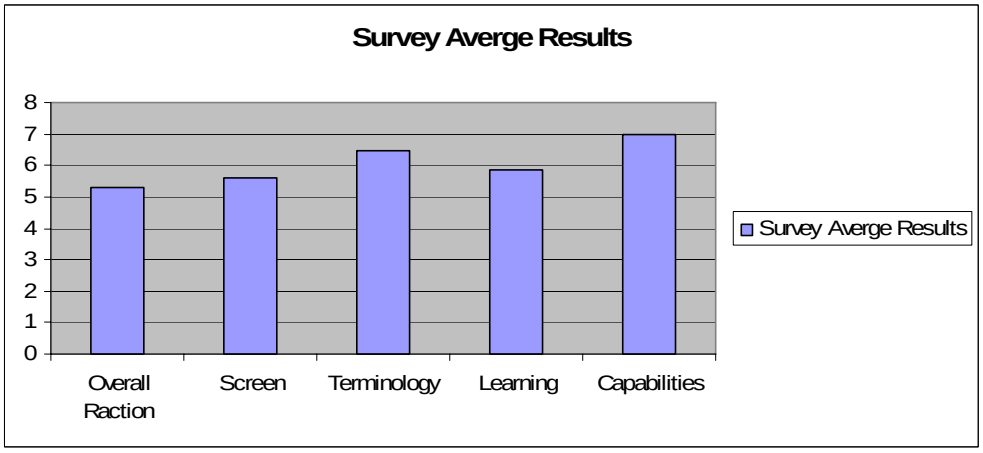

Figure 52 Average survey results.

In addition, to verify how well usability and user experience criteria were satisfied, we performed an analysis of tasks carried out by participants in terms of elapsed time to complete each step and also based on user ratings of the system. The following tables show the initial requirements and also summary empirical

findings based on the usability test sessions. The average time was 156 seconds for all 5 tasks. The expected times are a rough estimation based on number of dialogs that users had to go through.

Task	Expected Time	Time – P1	Time – P2	Time – P3	Time – P4	Time – P5	Time AVG
1	20	15	9	60	50	60	38.8
2	20	22	23	8	12	45	22.0
3	45	18	67	35	26	55	40.2
4	60	9	23	19	16	35	20.4
5	90	42	43	27	31	30	34.6
Total	235	106	165	149	135	225	156.0

Table 2 Task completion times in seconds.

In addition, the following figures show the distribution of time per each task for different participants and also the average time per task. Despite the fact that we expected task 5 to take the greatest amount of time, most people could finish it easily with the use of wizard-like dialogs that guided them through the process. Also, a few of them chose to select "detect new automation automatically" and did not go through the whole process which shows that considering "intelligent" options can make the process easier . Task 3 was a bit confusing for some participants as there were several ways to view the past history (tree view and forward/backward buttons) and they were not sure what the difference was between using tree view and forward/backward buttons. They also expected the automation history to show recently-added automation (which they defined),

while in our model its main purpose is to illustrate previous "executed" automations, not all available automations. This implies that we should provide a way for users to be able to perceive their defined automations, regardless of whether they have been executed or not.

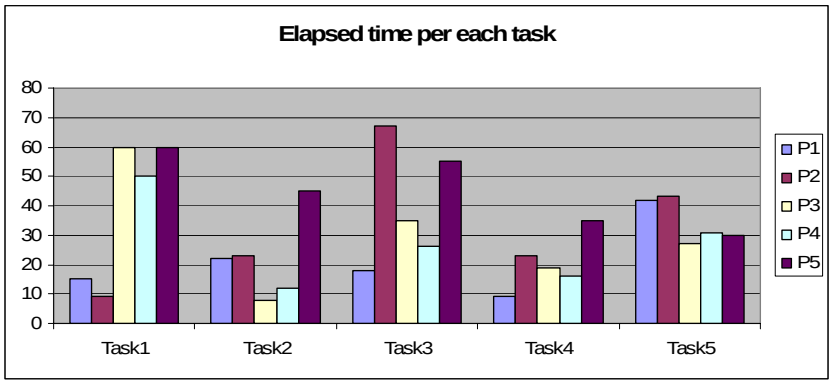

Figure 53 Completion times for each of the 5 tasks.

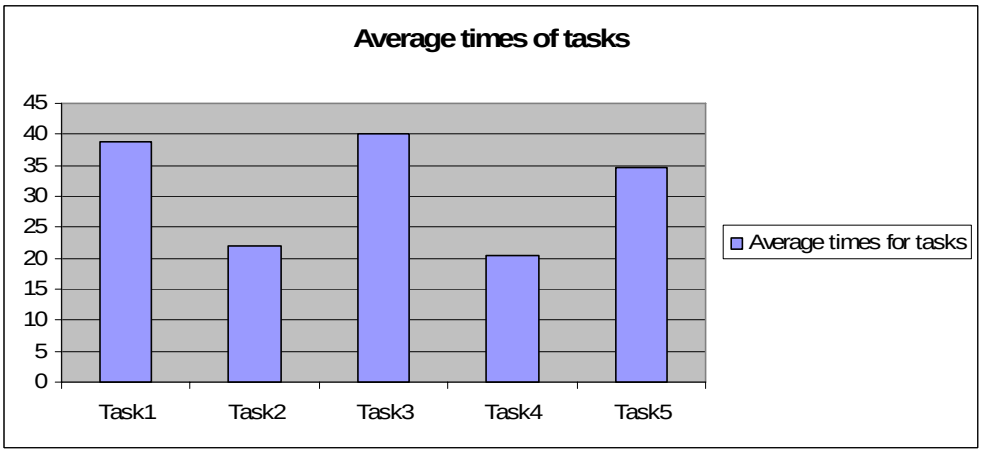

Figure 54 Average completion time.

Table 3 summarizes our findings where users' rating is 0.6 of our expected ratings.

Usability or U.E. Goal	**Relevant Empirical Result**	**Commentary**
Users should rate their overall reaction to system a 6 on 9.	5.28	NA
Users should rate the clarity of display a 7 or higher on a 9 point scale.	5.6	Distinguishing between different states of devices still was an issue for some participants.
Users should rate clarity of terminology an 8 or higher on a 9 point scale.	6.48	Some participants had problems with used terminology such as monitor, automate.
Users should rate the ease of use of this tool and its learning an 8 or higher on a 9 point scale.	5.86	Some participants claimed it was not easy to use as they didn't know what they should expect from a smart home interface.
Users should rate system's capabilities a 7 on a 9 point scale.	7.0	NA
User should be able to rate a desired activity within 20 seconds.	38.8	As it's in main window, most users had no problem returning quickly. However, some people despite the fact that they found it, they were trying to find out what aspects of activity they rated and they basically were trying to understand the concept of activity rating.
Users should be able to search and find a desired activity in less than 60 seconds.	20.4	It was an easy task for most of participants.
Users should be able to add a desired activity to "monitored" list in less than 20 seconds.	22	As it is in the main window, most users had no problem returning quickly.

Users should be able to add a desired activity within 90 seconds.	34.6	Some users preferred to use the "detect new activities automatically" option which resulted in less time than expected.

Table 3 Summary of usability test's findings.

We can see that 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 (consider above ratings). Some participants mentioned that the terminology is not very clear. For example, some did now know exactly what "monitoring an activity" means. Therefore, a revise in terminology is also necessary to use more common commands such as "Check activity for changes". Just like in the preliminary studies, despite the applied changes, some participants 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. Participants also expected to be able to interact with the map in the main page while we only allowed interaction via the "define new automation" or "modify automation" dialogs.

In a nutshell, the usability study revealed that due to conceptual model issues in smart homes such as spatial and temporal relations and high dimensional

parameters, more studies and revisions of software are needed in order to provide a user interface that is quite comprehensive and easy to use.

In addition to evaluating CASA-U, in a separate question, participants were asked whether they would be willing to use smart home technology 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. This also shows that we also need to perform usability tests regarding usefulness and acceptability of smart home technology.

Alternative Approaches

It is obvious that no method can claim to be the most perfect method for modeling a specific domain; therefore it is necessary to consider a number of alternative solutions at the design phase and choose the one that seems to be the most promising. For us, in the early stages of designing CASAS, we considered a number of alternative methods for modeling temporal relations, modeling activities and also for predicting automations.

For demonstrating temporal relations, we examined a number of alternative methods including: Allen's temporal logic [62], point algebra [63] and fuzzy representation [64]. Allen's temporal logic which is based on a set of thirteen atomic temporal relations between time intervals provides very good expressiveness power but it has been proven to be computationally intractable, the same can is true about fuzzy relations. Point algebra as tractable version of Allen's temporal logic uses time points rather than time intervals for comparison. We chose to adapt point algebra to a Markov decision process in order to model temporal relations in our model.

Another early design approach that was considered for modeling dependency of an activity and its various external factors was a neuro-fuzzy system that was also

able to handle vague instructions by resident. In a smart home setting, many external factors may be relevant to occurrence of a certain activity such as weather condition, resident's mood, etc, besides users might want to declare vague and fuzzy instructions such as "high temperature". In such this scenario, a fuzzy system can be used to represent vague values; at the same time, a neural network (associative neural network) can be used to learn patterns of activity. A combination of these two approaches will make a neuro fuzzy system. However, there were a number of issues:

1. Handling temporal relations between events indicates the necessity for recurrent elements in neural network; however the problem with traditional RNNs is that they have a very limited ability to look far back into the past due to insufficient, decaying error back flow. There have been a number of approaches to overcome this problem such as "Long Short-Term Memory" [67-68], but they are very complicated and require considerable computational effort. In addition, there a number of other issues, such as representing durations for a state and its start time as part of a recurrent network.

2. Representing background knowledge and general rules in neural networks is not very easy. For example, to learn the simple concept "whenever

someone enters room, turn on the light", a neural network needs to be presented with many examples.

3. Another issue with using a neural network is the explicit manipulation of activities by the resident. If the resident changes an activity pattern, how this change can be reflected in the neural network? It is possible to train the network on a number of virtual examples, but this can lead to incorrect solutions and changes in other activities.

Another early approach that was examined was the "transfer knowledge" approach [96]. Incorporating transfer knowledge into a smart home learning system is quite useful, because the resident can tell system in the form of simple IF_THEN rules what the smart home should do in similar situations. However, this might turn out to be a tedious job and users might not be willing to be so much involved with training a system. In addition, transfer methods provide mechanisms for transferring knowledge from a previously learned task into a new task, but this assumption does not always hold in a smart home setting, as sometimes the resident might wish the system to learn a totally new activity or one which is not similar to previous activities.

Conclusion and Future Work

In this thesis, 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 find frequent and periodic patterns of activity and also to adapt to the changes in these 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 extend the FPAM method to include discovery of parallel activities. In addition, we want 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. We may also consider other paradigms of pervasive computing such as transparent computing, and compare the results to traditional paradigms.

BIBLIOGRAPHY

1. Luhr. *Recognition of emergent human behavior in a smart home: A data mining approach.* Journal of Pervasive and Mobile Computing, 3, 2007.

2. R. Simpson, D. Schreckenghost, E.F. LoPresti, and N. Kirsch. *Plans and planning in smart homes.* In J. Augusto and C. Nugent (eds.), AI and Smart Homes, Springer Verlag, 2006.

3. S. Moncrieff. *Multi-model emotive computing in a smart house environment.* Journal of Pervasive and Mobile Computing, 3, 2007.

4. J.C. Augusto and C.D. Nugent, *Designing Smart Homes*, LNAI 4008, pp. 146 – 164, 2006. Springer-Verlag Berlin Heidelberg 2006.

5. Cook, Diane; Sajal Das (2004). *Smart Environments: Technology, Protocols and Applications.* Wiley-Interscience. ISBN 0-471-54448-5.

6. L. Hales. *Intelligent appliances are wave of the future*, January 22, 2006.

7. K. Swisher. *Intelligent appliances will soon invade homes*, 2006.

8. MIT. *Things that think*, 2006.

9. Philips. *Interactive tablecloth*, 2006.

10. J. A. Rode. *Appliances for whom? Considering place*. Personal and Ubiquitous Computing, 10(2-3):90, 94, 2005.

11. AIRE Group. *MIT Project AIRE About Us*, Jan 2004. http://www.ai.mit.edu/projects/aire.

12. 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.

13. M. Romn, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. *Gaia: A middleware infrastructure to enable active spaces.* IEEE Pervasive Computing, pages 74-83, 2002.

14. N. A. Streitz, J. Geiler, T. Holmer, S. Konomi, C. Mller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. *i-LAND: an interactive landscape for creativity and innovation.* In CHI, 1999.

15. G. D. Abowd and E. D. Mynatt. *Designing for the human experience in smart environments.* In D. J. Cook and S. K. Das, editors, Smart Environments: Technology, Protocols, and Applications, pages 153-174. Wiley, 2005.

16. A. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. *The gator tech smart house: A programmable pervasive space.* IEEE Computer, 38(3):50-60, 2005.

17. NIST. *Smart space.* NIST laboratory. http://www.nist.gov/smartspace

18. M. C. Mozer. *Lessons from an adaptive home.* In D. J. Cook and S. K. Das, editors, Smart Environments: Technology, Protocols, and Applications, pages 273-298. Wiley, 2005.

19. H. Duman, H. Hagras, and V. Callaghan. *Intelligent association selection of embedded agents in intelligent inhabited environments.* Journal of Pervasive and Mobile Computing, special issue on Design and Use of Smart Environments, 2007.

20. O. Brdiczka, J. Maisonnasse, and P. Reignier. *Automatic detection of interaction groups.* In Proceedings of the International Conference on Multimodal Interfaces, 2005.

21. S. W. Loke. *Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective.* The Knowledge Engineering Review, 19(3):213-233, 2005.

22. S. Luhr. *Recognition of emergent human behavior in a smart home: A data mining approach.* Journal of Pervasive and Mobile Computing, special issue on Design and Use of Smart Environments, 2007.

23. S. Moncrie. *Multi-modal emotive computing in a smart house environment.* Journal of Pervasive and Mobile Computing, special issue on Design and Use of Smart Environments, 2007.

24. M. Muehlenbrock, O. Brdiczka, D. Snowdon, and J. Meunier. *Learning to detect user activity and availability from a variety of sensor data.* In Proceedings of the IEEE International Conference on Pervasive Computing and Communications, 2004.

25. E. M. Tapia, S. S. Intille, and K. Larson. *Activity recognition in the home using simple and ubiquitous sensors.* In Proceedings of Pervasive, pages 158-175, 2004.

26. E. F. LoPresti, A. Mihailidis, and N. Kirsch. *Assistive technology for cognitive rehabilitation: State of the art.* Neuropsychological Rehabilitation, 14(1/2):539, 2004.

27. R. Simpson, D. Schreckenghost, E. F. LoPresti, and N. Kirsch. Plans and planning in smart homes. In J. Augusto and C. Nugent, editors, AI and smart homes. Springer Verlag, 2006.

28. R. L. de Mantaras and L. Saitta, editors. *The use of temporal reasoning and management of complex events in smart homes.* IOS Press, 2004.

29. F. Doctor, H. Hagras, and V. Callaghan. *A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments.* IEEE Transactions on Systems, Man, and Cybernetics, Part A, 35(1):55{65, 2005.

30. G. M. Youngblood and D. J. Cook. *Data mining for hierarchical model creation.* IEEE Transactions on Systems, Man, and Cybernetics, Part C, 2007.

31. The Computer for the 21st Century" - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991

32. G. Michael Youngblood, Edwin O. Heierman, Diane J. Cook, and Lawrence B. Holder. *Automated Hierachical POMDP Construction through Data-mining Techniques in the Intelligent Environment Domain.* In FLAIRS, 2005.

33. S. Lanspery, J. J. C. Jr, J. R. Miller, and J. Hyde. *Introduction: Staying put.* In S. Lanspery and J. Hyde, editors, Staying Put: Adapting the Places Instead of the People, pages 1-22. Baywood Publishing Company, 1997.

34. I. K. Zola. Living at home: *The convergence of aging and disability.* In S. Lanspery andJ. Hyde, editors, Staying Put: Adapting the Places Instead of the People, pages 25-40. Baywood Publishing Company, 1997.

35. AARP. Fixing to stay: *A national survey of housing and home modification issues*, 2000.

36. AARP. *These four walls... Americans 45+ talk about home and community*, 2003.

37. National Science Foundation. *IIS priorities for FY 2007*, 2006.

38. T. S. Barger, D. E. Brown, and M. Alwan. *Health status monitoring through analysis of behavioral patterns.* IEEE Transactions on Systems, Man, and Cybernetics, Part A, 35(1):22- 27, 2005.

39. J. Carter and M. Rosen. *Unobtrusive sensing of activities of daily living: A preliminary report.* In In Proceedings of the 1st Joint BMES/EMBS Conference, page 678, 1999.

40. H. Kautz, L. Arnstein, G. Borriello, O. Etzioni, and D. Fox. *An overview of the assisted cognition project.* In Proceedings of the AAAI Worskhop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care, pages 60-65, 2002.

41. A. Mihailidis, J. C. Barbenel, and G. Fernie. *The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate-to-severe dementia.* Neuropsychological Rehabilitation, 14(1/2):135-171, 2004.

42. M. Ogawa, R. Suzuki, S. Otake, T. Izutsu, T. Iwaya, and T. Togawa. *Long term remote behavioral monitoring of elderly by using sensors installed in ordering houses.* In Proceedings IEEE-EMBS special topic conference on microtechnologies in medicine and biology, pages 322-335, 2002.

43. M. E. Pollack. *Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment.* AI Magazine, 26(2):9-24, 2005.

44. BT. *Telecare Overview.* BT Exact, 2005. Website: www.btexact.com/research/researchprojects/currentresearch?doc=42834.

45. Intel Corporation. *Digital Home Technologies for Aging in Place*, 2005. Website: www.intel.com/research/exploratory/digitalhome.htm.

46. Siemens. *Smart Homes,* 2005. Website: www.siemens-industry.co.uk/main/business%20groups/et/smart%20homes.

47. Karen E. Peterson. *Home Sweet Ambient Home*, from Philips, 2002. Website: www.10meters.com/homelab1.html.

48. Augusto, J., Nugent, C.: *Smart homes can be smarter*. In Augusto, J., Nugent, C., eds.: Designing Smart Homes: The Role of Artificial Intelligence. Springer, Berlin (2006)

49. Chan, M., C. Hariton, P. Ringeard, E. Campo, *Smart House Automation System for the Elderly and the Disabled*, IEEE international conference on Systems, Man and Cybernetics, 1995, Vol. 2, pp. 1586-1589.

50. Mozer, M. C., The Neural Network House: *An Environment that adapts to its Residents*, Proceedings of the American Assocation for Artificial Intelligence, 1998, pp. 110-114.

51. Hagras, H., V. Callaghan, G. Clarke, M. Colley, Anthony Pounds Cornish, Arran Holmes, Hakan Duman, *Incremental Synchronous Learning for Embedded Agents Operating in Ubiquitous Computing Environments,* Soft Computing Agents- V. Loia (Eds), IOS Press, 2002, pp. 25-55.

52. Darnall, J. M, I. A. Essa, and M. H. Hayes ,*Exploiting Human Actions and Object Context for Recognition Tasks.* Proceedings of 7th IEEE international Conference on Computer Vision , 1999, Vol. 1, pp. 80-86.

53. Bobick, A. *Movement, Activity, and Action: The Role of Knowledge in the Observation of Model*, Royal Society Workshop on Knowledge based Vision in Man and Machine, 1997.

54. Tapia, E. M., Intille S. S. and Larson K., *Activity Recognition in the Home Using Simple and Ubiquitous Sensors*, Proceedings of Pervasive, LNCS, 2004, pp. 158-175.

55. Meier, A., N. Werro, Albrecht M. and Sarkinos, M., *Using a fuzzy classification query language for customer relationship management* Proceedings of the 31st International Conference on Very Large Databases, 2005, pp. 1089-1096.

56. http://datamining.itsc.uah.edu/adam/tutorials/adam_tut_02_overview_05.html

57. Tsechpenakis, G., D. Metaxxas, M. Adkins, J. Kruse, J.K. Burgoon, M.L. Jensen, T. Meservy, D. P. Twitchell, Deokar, A. and Nunamaker, J.F. *HMM based deception recognition from visual cues*, Proceedings of IEEE International Conference on Multimedia and Expo, 2005.

58. Illingworth, F.R., V. Callaghan, and Hagras H., *A Neural Network Agent Based Approach to Activity Detection in AmI Environments*, Proceedings of IEE International Workshop on Intelligent Environments, (IE05), 2005.

59. 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, 2002.

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

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

62. Allen, J.: *Maintaining knowledge about temporal intervals.* Communications of the ACM 26 (1983) 832–843

63. Vilain, M., Kautz, H.: *Constraint propagation algorithms for temporal reasoning.* In: Proc. AAAI-86, Philadelphia, Pennsylvania (1986) 377–382.

64. L. Vila and L. Godo. *On fuzzy temporal constraint networks.* Mathware and Soft computing, 3:315--334, 1994.

65. Bjiorn Gottfried Hans W. Guesgen, Sebastian Hubner, *Spatiotemporal Reasoning for Smart Homes*., Designing Smart Homes, 2006, pp 16-34

66. Carlo Combi and Rosalba Rossato, *Temporal Constraints with Multiple Granularities in Smart Homes*., Designing Smart Homes, 2006, pp 35-56

67. P. Ning, S. Jajodia, and X. S. Wang. *An algebraic representation of calendars.* Annals of Mathematics and Artificial Intelligence, 36:5–38, 2002.

68. R. Basilio, G. Zaverucha, and A. S. d'Avila Garcez. *Inducing relational concepts with neuralnetworks via the LINUS system.* In Proceedings of the Fifth Interna - tiona l Conference on Neura l Inform Processing ICONIP98, pages 1507---1510, 1998.

69. S. V. Vaseghi, *State duration modelling in hidden Markov models*, signal processing, Volume 41, Issue 1 , January 1995, Pages 31-41

70. E. Wiewiora, G.W. Cottrell, and C. Elkan. *Principled Methods for Advising Reinforcement Learning Agents.* ICML 792-799, 2003.

71. A. L. Thomaz and C. Breazeal. *Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance.* In Proceedings of the National Conference on Artificial Intelligence, 2006.

72. V.N. Papudesi and M. Huber. *Learning from Reinforcement and Advice Using Composite Reward Functions*, Proceedings of the 16th International FLAIRS Conference, St. Augustine, FL, pp. 361-365, 2003.

73. J.R. Quinlan. *Induction of Decision Trees*, Machine Learning, (1), 81-106, 1986.

74. A.A. Markov. *"Extension of the limit theorems of probability theory to a sum of variables connected in a chain".* reprinted in Appendix B of: R. Howard. Dynamic Probabilistic Systems, volume 1: Markov Chains. John Wiley and Sons, 1971.

75. John F. Roddick, Myra Spiliopoulou. *A Survey of Temporal Knowledge Discovery Paradigms and Methods*, IEEE Transactions on Knowledge and Data Engineering, vol. 14, No. 4, July/August 2002

76. R. Agrawal and R. Srikant. *Mining Sequential Patterns*, Proc. 11th Int'l Conf. Data Eng., P.S. Yu and A.S.P. Chen, eds., pp. 3-14, 1995.

77. T. Fawcett and F. Provost. *Activity Monitoring: Noticing Interesting Changes in Behavior*, Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining, S. Chaudhuri and D. Madigan, eds., pp. 53-62, 1999.

78. H. Mannila and H. Toivonen. *Discovering Generalised Episodes Using Minimal Occurences*, Proc. Second Int'l Conf. Knowledge Discovery and Data Mining (KDD-96), pp. 146-151, 1996.

79. C. Bettini, S.X. Wang, S. Jagodia, and J.-L. Lin. *Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences*, IEEE Transactions on Knowledge and Data Engineering, vol. 10, no. 2, pp. 222-237, Mar./Apr. 1998.

80. S. Laxman and P.S. Sastry. *A survey of temporal data mining*, Sadhana Vol. 31, Part 2, pp. 173–198, 2006.

81. C.-H. Lee, M.-S. Chen, and C.-R. Lin. *Progressive pattern miner: An efficient algorithm for mining general temporal association rules.* IEEE Transactions on Knowledge and Data Engineering 15: 1004–1017, 2003.

82. N. Meger and C. Rigotti. *Constraint-based mining of episode rules and optimal window sizes.* In Proc. 8th Eur. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'04), Pisa, Italy, 2004.

83. S. Laxman, P.S. Sastry, and K.P. Unnikrishnan. *Fast algorithms for frequent episode discovery in event sequences.* In Proceedings of the 3rd Workshop on Mining Temporal and Sequential Data, Seattle, WA, 2004.

84. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Second Edition, Prentice Hall, 2003.

85. T. M. Mitchell. *Machine learning.* McGraw-Hill Companies, 1997.

86. D. Norman and S. Draper. *User-centered system design: New perspectives on human-computer interaction*. Lawrence Erlbaum Assoc., Mahwah, NJ, 1986.

87. 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, 2002.

88. J.D. Gould and C. Lewis. *Designing for usability – key principles and what designers think.* Proceedings of the ACM CHI Conference on Human Factors in Computing Systems, pages 50-53, 1983.

89. L. Hales. *Intelligent appliances are wave of the future*, January 22, 2006.

90. A. Fox, B. Johanson, P. Hanrahan, and T. Winograd. *Integrating information appliances into an interactive space*. IEEE Transactions on Computer Graphics and Applications, 20(3):54{65, 2000.

91. J. Han and M. Kamber. *Data Mining: Concepts and Techniques* Morgan Kaufmann Publishers, August 2000.

92. B. Schölkopf, K. Tsuda, and J. P. Vert. *Kernel Methods in Computational Biology*, MIT Press, Cambridge, MA, 2004.

93. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.

94. R. Srikant and R. Agrawal. *Mining sequential patterns: Generalizations and performance improvements.* In Proc. 5th Int. Conf. Extending Database Technology (EDBT'96), pages 3-17, Avignon, France, Mar. 1996.

95. J.P. Chin, V.A. Diehl, and K.L. Norman. (1988). *Development of an instrument measuring user satisfaction of the human-computer interface.* Proceedings of SIGCHI '88, (pp. 213-218), New York: ACM/SIGCHI.

96. D. Silver. *Inductive Transfer: 10 Years Later.* NIPS 2005 Workshop.