Managing Adaptive Versatile Environments

G. Michael Youngblood, Diane J. Cook, and Lawrence B. Holder
Department of Computer Science & Engineering
The University of Texas at Arlington
Box 19015, Arlington, Texas 76019-0015
{youngbld, cook, holder}@cse.uta.edu

Abstract

The goal of the MavHome project is to develop technologies to **M**anage **A**daptive **V**ersatile environments. In this paper, we present a complete agent architecture for a single inhabitant intelligent environment and discuss the development, deployment, and techniques utilized in our working intelligent environments. Empirical evaluation of our approach has proven its effectiveness at reducing inhabitant interactions in simulated and real environments.

1 Introduction

The MavHome Project (**M**anaging an **A**daptive **V**ersatile **Home**) is focused on conducting research in smart home technologies from the aspect of treating an environment as an intelligent agent [10]. We seek to develop and integrate components that will enable the intelligent environments of the future. The goals of these environments are to maximize the comfort of the inhabitants, minimize the consumption of resources, and maintain safety and security.

Work in intelligent environments is an important step in the forward progress of technology. As computing becomes more pervasive and people's lives become busier, advances in intelligent environments can aid by automating the simple things (e.g., lighting and HVAC control), work to actively conserve resources (reducing cost), and improve safety and security. Environments that sense their own well-being and can request repair or notify inhabitants of emergencies can save property and lives. Homes that can increase their own self-sufficiency over time can augment busy or aging inhabitants allowing people to live in their homes longer (potentially alleviating some health care system burdens) and free time to allow people to focus on other aspects of their lives. These are just some of the potential benefits of working intelligent environments, research and advancements in this area of science stand to make a large impact on the future.

The goal of this paper is to present one possible engineered approach to developing intelligent environments. We present the MavHome architecture, some of the lessons learned, some of our experimental results, and background work in this area of research.

2 Approach

Our work focuses on learning to automate the intelligent environment. The motivation for this work is the development of systems to meet this focus in an accurate and efficient manner. There are a number of significant challenges in this work in order to meet our goals, which are to learn a model of the inhabitant of an intelligent environment, automate devices to the fullest extent possible using this model in order to maximize the comfort of the inhabitant while maintaining safety and security, and adapt this model over time to maintain these requirements. In order to accomplish these goals, we must first learn a model of inhabitant activities, and then incorporate this into an adaptive system for continued learning and control.

Our development goal is to create an agent based system, which the essence of this system is to perceive the environment through sensors, reason about this information in order to make decisions on whether or not an action should be taken to change the state of the environment in which the agent is situated, and then perform this action through actuators which will affect the perceived state continuing the cycle *ad infinitum*. This work focuses on an agent based system centered around a known single inhabitant in one of our environments.

The sensing and control capabilities of our intelligent environments fit into the generalized models of any sensed and controlled system. The sensors, and for that matter all objects, in our environments are designated with a zone-number combination (e.g., B3) for uniqueness. In our environments, there is a one-to-one correspondence between state and action (e.g., an action such as turning on a light produces the state change of that light being on) which is an attribute shared by many systems but certainly not all.

Environments of this nature provide significant challenges. The largest involves the curse of dimensionality [23]. The state space of an intelligent environment is enormous. For example, if we were to examine a very small environment with ten motion sensors and five lights for a total of fifteen objects and each of these objects has only two states (they are binary) that would give us $2^{15} = 32,768$ unique states. If we can reason about each one for 0.01 seconds it would take 5.46 minutes to make a decision. Our environments have state spaces closer to the size of 2^{150} or 1.43x10^{45} unique states. The size of the problem space makes it difficult to develop real-time reasoning for intelligent environments.

The second largest problem is the curse of generalization. Most approaches to state space reduction involve generalization techniques that reduce the state spaces into similar groups, mostly around commonalities that often do not provide sufficient information to discern context and context-specific action. In the intelligent environment domain where inhabitants are involved in specific local activities generalizing will often produce undesirable results. For example, if an inhabitant reads, listens to music, and watches television all in the same room and the commonality between these events is that the same light is on in the room, then a generalized approach will only provide an automation of that light, missing the desired automation of the CD player and television as appropriate. The challenge is to develop a solution that can maintain a small state space for macro reasoning, but still maintains the details for micro reasoning and automation. It is important for a model to help the system understand the current inhabitant activity with sufficient detail so that correct automation decisions can be made.

The big assumption we make is that people *are* creatures of habit and will provide some periodicity and/or frequency to a number of activities they perform in any given environment, that these patterns can be observed through sensor perception, and that these patterns can be represented as Markov chains. This base pattern representation of a Markov chain represents a certain identifiable pattern of activity or *episode*. These episodes may be abstracted into higher-level episodes that represent a grouping of related episode activity. In order to distinguish pattern permutations when building hierarchies we add history to the transitions to determine the correct transition probabilities.

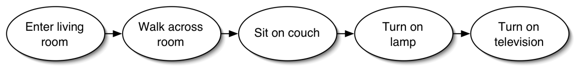


Figure 1: Watching television Markov chain

As a basic example, Figure 1 shows a typical Markov chain of inhabitant activity, namely the pattern of watching television. Patterns similar to watching television such as listening to music and reading a book can be grouped together because they occur in the same space in an environment. Furthermore, activities that occur on that side of the house could be grouped together and eventually all activities in a house fall under the root note as shown in Figure 2.

This location-based hierarchical decomposition of activities illustrates the type of information we are trying to learn about the inhabitants of an environment—how they utilize the environment. The model influenced by location is typical of human partitioned state spaces, but in this work we seek to learn the structure automatically through observation. Hierarchical decomposition of the Markov activity model will be guided by how the inhabitant interacts in the environment. In other words, the hierarchies we learn are based upon observed patterns so that if the inhabitant eats then watches television followed by a period of sleep then those activities are more likely to be grouped together because at a higher level they form a pattern.

In the real world the current state of our environments

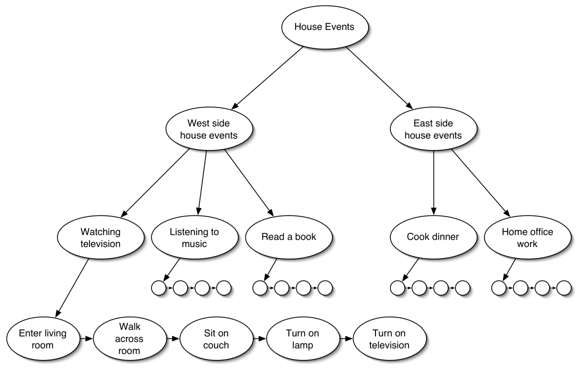


Figure 2: Basic hierarchical Markov model

is never fully understood. We can make observations and infer about the general state of the environment, but the environment is still only partially observable—we cannot observe what takes place in people's minds, in the duct-work, behind the couch, inside the television, and so forth. In our environments what we actually learn are Hidden Markov Models (HMMs). HMMs still describe a process that goes through a series of states, but each state has a probability distribution of possible transitions [34]. Each state also represents a perceived observation that encapsulates many possibly unseen events that are *hidden* from the observer. We depart from the traditional state-based chains of the Markov model which typically represent the entire world state in favor of an event-based chain, one in which the world state is represented only by the single change observed at that point in time. Since our model focus is on a single inhabitant, we concentrate on the changes made to the world state by that individual and assume that the rest of the world has not changed. Our model, as such, represents a chain of events where each event represents the observation that we make at a given point. Each event encapsulates all of the hidden acts that may occur as well. You can build HMMs into a hierarchy and call it a Hierarchical Hidden Markov Model (HHMM). If you tie actions and rewards to the transitions between states this model becomes known as a Hierarchical Partially-observable Markov Decision Process (HPOMDP) [13, 38].

Our main challenge is to learn an inhabitant model solely from observation. The learned model can only utilize data from the perception of the environment and designed mechanisms for converting that data into a useful knowledge representation. The model should be computationally tractable, accurately reflect the interactivity patterns of the inhabitant, and provide for the accurate and efficient automation of the environment.

Learning past the initial model is our second challenge. Automation systems for intelligent environments are only useful in the real world if they can adapt to the ever changing lifestyles of the inhabitants to whom they cater. The system should accommodate for both a slow drift in patterns and for dramatic shifts. The system should adapt quickly while minimizing the loss of accuracy and efficiency. The goal is to provide for the life-long adaptation of the system with the inhabitant of the environment.

This work seeks to utilize information presented to it. The better the quality of information, the better the model, and the better the control policy. Central to our approach is the necessity to recognize the Markov chain patterns of the life of an inhabitant in one of our environments and to recognize the patterns of the abstract patterns, comprising a sequence of the low-level patterns—all from observation data. The large quantities of observed data and the desire to extract the patterns from it have led us to the data-mining community. If we could employ a data-mining technique to discover the periodic and frequent episodes of behavioral patterns in the data, we could use that knowledge to build our inhabitant models. We utilize the work by Ed Heierman in his Episode Discovery (ED) technique [17] as a tool for extracting the desired knowledge from the data stream.

If a data-mining technique can generate knowledge to create a hierarchical model, then in order to be able to use it for automation we will require information that will provide a mapping from the real world observations to the specific location within our model. The event stream coming into the system provides one clue as to which pattern we are currently observing, but may be insufficient to truly pinpoint the exact chain of current activity. What we need to develop is a belief in which state the current world is engaged in order to utilize our learned model to automate future events. Understanding what is the most probable next event to occur would assist in this belief. A prediction algorithm trained on our observation data sets and with reasonable accuracy could be used to provide this type of information. In addition, since our approach is to create hierarchical layers that are labeled as groups of events there will always be a probability of membership given an observation data stream to these groupings. An algorithm that produces a probability of membership given the current event stream to learned groups would provide information that could narrow the choices of specific patterns observed and improve the belief of which state the system is currently engaged. The combination of recent history, the current event stream, a membership probability across the hierarchical layers, and a prediction of the next event to occur yield a belief state of where in the derived model the current inhabitant is interacting. If we look ahead in the model we can determine events that will occur in the near future, and if these events are within the control of the system it can issue actions to automate them.

Invariably, there will be events that escape periodic or frequent patterns, but are desired items for automation. The notion of encoding safety, security, and user preferences into the system is important to our system goals. In order to accommodate these needs, we are employing the use of a rules engine that will maintain a knowledge base of user preference, safety, and security rules and constraints. These rules would incorporate knowledge such as not opening the mini-blinds at night or turning on exhaust fans at high humidity levels. It can also accommodate user preference that could specify rules such as to not automate particular items perhaps because it is their favorite lamp or they just do not feel comfortable with the automation for a particular device. These rules can also be used to incorporate desired events outside the realm of normal observation by the system. For example, patterns that cannot be performed by the inhabitants such as turning off all of the lights when the inhabitant leaves the environment can be encoded as a rule.

Since our goal is to learn how to automate the intelligent environment, the rules engine can also serve as a feedback mechanism. Whenever a rule is violated or fires, feedback can be given to the learning mechanisms of the decision-making component to incorporate into its knowledge for the future. Ideally, the decision-maker would learn not to violate the safety and security rules and automate the inhabitant-designed rules as well.

3 Architecture

A decision-maker is our core control policy component. Our approach is to utilize an overall control algorithm in a three-phase system. The first phase will extract the appropriate observation data from a database and control the data-mining algorithm in order to find patterns and patterns of patterns to build a hierarchical hidden Markov model. This HHMM will be extended with actions and rewards to form a HPOMDP model of the inhabitant for the environment under evaluation. The observation data will also be used to train a prediction algorithm. The observation data and data-mined patterns will be used to train an episode membership algorithm. After the initial information is processed, the model is derived, and components are trained, we can move into the next phase.

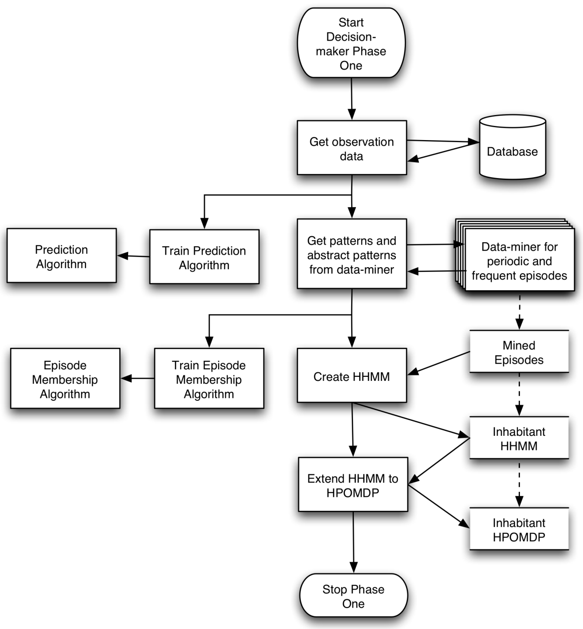

Start Decision-maker Phase One
Get observation data
Database
Prediction Algorithm
Train Prediction Algorithm
Get patterns and abstract patterns from data-miner
Data-miner for periodic and frequent episodes
Mined Episodes
Episode Membership Algorithm
Train Episode Membership Algorithm
Create HHMM
Inhabitant HHMM
Extend HHMM to HPOMDP
Inhabitant HPOMDP
Stop Phase One

Figure 3: Phase 1: Knowledge discovery and initial learning phase

The second phase involves the operational use of the components under the direction of the decision-maker to automate the environment. The decision-maker takes the incoming data stream and provides the information to the predictor and the episode membership algorithms to receive a predicted state and membership probabilities. Based upon the current event, the recent history, the next state prediction, and the probabilities of membership the decision-maker will develop a belief state of where in the learned HPOMDP model the inhabitant's activities are currently engaged. If the belief is strong enough and exists in a series of non-abstract events (i.e., there is sufficient evidence and probability that current observations are part of a known low-level Markov chain) then the decision-maker will look ahead and make an action decision (if one exists). These action decisions automate the environment. While the second phase continues to automate the environment, as feedback is returned from the rules engine and the inhabitant interacting in the environment, we enter the third phase.

The third phase involves adaptation and learning by the decision-maker altering the transition probabilities between events based on feedback in order improve automation performance. These local changes to the model accommodate minor changes in the activity patterns of the inhabitant over time. The decision-maker will also continue to periodically reexamine the historical data using the data-mining tool to determine if new patterns are emerging with the goal of detecting shifts in the patterns. Large lifestyle changes in the inhabitant may lead to a breaking of the current model. In order to accommodate such shifts the decision-maker must evaluate performance and pattern change information in order to contemplate potential reset of the entire system in order to accommodate a major change in the inhabitant's patterns. These three phases are designed to initiate, operate, and maintain a system for the automation of the intelligent environment.

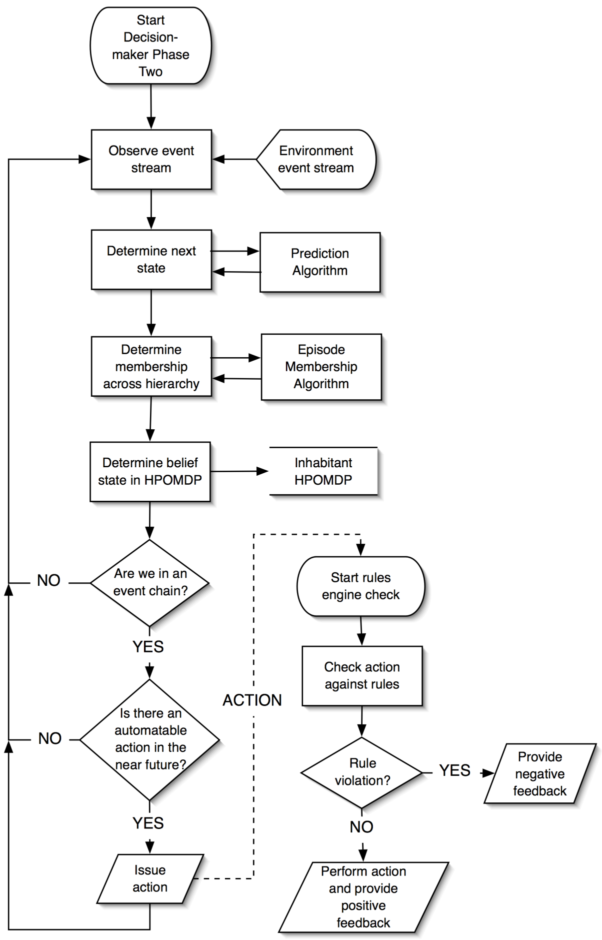
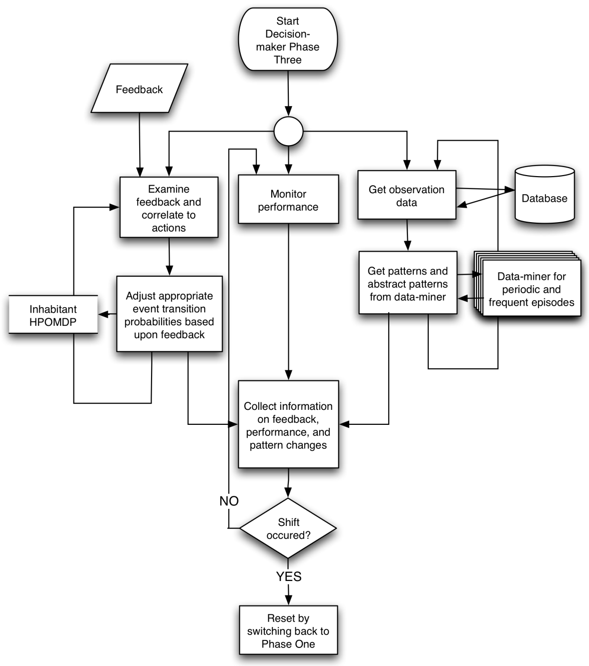

Start Decision-maker Phase Two
Observe event stream
Environment event stream
Determine next state
Prediction Algorithm
Determine membership across hierarchy
Episode Membership Algorithm
Determine belief state in HPOMDP
Inhabitant HPOMDP
NO
Are we in an event chain?
YES
NO
Is there an automatable action in the near future?
YES
Issue action
ACTION
Start rules engine check
Check action against rules
Rule violation?
YES
Provide negative feedback
NO
Perform action and provide positive feedback

Figure 4: Phase 2: Operational phase

There are three distinct phases to our approach as just described. The first phase as shown in Figure 3 is the *Knowledge Discovery and Initial Learning* phase which involves the decision-maker utilizing the data-miner to produce hierarchical knowledge of inhabitant activity patterns, creating a model, and training the prediction and episode membership algorithms. The second phase as shown in Figure 4 is the *Operational* phase which involves observing the event stream and providing current observation data to then receiving next observation and membership probability information from the predictor and episode membership algorithms in order to form a belief state in the inhabitant model. This information is used to potentially make an automation decision. The rules engine is constantly running during this phase. The third phase as shown in Figure 5 is the *Adaptation and Continued Learning* phase which involves feedback from the rules engine to adjust the transition probabilities in the model to improve performance, monitoring of system performance, and the monitoring of data-mined inhabitant activity patterns to observe shifts in the inhabitant's activities. Together this system is designed to learn a model of the inhabitant of the intelligent environment, automate devices to the fullest extent possible using this model in or-



Start Decision-maker Phase Three
Feedback
Examine feedback and correlate to actions
Monitor performance
Get observation data
Database
Get patterns and abstract patterns from data-miner
Data-miner for periodic and frequent episodes
Inhabitant HPOMDP
Adjust appropriate event transition probabilities based upon feedback
Collect information on feedback, performance, and pattern changes
NO
Shift occured?
YES
Reset by switching back to Phase One

Figure 5: Phase 3: Adaptation and continued learning phase

der to maximize the comfort of the inhabitant while maintaining safety and security, and adapt this model over time to accommodate shifts and drifts in the inhabitant's life patterns.

3.1 The System Framework

Given the problem and our chosen approach, it is important to develop a framework in which to support our work. Our system framework is designed of modular components and open source software. Modularity is chosen over a monolithic system to promote ease of maintenance and replacement. The architecture is designed to allow components to be swappable, potentially even hot-swappable, in order to create a robust and adaptive system. We present the framework first in a functional abstract view and then in a detailed concrete form.

3.1.1 Abstract View

The system framework shown in Figure 6 consists of four cooperating layers. Starting at the bottom, the *Physical* layer contains the hardware within the environment. This includes all physical components such as sensors, actuators, network equipment, and computers. The *Communication* layer lies available to all layers to facilitate communication and service discovery between components. The communication layer includes the operating system, device drivers, low-level component interfaces, device proxies, and middleware. The *Information* layer gathers, stores, and generates knowledge useful for decision making. The information layer contains prediction components, databases, user interfaces, data mining components, resource utilization information providers, and high-level aggregators of low-level interfaces (e.g., combined sensor or actuator interfaces). The *Decision* layer takes in information, learns from stored information, makes decisions on actions to automate in the environment, determines if faults occur and correlates them back to using components, and develops policies while checking for safety and security.

Perception is a bottom-up process. Sensors monitor the environment and make information available through the communication layer to information layer components. The database stores this information while other information components process the raw information into more useful knowledge (e.g., predictions, abstractions). New information is presented to the decision layer components upon request or arrangement. The decision layer uses learned experience, observations, and derived knowledge to select an action (which may be no action). The decision is checked for safety and security concerns and, if allowed, signals the beginning of action execution. Action execution flows top-down. The decision action is communicated to the information layer which records the action and communicates it to the physical layer. The physical layer performs the action, thus changing the state of the world and triggering a new perception. The process repeats *ad infinitum* with periodic retraining of the decision layer components, policy development, database archiving, and component maintenance.

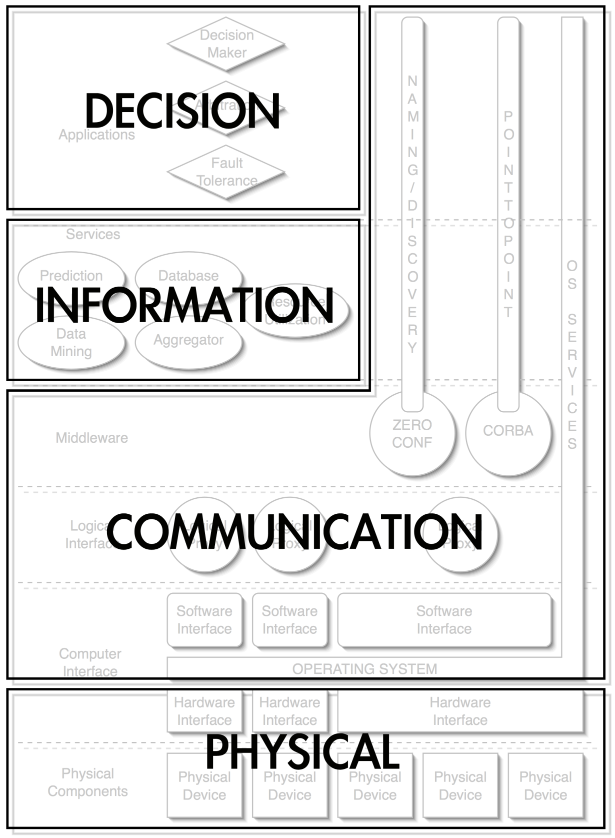


Figure 6: Abstract framework

3.1.2 Concrete View

The abstract layers of the system framework are realized through a set of concrete functional layers. These concrete layers are shown with components in Figure 7. The base layer is the *Physical Components* layer which consists of all real devices utilized in the system. These devices include powerline control interface hardware, sensor networks, input devices, cameras, and so forth, with the exception of the computer with which equipment is interfaced. The physical computer(s) and associated network this system resides on is considered the host of all layers above the physical. The *Computer Interface* layer contains the hardware interfaces to physical devices (e.g., PCI card interfaces, USB, Firewire), device drivers to utilize the hardware, the operating system of the computer, and all software interfaces that provide services or APIs for hardware access. It should be noted that since all components of above layers reside and utilize operating system services, these services are shown to extend to all layers. In the *Logical Interface* layer, the hardware device services and APIs are utilized to create simple, light-weight programs that create a series of atomic services around each sensor and effector in the system. These *logical proxies* provide information and control via socket and shared memory based interfaces in a modular design. All of the lower layers are based on simple single application components, but in higher layers the components become more complex. The *Middleware* layer provides valuable services to the upper layers of the architecture to facilitate communication and service discovery. The MavHome architecture specifies middleware that provides both point-to-point (through CORBA) and publish-subscribe (through multicast messaging utilizing OS socket services and the IP stack) types of communication and naming/service discovery provisions. The *Services* layer utilizes the middleware layer to gather information from lower layers and provide information to system applications above. Services either store information, generate knowledge, aggregate lower-level components, or provide some value-added non-decision making computational function or feature (e.g., user interfaces). The *Applications* layer is where learning and decision-making components operate.

3.1.3 Implementation

To provide the reader with a better understanding of the system framework we employ, we will discuss some implementation-specific details.

Lighting control is the most prominent effector in most intelligent environments. We currently use X-10-based devices in the form of lamp and appliance modules to control all lights and appliances. The CM-11A interface is used to connect computers to the power system to control the devices. Radio-frequency based transmitters (in remote control form factor) and receivers are also used for device interaction. X-10 was chosen because of its availability and low price. Many home users utilize X-10 technology, so immediate benefits to the current home user are possible.

Perception through light, humidity, temperature,

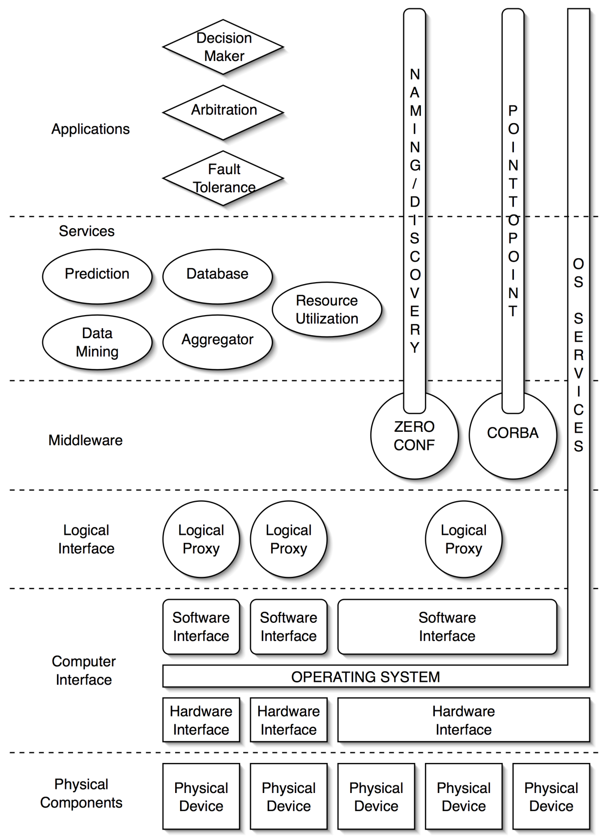


Figure 7: Concrete framework

smoke, gas, motion, and switches is performed through a sensor network we developed. The Argus network system is a PIC16F877-based system comprised of a master board that interfaces to the computer via a serial interface and connects up to 100 slave boards that host up to 64 sensors each, ganged in groups of four on a sensor dongle. Special masters have also been developed for high speed digital and mixed digital/analog sensing applications. A stepper-motor master has also been developed to control up to four mini-blinds.

A key element in perception is inhabitant localization. The Argus Digital Master is used in conjunction with passive infrared (PIR) sensors placed on the ceiling in traffic areas to detect motion. The sensors have a 60° field of view and are placed between eight and ten feet from the ground depending on the height of the ceiling. In order to reduce the sensing area, tubes are placed over the sensors to reduce the floor footprint to a three to four foot sensing circle. Tests in our environments show a consistent single inhabitant location detection rate of 95% or better accuracy. Multiple inhabitant studies will require augmenting technology.

All system framework components interface through either serial, USB, or firewire interfaces. The system framework and components have been developed on Intel-based PCs (Pentium 4) and use the Linux operating system (SuSE 9.1).

The logical interfaces for all X-10 and Argus-based components have been written as light-weight configurable modules. The logical proxies maintain the current state of each device and provide a mechanism for reading and, if applicable, control. The communication protocols for X-10 devices and Argus components are well defined and interface availability is advertised through zero configuration (ZeroConf) technology [19].

Components desiring to find X-10 or Argus components merely need to perform a link-local query for devices that follow the defined MavHome X-10 and Argus protocols and a list of available devices will be presented to the requester. Contact information is returned to the requester to allow connection to the logical proxy. Through this mechanism no configuration is required and the system is very adaptive and dynamic. New proxies advertise their availability and older ones remove theirs before they shut down. We have had a high level of success using ZeroConf technology with very few problems once the components were developed. When we were using a CORBA name server we had close to a 50% component communication or discovery failure rate at any given time mostly due to CORBA TCP communication errors in the name server ORB.

The system framework uses two main middleware packages. Communication between high level components is performed using the Common Object Request Broker Architecture (CORBA) due to the clarity of interface design provided by the Interface Description Language (IDL), ease of integration, maturity and stability of the technology, and object-oriented design compatible with our C^{++} implemented components. Zero configuration technologies are used for replacing the CORBA naming service and utilizing service discovery. They are provided by the Apple Multicast DNS responder and adherence to the ZeroConf standard.

Implemented services include a PostgreSQL database that stores information, user interfaces, prediction components, data mining components, and logical proxy aggregators (e.g., the projector screen aggregator that takes simple "up" or "down" commands to coordinate the efforts of a timed control of three switches to place the screen in the proper position). Resource utilization services monitor current utility consumption rates and provide usage estimates and consumption queries, but are not used in this work.

The core of this work resides at the application layer which along with some of the services comprise the core system architecture of this approach.

3.2 The System Architecture

Inside the system framework exists the core system architecture for our approach. At this point, we have presented our overall approach, architecture, the system framework, and base technologies. In this section, we will describe the specific information and decision components implemented with the techniques they employ to create our working system.

3.2.1 ProPHeT

Decision making is performed in the ProPHeT (**Pro**viding Partially-observable **H**idden (HMM/ POMDP) based d**e**cision **T**asks) component. The world representation at this level is the Hierarchical Hidden Markov Model (HHMM) [13] based upon a hierarchy of episodes of activity mined from stored observations. Episode Discovery (ED) is used to generate low-level episode Markov chains and build the hierarchy of abstract episodes under the direction of ProPHeT. Learning is performed by extending the HHMM to a hierarchical Partially Observable Markov Decision Process (HPOMDP) and applying temporal-difference learning. Constant feedback from ARBITER is used for continuous learning using TD(0) reinforcement learning [37]. Action decisions are made by using the incoming event stream, recent history, the stream episode membership features of Episode Membership (Epi-M) to provide input into the current belief state in the model, and the Active LeZi (ALZ) prediction of the next event to chose the appropriate transitional action.

3.2.2 Episode Discovery (ED)

The Episode Discovery (ED) data-mining algorithm discovers interesting patterns in a time-ordered data stream. ED processes a time-ordered sequence, discovers the interesting episodes that exist within the sequence as an unordered collection, and records the unique occurrences of the discovered patterns. These features make ED a suitable algorithm for mining an intelligent environment data stream.

Our approach to state space reduction from the large number of potential environment observations is to abstract inhabitant activity to episodes that represent the current task of involvement. Given the inhabitant task episode, observations not related to the task can be pruned. A difficult problem is how to discover these episodes.

We use the *Episode Discovery* (ED) algorithm [16] for finding inhabitant episodes in the collected data and for episode classification of streamed observations. ED is an input, not transaction, based algorithm that mines device activity streams trying to discover clusters of interactions that are closely related in time. Significance testing is performed on discovered clusters to generate sets of significant episodes based on the frequency of occurrence, length, and regularity. Further processing using the Minimum Description Length (MDL) principle [33] and greedy selection produces sets of significant episodes. These are labeled and directly correspond to an inhabitant task.

When an inhabitant is first introduced to an intelligent environment no automation should occur for an initial observation period. This allows the building of a database of potential episodes of normal task activity. This is inhabitant centric and the observation period duration is determined by data compressibility which is used to determine the stability of the data with relation to episode discovery. A stable, consistent data compression as reported by ED indicates an end to observation. Identification of concept drift and shift is performed by continued monitoring of streaming data and compressibility. Changes in compressibility indicate a need to re-evaluate the discovered episodes.

Episode discovery, classification, and identification are utilized to reduce the state space of an intelligent environment to a set of inhabitant-centric tasks. Thus, the

MavHome architecture is inhabitant-centric.

3.2.3 Active LeZi (ALZ)

An intelligent environment must be able to acquire and apply knowledge about its inhabitants in order to adapt to the inhabitants and meet the goals of comfort and efficiency. These capabilities rely upon effective prediction algorithms. Given a prediction of inhabitant activities, MavHome can decide whether or not to automate the activity or even find a way to improve the activity to meet the system goals.

Specifically, the MavHome system needs to predict the inhabitant's next action in order to automate selected repetitive tasks for the inhabitant. The system will need to make this prediction based only on previously-seen inhabitant interaction with various devices. It is essential that the number of prediction errors be kept to a minimum–not only would it be annoying for the inhabitant to reverse system decisions, but prediction errors can lead to excessive resource consumption. Another desirable characteristic of a prediction algorithm is that predictions be delivered in real time without resorting to an offline prediction scheme.

Based upon our past investigations, MavHome uses the *Active-LeZi* algorithm [14] to meet our prediction requirements. By characterizing inhabitant-device interaction as a Markov chain of events, we utilize a sequential prediction scheme that has been shown to be optimal in terms of predictive accuracy. Active-LeZi is also inherently an online algorithm, since it is based on the incremental LZ78 data compression algorithm.

3.2.4 Episode Membership (Epi-M)

Effective utilization of the derived HHMM/HPOMDP-based inhabitant model requires an understanding of how to map the current observation stream into the derived abstractions. Episode Membership (Epi-M) performs this function by using the information learned from Episode Discovery to build internal correlation tables and further augment those tables with time-based occurrence information based on circular probability [5] capture from the same data stream. Data stream observation over the specified window span supplied to ED can be used to generate match probabilities with the episode sets over each layer of abstraction. Augmenting the probability with the likelihood of occurrence based on the observed occurrence time distribution for each of the discovered episodes with relation to the current time further improves the accuracy of possible episode membership reporting. For example, if the current observation stream matches with 90% probability either reading a book or the pattern of sleeping on the couch, but the inhabitant has never slept on the couch at this time of day, then the probability of sleeping can be discounted to promote reading as the most probable episode of membership. Epi-M output is used by ProPHeT to determine belief state in the operational phase for the current event observations.

3.2.5 ARBITER

When issues of safety and security are of the highest importance in a system there is the need for an enforcer of rules before actions are made. This system works by using a knowledge base of rules and evaluating each action event against these rules to determine if the action violates them. Actions in violation will be prevented from occurring and feedback will be sent back to the originating system (i.e., the decision-maker). Rules are not required to be just of a safety and security type, any type of rule can be used in order to guide the behavior of the system. Cases where system behaviors are desired but will never be trained by streaming data or interactions can be handled by the addition of rules to provide feedback and facilitate learning of the desired behavior.

Before an action is executed it is checked against the policies in the policy engine, ARBITER (A Rule-Based InitiaTor of Efficient Resolutions). These policies contain designed safety and security knowledge and inhabitant standing rules. Through the policy engine the system is prevented from engaging in erroneous actions that may perform such activities as turning the heater to 120° F or from violating the inhabitant's stated wishes (e.g., a standing rule to never turn off the inhabitant's night light).

These components work in concert to learn, adapt, and automate the inhabitants' lives in an intelligent environment.

3.2.6 MavCore

The core of this work lies in the data-mining–decision-making–belief–rule/feedback chain or in what we call the EPBA chain comprised of the Episode Discovery (ED), ProPHeT, belief through Active LeZi and Episode Membership (Epi-M), and the ARBITER components.

Information and action flow through the system according to the three main system phases. These phases as previously stated can be restated as being *Initialization, Operation,* and *Adaptation*.

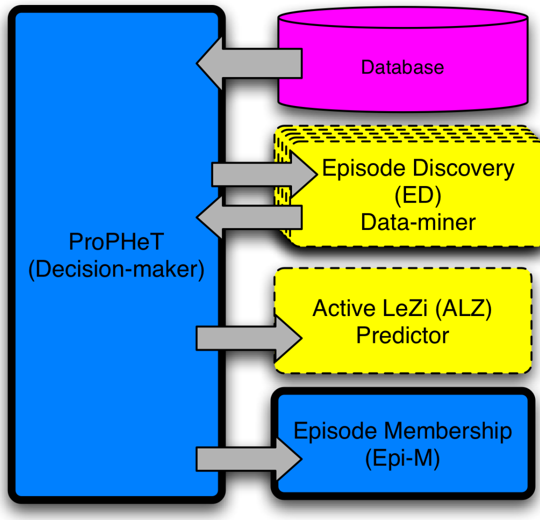

Database
Episode Discovery (ED) Data-miner
ProPHeT (Decision-maker)
Active LeZi (ALZ) Predictor
Episode Membership (Epi-M)

Figure 8: Core system architecture in initialization phase

The structure of the core components in the initialization phase and the flow of information are shown in Figure 8. During initialization, information flows from the database through ProPHeT into ED. As many instances of ED as are necessary to process the data into a hierarchy of discovered patterns are started by ProPHeT, provided information, and return information on the discovered patterns to ProPHeT. ProPHeT then trains ALZ over the same observation data from the database. ProPHeT determines what information to pull from the database and performs the necessary data conversions and filtering for each component to accept the data and perform computation within a reasonable amount of time. After ALZ, ProPHeT trains Epi-M with the same observation data and the returned hierarchical data returned from ED. After deriving the data from the target observation data set and training the belief supporting components, ProPHeT generates the HHMM and subsequent HPOMDP models.

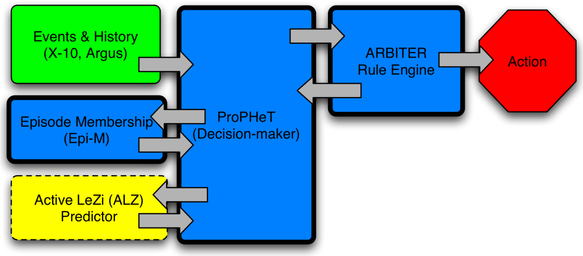

Events & History (X-10, Argus)
Episode Membership (Epi-M)
Active LeZi (ALZ) Predictor
ProPHeT (Decision-maker)
ARBITER Rule Engine
Action

Figure 9: Core system architecture in operation phase

The structure of the core components in the operation phase and the flow of information are shown in Figure 9. During operation, information flows from events generated in the environment and perceived by the logical proxies and presented to ProPHeT. ProPHeT relays these events to ALZ and Epi-M and receives prediction and membership information. Based upon the incoming event, history, prediction, membership probabilities, and the HPOMDP model an action decision may be made. Any action decisions, or at a minimum the current event, will flow through to ARBITER to be checked for rule violations. If a satisfactory action is to be performed, ARBITER will contact the appropriate logical proxy to initiate the action.

The structure of the core components in the adaptation phase and the flow of information are shown in Figure 10. During adaptation, information flows from the event stream through ProPHeT to ARBITER, often accompanied with an action. Rule violations and any other feedback are relayed back to ProPHeT from ARBITER including inhabitant feedback correlation to countermanded automation. ProPHeT uses feedback from ARBITER to adjust the HPOMDP structure to improve performance and accommodate for pattern drift. Internally ProPHeT is evaluating performance based on feedback and usage. Information also flows into and out of ED as ProPHet will periodically evaluate the continuously-growing observation database—it should be noted that in our system the

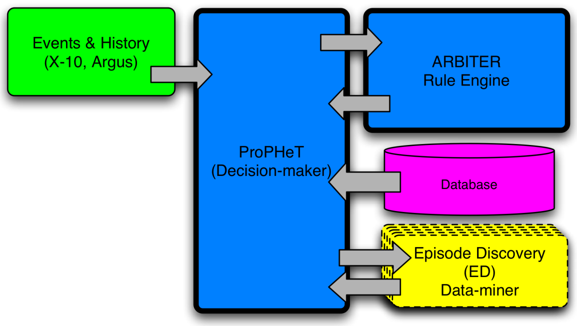


Figure 10: Core system architecture in adaptation phase

database component has an event listener that logs all events into the database—for changes in patterns and hierarchy in order to detect pattern shift using ED continually in the background. ProPHeT will decide based upon performance as well as indications of pattern shift and drift whether a reset of the decision-maker is required. The reset is essentially a reboot of the system using a new set of observations—one that will hopefully better fit the inhabitant.

The operation and adaptation phases occur simultaneously until interrupted by the adaptation phase, usually to return through the initialization phase. All components with a bold, solid border in the architecture Figures 8, 9, and 10 represent components that are developed as part of this work, items with a dashed border are components developed on the MavHome project but were developed by others, and components with thin borders represent data or storage.

This is only one possible approach to developing a solution given the problem and goals. There are many other possible approaches, but others remain to be explored. We are exploring this method because it had not been previously explored in this domain.

4 Experimental Environments

This work uses two real environments and their simulated counterparts. The MavPad is an on-campus apartment and the MavLab is the workplace of the researchers of this project. ResiSim is an in-house developed "residential simulator" for interactive simulation of intelligent environments.

4.1 MavPad

The MavPad is an on-campus apartment that hosts a full-time student occupant that participates in the project allowing us to learn about them and automate their life. The MavPad hosts automation capability through 25 X-10 controllers (3 fans, 13 lights, 1 HVAC unit, and 7 electrical outlets) and 2 ArgusM mini-blind control systems. Sensing capability is provided by the ArgusMS and ArgusD systems that provide 18 light, 11 temperature, 4 humidity, 4 leak detection, 4 door open/close, 3 window open/close, 2 seat occupancy, 4 HVAC vent position, 2 smoke detectors, 2 CO detectors, and 36 motion sensors. The MavPad has been operational for over a year and has hosted three inhabitants.

4.2 MavLab

The MavLab is the project name for the Artificial Intelligence Lab which is base of operations for this research. The MavLab is a workspace setting with offices, cubicles, a break area (MavKitchen), a lounge (MavDen), and a conference room. The MavLab hosts automation capability through 54 X-10 controllers (49 light, 5 appliances, a projection screen) and 14 ArgusM mini-blind control systems. Sensing capability is provided by the ArgusMS and ArgusD systems that provide 36 light, 10 temperature, 3 humidity, 2 door open/close, 6 seat occupancy, and 25 motion sensors. MavLab has been in various operational states for the last two years.

4.3 ResiSim

It is important that this work be well grounded in reality and in dealing with the real world, especially with the goals being directed to the intelligent environment domain. This research work is aimed at real-time decision-making in the real-world. However, it is not always feasible to study interactions solely in the real world. A tool for interactive simulation and data visualization was needed. Since this is a relatively new area of study there are not any available simulation tools designed for intelligent environments or home simulation, so the development of a tool became an important issue for performing this type of work.

ResiSim is a residential simulation environment tool. It is designed to provide a simulation environment for any indoor environment where people would typically spend time. It features Markov model-based virtual inhabitants that interact with the environment and can react to changes in that simulation. It can also provide real inhabitant data playback and limited interaction. As an evaluation tool it can track automation and its source whether from an external system, playback, or elsewhere.

5 Experimentation and Results

We have implemented and tested our systems in the previously described environments and present some of our findings starting with simulation work with MavLab in ResiSim followed by MavPad experiments involving real inhabitants.

5.1 MavLab Simulated Inhabitant

As an illustration of our technique and the software system, we have evaluated a simulated typical week in an inhabitant's life with the goal of reducing the inhabitant's interactions in the MavLab. The data was generated from a ResiSim virtual inhabitant based on captured data from the MavLab and was restricted to just motion and lighting interactions which account for an average of 1400 events per day (filtered to 250/day). There are on average 25 lighting device interactions a day with the remainder being motion information. Using our ResiSim (Residentual Simulator) tool which exactly replicates the real MavLab, we trained ALZ and ED on real data and then repeated a typical week in the simulator to determine if the system could automate the lights throughout the day in real-time.

ALZ processed the data and converged to 99.99% accuracy during training on test data from the data set in 10 iterations. When the system was run with automation decisions being made by ALZ alone, it was able to reduce interactions by 9.7% event as shown in Figure 13. ALZ performance on streaming data maintained between 24-56% accuracy converging to 54% as shown in Figure 11.

ED processed the data and found 10 interesting episodes, 8 that correspond to automatable actions. This was abstracted through ED to two abstract nodes. A HPOMDP was constructed in ProPHeT. This system was able to reduce interactions by 76%. As a comparison, the HHMM produced was flattened and the abstract nodes removed to produce a flat HMM. This HMM was still able to reduce interactions by 38.3%. Comparative results are shown in Figure 12.

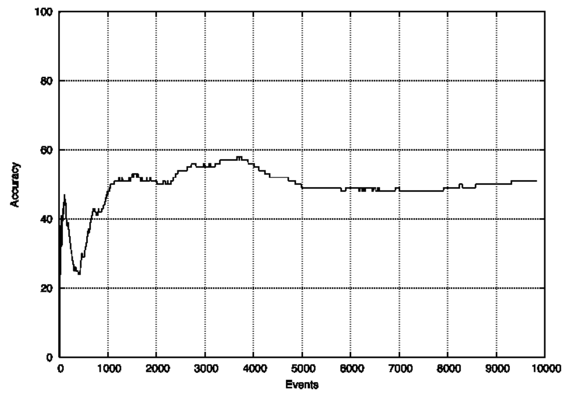


Figure 11: ALZ Accuracy.

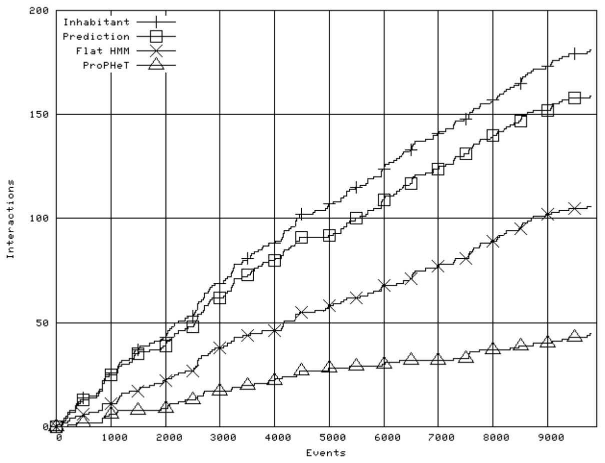


Figure 12: MavLab virtual inhabitant interaction reduction.

The additional abstractions in the hierarchy coupled with a next state produced by ALZ and a probability of membership from ED to provide input to the belief state create a system that improves automation performance over a flat model or prediction alone. This simulation provided very clean data with consistent patterns which made for good episode discovery results. Some of the patterns started with automatable events which make them almost impossible to automate because when ProPHeT believes that the episode is being observed, the automatable event opportunity has already passed. This experiment represents some of the best performance of this technique, but it does not include any ARBITER rules—only automation through the learned model.

5.2 MavPad Inhabitant 2

As an illustration of the viability of the architecture, the techniques described in this paper, and the MavHome system deployed in the MavPad, we have evaluated a typical day in the inhabitant's life with the goal of reducing the inhabitant's interactions with the lighting in the MavPad. The data was restricted to just motion and lighting interactions which account for an average of 10,310 events per day (filtered to approximately 420 events/day). There are on average 18 lighting device interactions a day with the remainder being motion information. Using our ResiSim tool which exactly replicates the real MavPad, we trained ALZ and ED on real data and then repeated a typical MavPad inhabitant day in the simulator to determine if the system could automate the lights throughout the day in real-time.

ALZ processed the data and converged to 99.99% accuracy on test data from the data set. When the system was run with automation decisions being made by ALZ alone, it was able to reduce interactions by one event as shown in Figure 13. ALZ performance on streaming data maintained between 40-60% accuracy.

ED processed the data and found 10 interesting episodes that correspond to automatable actions. This was abstracted through ED to three abstract nodes. A HHMM was constructed in ProPHeT. Figure 14 shows this model without the production nodes (it is difficult to show the full models because even the simple models when printed take over seven feet of paper in order to be legible). This system was able to reduce interactions by 72.2% to five interactions. As a comparison, the HHMM produced was

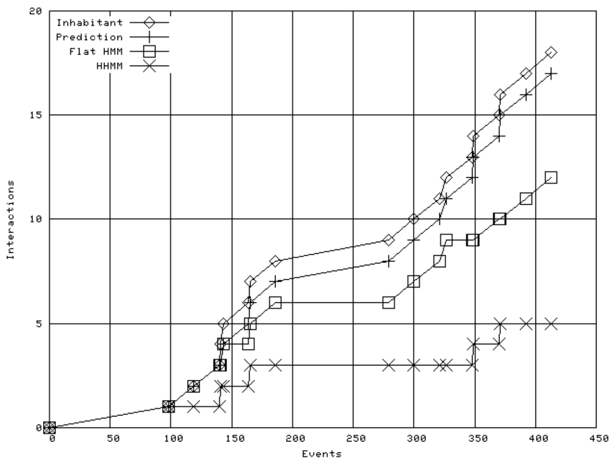


Figure 13: MavPad Inhabitant 2 Interaction reduction.

flattened and the abstract nodes removed to produce a flat HMM. This HMM was still able to reduce interactions by 33.3% to 12. Comparative results are shown in Figure 13.

We also expanded the experiment beyond the automation of just a single day of real inhabitant data to a full ten days under the same conditions. Performance dropped to 54.9% reduction (83 automations out of 184) for ProPHeT, 26.6% (49/184) for a flat HMM, and a 10.9% (20/184) reduction for ALZ. Missed automations were due to insufficient belief state for automation due to some pattern performance inconsistencies and two patterns that began with an automatable action. Progressive degradation is a fact of the real data and increasing pattern inconsistencies as well as inhabitant activity patterns that were not discovered originally because they lacked sufficient frequency or periodicity.

5.3 MavPad Inhabitant 3

Our third MavPad inhabitant has lived in the apartment for nine months undergoing various levels of observation and automation. During the course of this time we have worked closely with the inhabitant to develop good ARBITER rules from a safety and security standpoint, user preference, and some general desired automations not captured by episode discovery. Inhabitant three is a very busy person with a somewhat erratic schedule that made

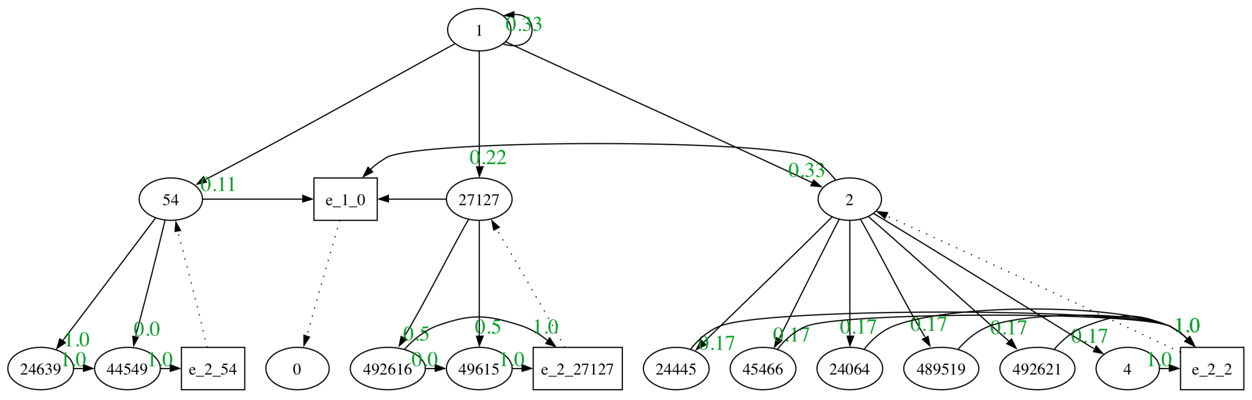


Figure 14: Learned HHMM from MavPad inhabitant 2 data.

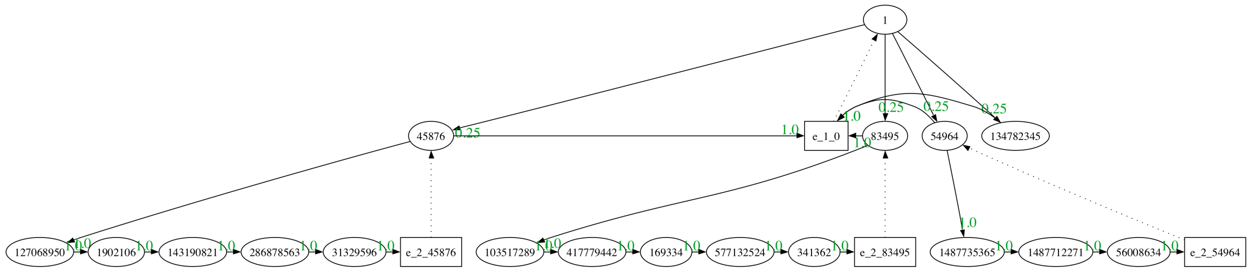


Figure 15: Learned HHMM from MavPad inhabitant 3 data.

experimentation very interesting.

A three week full automation experiment was conducted. Seven weeks of data was used to train ED which in the raw form consisted of 4,371,179 events (apx. 89K/day) with 2,163 automatable events and was reduced to 21,863 events (apx. 446/day) with 1,952 automatable events by filtering unnecessary and noisy data—our filters key in around the automation areas and filter out large spans of data where no automations occur, we also filter some duplicate data and data inconsistencies. There are on average 40 device interactions a day which included lights, fans, and mini-blinds (tri-state) with the remainder being motion information. ED processed the data for over 96 hours to find 14 production node-filled abstract episodes and three hierarchical abstractions of those for a four-tier HHMM as shown in Figure 15. The model was extended to a HPOMDP and ARBITER was loaded with two safety and security rules, one inhabitant preference rule, and seven general automation rules. ALZ and Epi-M were trained—ALZ trained to within 99% as previously observed.

The full system was allowed to automate the environment which included lights, fans, and mini-blinds and was able to automate 39.98% (345/863) of the inhabitant's life as shown in Figure 16. The system learned the 2 safety and security rules and the inhabitant preference rule within 6 violations with ARBITER feedback. Those rules were all based on preventing specific lights from being automated. The system learned two of the seven general automation rules because they were modifications of existing episodes, but the others were not learned because they did not associate to existing episodes. With regularity those rules should be found by ED and eventually added to the system on a reboot—we did observe one of

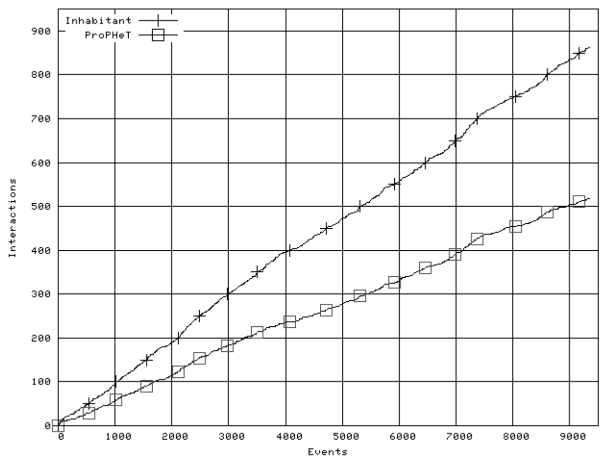


Figure 16: MavPad inhabitant 3 interaction reduction.

the seven show up as a frequent episode in the continuous ED checks for shift and drift. Figure 17 shows the reduction of rule firing over the experiment. Note that in that figure the rule violation curve slope decreases over time as the system learns rules and the number of violations decreases over time. On average five rules per day were fired for what would have been a total of 112 over the experiment, but only 81 actually fired yielding a 27.7% reduction in rule violations.

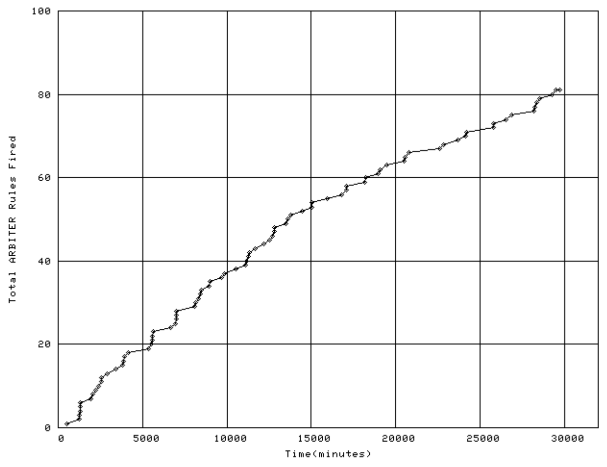


Figure 17: MavPad inhabitant 3 rule violation reduction.

The experiment brought forth some interesting observations. Inhabitant patterns for turning off household items (particularly lights) is much more consistent than turning them on. Object off patterns make up approximately 60% of the episodes and are also more easily learned by the system—all of the adaptations involved turning objects off. Inhabitants are slow to correct the system, and are willing to accept incorrect actions. There is a definite need to work more in a partnership with the inhabitant to meet system goals.

5.4 Observations

There are a number of interesting observations we have made in the course of design, implementation, and experimentation. The more unique challenges have come from working with our MavPad inhabitants. We did not automate our first inhabitant because we were still concentrating on sensor reading acquisition and determining if there was a perceivable pattern to inhabitant activities—fortunately, there are discoverable patterns. It was observing the first inhabitant that made it clear that we needed a good period of observation and that it would take several weeks if not months depending on the consistency of the inhabitants lifestyle. Our second inhabitant participated for a summer and was automated for two weeks using ARBITER and a very basic version of ProPHeT (without adaptation). This inhabitant lived a very regimented lifestyle—even taking showers at approimately the same time every day (surprisingly between 1:00-1:30 am). Our third inhabitant lives a very chaotic lifestyle and has been a challenge for our systems.

Sensor network failure, unreliability, and general chaotic behavior at times forced a lot of effort towards improving the systems by adding additional fault tolerance mechanisms, watch dog timers, performance monitors, and many additional software objects that focus on maintaining high-availability. Sensor instability due to software problems and I^2C communication errors plagued the first couple of months, but those issues have been corrected and the sensors are very stable. One of the biggest problems on the project so far has been the stability of the power grid and power loss issues in the stormy seasons. Operating system application stability for programs operating 24-7 has also been problematic.

A consistent problem we have is that many episodes actually begin with the automatable event; thus, automation is difficult if not impossible given our approach in those instances. Interestingly, ED finds many patterns that are filtered due to a lack of automatable actions. There is often a large number of pacing patterns—back and forth walking in the MavPad. The inhabitants, all students, have all admitted to this behavior when the are "thinking."

With inhabitant three, we noticed a new phenomenon in the course of our experimentation—the system did more training of the inhabitant than the inhabitant did to the system. There seemed to be a reluctance to give prompt feedback on the inhabitant end. On interview, the inhabitant said that they were learning to live in the dark because it was too bothersome to correct the system. This is probably human nature. We also observed a few fights between the system and the inhabitant over control that ultimately was won by the inhabitant when feedback caused the system to change behavior, but for a short duration the system caused some duress to the inhabitant—not a desired effect.

6 Related Work

Our work focuses on the emerging domain of intelligent environments or smart homes and buildings. Generally, these environments are defined by the way in which people interact with them or in the way that these places interact with the inhabitants. Benefits include providing comfort and productivity for inhabitants and generating cost savings for utility consumption. There are many researchers working on interesting problems in this domain. Due to space considerations we provide a brief mention of work not directly related to our own to provide a starting point for the interested reader, and we present more information on those projects closely related to our own.

There are many projects that are engaged in developing frameworks to support intelligent environment work. These include the MIT Artificial Intelligence Lab AIRE (Agent-based Intelligent Reactive Environments) group [3], the AMBIENTE division of the Fraunhofer-IPSI Research Institute in Germany who is a partner of the fifteen member European AMIGO project [4], the Interactive Workspaces project at Stanford University [36], the Laboratory for Communication Engineering (LCE) at Cambridge University (originally in conjunction with AT&T Laboratories Cambridge) [6], the Gaia project at the University of Illinois at Urbana-Champaign [39], and the University of Florida Gator Tech Smart House [18].

Other projects focus more on applications and developing useful objects. These include the Aware Home Research Initiative (AHRI) at the Georgia Institute of Technology [2] and MIT Media Lab's Consortia on Things That Think (TTT) and their special-interest group on Counter Intelligence [27].

Industry is interested in providing services and technology for the intelligent environment and have embarked on several initiatives. These include the Pervasive Computing Lab at IBM Research [40], the Vision Group at Microsoft Research and their Easy Living project [25], British Telecom's Telecare project [7], CISCO Network's Internet Home [8], Intel Corporation's Proactive Health Lab [22], Siemens AG's "living made easy" home automation products [35], Royal Philips HomeLab [32], and Accenture's Room of the Future [1]. A number of commercial retail companies are selling products to make people's homes "smart." The most prominent of these companies is SmartHome (www.smarthome.com).

There are also a number of healthcare initiatives. These include the Medical Automation Research Center (MARC) smart house project at the University of Virginia [24] and the Robert Gordon University CUSTODIAN (Conceptualization for User involvement in Specification and Tools Offering the efficient Delivery of system Integration Around home Networks) project [11].

There are two prominent learning and adapting initiatives that are similar to ours.

The Adaptive House project at the University of Colorado at Boulder under the direction of Michael Mozer tackles the issue of overcoming the programming problem with home automation (i.e., where someone must program the rules for automation and reprogram them over time as the inhabitant's lifestyles change). Their work involves developing a system that controls the HVAC, water heater, and interior lighting of a home, learning how to control these features based on the lifestyle and desires of the inhabitants, and adapting the control policy over time in an environment with a minimal user interface. This project uses an actual residence called the neural network house equipped with 75 sensors that monitor temperature, ambient light levels, sound, motion, door and window openings as well as actuators that control the furnace, space heaters, water heater, lighting units, and ceiling fans [28]. The control systems in this work are based on neural network, reinforcement learning, and prediction techniques called ACHE (Adaptive Control of Home Environments). Q-learning (a reinforcement learning technique [37]) uses event-based segmentation over clock-based in order to make the problem tractable and initiates actions based on perceived state and reward. In order to simplify the state space the automation task was decomposed into zones and a heuristic based determination of event separation factors was used to partition the experience into events for the event-based control system. The control policy involved a mixing of inhabitant comfort and energy conservation goals. The system used a state estimator to form high-level state representations which were based upon inhabitant activities (through an occupancy model and an anticipator which was neural network based and provided a prediction of occupancy of a space) and light levels in each zone (through a natural light estimator). This information and the decomposition of spaces were utilized with multiple Q controllers to automate the home. Their work also involved some exploration in order to reduce energy consumption by occasionally testing the inhabitant by altering the control policy unless counteracted by the inhabitant [30].

The adaptive house involved a lot of knowledge engineered into the system in the form of event partitioning heuristics, look-up tables for Q-values, and human-directed partitioning of the system into multiple Q-learners controlling specific areas of the house. In relation to our work, the anticipator and occupancy models are not necessary in our systems due to a better hardware design and implementation. We agree with the vision and premise of the adaptive home—that an intelligent environment should adapt to the needs, lifestyle, and desires of its inhabitants [29]. We also agree with the idea of using a minimal and natural interface as well as focusing on the goal of solving the user programming problem. Many of the heuristic decisions and partitioning schemes in the adaptive house are avoided by learning the requirements for decompositional and hierarchical construction of a control policy through observation. In earlier versions of our work we utilized Q-learning with CMAC tiling in a control mechanism similar to Mozer's work [9]; however, we also discovered similar state space abstraction problems in automation and a need for better sensor information in order to accomplish our goals. Our work can be seen as a direct extension of this shared vision towards creating environments with control policies observed from their inhabitants.

Researchers from the Intelligent Inhabited Environments Group (IIEG) at the University of Essex in the United Kingdom are creating an *ambient-intelligence* environment using embedded agents called the *iDorm* [20]. Their approach involves the use of a fuzzy logic based *Incremental Synchronous Learning* (ISL) system to learn and predict inhabitant needs. Their testbed environment, a dorm room, involves the access of 11 environmental parameters and nine effectors (mostly lights). The use of parallel fuzzy logic controllers (FLC) in a hierarchy is used to learn and encode rules. Each FLC has a single modifiable parameter and is used to learn a particular aspect of the environmental control. The FLCs are either static (i.e., pre-seeded with knowledge) or dynamic (i.e., observed from the inhabitant). In combination all of the FLCs form the ISL system and encode the desired control behavior of the environment. Other management systems prune down the number of FLCs by observing factors of redundancy and low usage to keep the system computationally manageable. The researchers have presented evidence via empirical evaluation of iDorm inhabitants that the system can perform initial and lifelong learning of inhabitant needs over a 132 hour experiment [15].

The iDorm project has many similarities to our work. The notion that there first needs to be an observation period followed by usage and learning has been a part of our systems. The general focus on learned automation is also the same. Besides the obvious difference in approach, we seek to learn the hierarchy as well and not engineer it through heuristics. We agree that it is an important feature of intelligent environment control systems to have specific rules for automation and not generalized ones which tend to not automate the environments correctly, the issue is on how to deal with environments with large state spaces in order to perform real-time calculations and decision tasks with the correct level of particularization. In addition there are a number of other intelligent environment initiatives. These include the Changing Places/House_n project at MIT which is focused on how technology, materials, and design strategies can create dynamic, evolving places that respond to the lives of their inhabitants [26]; the National Institute of Information and Communications Technology (NiCT) in Japan and their Keihanna Human Info-Communications Research Center which is focused on the development and testing of the Ubiquitous Home with a goal to support and optimize the usage of information appliances in the home across the users regardless of age or lifestyles [31]; the United Kingdom Equator Interdisciplinary Research Colloboration comprised of eight member colleges and universities is another supergroup of researchers working on pervasive computing and intelligent environment work focused on the integration of physical and digital interaction in order to bridge the gap between reality and virtual reality [12]; and the PRIMA project at INRIA which is concerned with the scientific foundation for interactive environments [21].

There are many other intelligent environment projects in academia, government, industry, and even amongst the enthusiastic general population and home hobbyists. The current trend is that many groups are combining their efforts to eliminate redundancy of effort and focus on the research challenges. These super efforts from groups such as Equator and AMIGO are beginning to produce measurable results and forward knowledge in intelligent environment research.

7 Conclusions

Overall, our approach, design, and experimentation provide a level of environmental automation for both virtual and real inhabitants from a data-driven automatically learned model that can adapt to user pattern changes over time. The key strength of our work is that the model does not require a human to create the model or for knowledge to be created in the system for the model to be generated. A minimal amount of knowledge is required to automate and adapt—namely the automatable actions. Our model is also not state restricted since we do not consider all possible states but just the states actually observed. Our data-driven model is only as good as the data that is used to generate it. The less consistent the inhabitant, the less ability there is to automate their life. Our techniques are also not very noise tolerant having difficulty discovering and identifying episodes in noisy sensor environments. The intelligent environment domain presents some very difficult problems, we have provide an approach with some success at automation and insight into the unique challenges ahead.

8 Future Work

We continue to improve our approach and experiment with inhabitants in our environments. Our immediate goals are focused on techniques to assist with resource consumption reduction and to handle sensor noise. We are also working more in-concert with the inhabitant to alleviate the control struggle and provide a more natural partnership between the inhabitant and the environment. We see our approach becoming a part of a system that works more closely with the inhabitant instead of just automating without regard. We are also learning how to better design sensor networks and environments to promote more easily learned (more consistent) patterns–developing a set of general design principles for building intelligent environments.

9 Acknowledgements

This work was supported by National Science Foundation grants IIS-0121297 and EIA-9820440.

References

[1] Accenture. Room of the Future, 2005. Website: `www.accenture.com/xd/xd.asp?it=enweb\&xd=services/technology/research/i%hs/room_future.xml`.

[2] AHRI. [AHRI] - Aware Home Research Initiative, Oct 2003. Website: `www.cc.gatech.edu/fce/ahri`.

[3] AIRE Group. MIT Project AIRE, March 2005. Website: `aire.csail.mit.edu/projects.shtml`.

[4] Ambient Intelligence research and Development Consortium. Ambient Intelligence for the Networked Home Environment, 2005. Website: `www.amigo-project.org`.

[5] E. Batschelet. *Circular Statistics in Biology*. Academic Press, 1981.

[6] A. Beresford. CUED: Laboratory for Communication Engineering, Oct 2003. Website: `www-lce.eng.cam.ac.uk/research/?view=2\&id=7`.

[7] *Telecare Overview*, 2005. Website: `www.btexact.com/research/researchprojects/currentresearch?doc=42834`.

[8] CISCO. The Internet Home, 2005. Website: www.cisco.com/warp/public/3/uk/ihome.

[9] D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. MavHome: An Agent-Based Smart Home. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*, pages 521–524, 2003.

[10] S. K. Das, D. J. Cook, A. Bhattacharya, E. O. Heierman, III, and T.-Y. Lin. The Role of Prediction Algorithms in the MavHome Smart Home Architecture. *IEEE Wireless Communications Special Issue on Smart Homes*, 9(6):77–84, 2002.

[11] *The Potential for 'Smart Home' systems in meeting the care needs of older persons and people with disabilities*, volume 10:1, 2000. Website: www.rgu.ac.uk/sss/research/page.cfm?pge=2546.

[12] Equator. Equator Website, 2005. Website: www.equator.ac.uk/index.php.

[13] S. Fine, Y. Singer, and N. Tishby. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1):41–62, 1998.

[14] K. Gopalratnam and D. J. Cook. Active LeZi: An Incremental Parsing Algorithm for Device Usage Prediction in the Smart Home. In *Proceedings of the Florida Artificial Intelligence Research Symposium*, pages 38–42, May 2003.

[15] *Creating an Ambient-Intelligence Environment Using Embedded Agents*, 2004.

[16] E. Heierman and D. J. Cook. Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns. In *Proceedings of the International Conference on Data Mining*, pages 537–540, 2003.

[17] E. O. Heierman. *Using Information-theoretic Principles to Discover Interesting Episodes in a Time-ordered Input Sequence*. PhD thesis, The University of Texas at Arlington, 2004.

[18] *The Gator Tech Smart House: A Programmable Pervasive Space*, March 2005.

[19] IETF Zeroconf Working Group. Zero Configuration Networking, 2005. Website: www.zeroconf.org.

[20] IIEG. Welcome to IIEG, 2005. Website: cswww.essex.ac.uk/intelligent-buildings.

[21] INRIA. PRIMA Project, 2005. Website: www.inria.fr/recherche/equipes_ur/prima.en.html.

[22] Intel Corporation. Digital Home Technologies for Aging in Place, 2005. Website: www.intel.com/research/exploratory/digital_home.htm.

[23] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. Technical Report CS-96-08, Brown University, Providence, RI, 1996.

[24] MARC. Smart In-Home Monitoring System, 2005. Website: marc.med.virginia.edu/projects_smarthomemonitor.html.

[25] Microsoft Research Vision Group. Easy Living, Oct 2003. Website: research.microsoft.com/easyliving.

[26] H. H. R. G. MIT Dept of Architecture House_n Research Group. MIT House_n, 2005. Website: architecture.mit.edu/house_n.

[27] MIT Media Lab. MIT Media Lab: Projects List Database, Oct 2003. Website: www.media.mit.edu/research/index.html.

[28] M. Mozer. The adaptive house. Website: www.cs.colorado.edu/˜mozer/house.

[29] M. Mozer. An Intelligent Environment must be Adaptive. *IEEE Intelligent Systems*, 14(2):11–13, March/April 1999.

[30] M. C. Mozer. *Smart Environments: Technology, Protocols, and Applications*, chapter Lessons from an Adaptive House, pages 273–294. J. Wiley & Sons, Hoboken, NJ, 2004.

[31] NiCT. The Ubiquitous Home, 2005. Website: www2.nict.go.jp/jt/a135/eng.

[32] K. E. Peterson. Home Sweet Ambient Home, from Philips, 2002. Website: www.10meters.com/homelab1.html.

[33] J. Rissanen. *Stochastic Complexity in Statistical inquiry*. World Scientific Publishing Company, 1989.

[34] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 1995.

[35] Siemens. Smart Homes, 2005. Website: www.siemens-industry.co.uk/main/business\%20groups/et/smart\%20homes.

[36] Stanford Interactivity Lab. Interactive Workspaces, Oct 2003. Website: iwork.stanford.edu.

[37] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.

[38] G. Theocharous, K. Rohanimanesh, and S. Mahadevan. Learning Hierarchical Partially Observable Markov Decision Processes for Robot Navigation, 2001. IEEE Conference on Robotics and Automation.

[39] UIUC Software Research Group. Gaia homepage, 2005. Website: gaia.cs.uiuc.edu.

[40] C. A. York. IBM's advanced PvC technology laboratory, Oct 2003. Website:www-106.ibm.com/developerworks/wireless/library/wi-pvc.