Seamlessly Engineering a Smart Environment *

Michael Youngblood, Diane J. Cook, Lawrence B. Holder
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019
{youngbld,cook,holder}@cse.uta.edu

Abstract

Developing technologies and systems for automated control of home and workplace environments is a challenging problem. We present a complete agent architecture for learning to automate a smart environment and discuss integration of AI and middleware technologies necessary to achieve the goals of this project. Results are demonstrated using the MavPad and MavLab intelligent environments.

1 Introduction

Since the beginning, people have lived in places that provide shelter and 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. The MavHome Project (Managing and Adaptive Versatile Home) is focused on providing such an environment. We define a smart environment as *one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience in that environment.* We take the viewpoint of treating an environment as an intelligent agent [1], which perceives the state of the environment using sensors and acts upon the environment using device controllers. The MavHome project goal is to develop and integrate components that will enable these intelligent environments to maximize the comfort of the inhabitants, minimize the consumption of resources, and maintain safety and security of the environment and its inhabitants.

Our work goes beyond just home and office environments to encompass all environments in which sensors can perceive the state of the environment, the system can reason about those observations, and actions can be taken to automate features of that environment. We conduct research in the MavLab, our workplace automated environment at UTA, and in an on-campus apartment called the MavPad which hosts a full-time student resident.

There are many intelligent environment projects producing valuable research with similar goals to our own. The Georgia Tech Aware Home is working on aspects of inhabitant localization, context-aware computing, and many HCI applications [5]. In a conference room setting, the Interactive Workspaces Project is exploring work collaboration technologies in technology-rich environments [2].

In a project similar to ours, the Adaptive Home at UC-Boulder utilizes a neural network to control the lighting, HVAC, and water temperature in a manner that minimizes operating cost [6]. The field of intelligent environment research has many niches. The MavHome Project is unique in that it focuses on the entire environment management and not just a single area of control, it strengthens AI techniques by using them in combination (e.g., data mining results improve prediction and automation), and is designed for long term usage and growth with the inhabitants.

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. This in turn will allow people to live in their homes longer and free time to allow people to focus on other aspects of their lives.

*This work is supported by National Science Foundation grants MRI-0115885 and IIS-0121297.

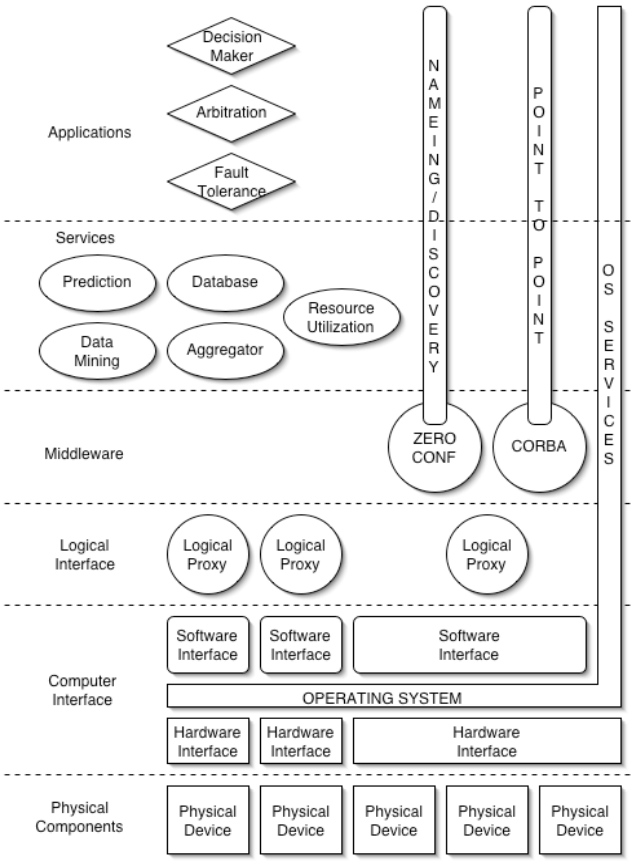

Figure 1: MavHome concrete architecture.

2 Software Architecture

The MavHome architecture 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, in order to create a robust and adaptive system.

The MavHome architecture shown in Figure 1 consists of cooperating layers. Perception is a bottom-up process. Sensors monitor the environment using physical components (e.g., sensors) and make information available through the interface layers. The database stores this information while other information components process the raw information into more useful knowledge (e.g., patterns, predictions). New information is presented to the decision making applications (top layer) upon request or by prior arrangement. Action execution flows top-down. The decision action is communicated to the services layer which records the action and communicates it to the physical components. The physical layer performs the action using powerline control, and other automated hardware, thus changing the state of the world and triggering a new perception.

All of the MavHome components are implemented and are being tested in two physical environments, the MavLab working environment and MavPad home environment. Powerline control automates all lights and appliances, as well as HVAC, fans, and miniblinds. Perception of light, humidity, temperature, smoke, gas, motion, and switch settings is performed through a sensor network developed in-house.

Communication between high-level components is performed using CORBA, and each component registers its presence using zero configuration (ZeroConf) technologies [4]. Implemented services include a PostgreSQL database that stores sensor readings, prediction and data mining software, and logical proxy aggregators. Resource utilization services monitor current utility consumption rates and provide usage estimates and consumption queries.

3 Learning to Automate the Smart Environment

To automate the environment, we collect observations of manual inhabitant activities and interactions with the environment. We then mine sequential patterns from this data using a sequence mining algorithm. Next, we predict the inhabitant's upcoming actions using observed historical data. Finally, a hierarchical Markov model is created using low-level state information and high-level sequential patterns, and is used to learn an action policy for the environment. Figure 2 shows how these components work together to improve the overall performance of the smart environment.

3.1 Mining Patterns Using ED

Our data mining algorithm, ED, is based on the idea of mining sequential patterns from time-ordered transactions. Typically, each inhabitant-home interaction event is characterized as a triple consisting of the device manipulated, the resulting change that occurred in that device, and the time of interaction. We move a window in a single pass through the history of events or inhabitant actions, looking for episodes (sequences) within the window that merit attention. The input sequence is partitioned into candidate episodes by moving a fixed-sized window over the input sequence and collecting episode descriptions as well as frequency information for each candidate. Candidate episodes are evaluated and the episodes with values above a minimum accept-

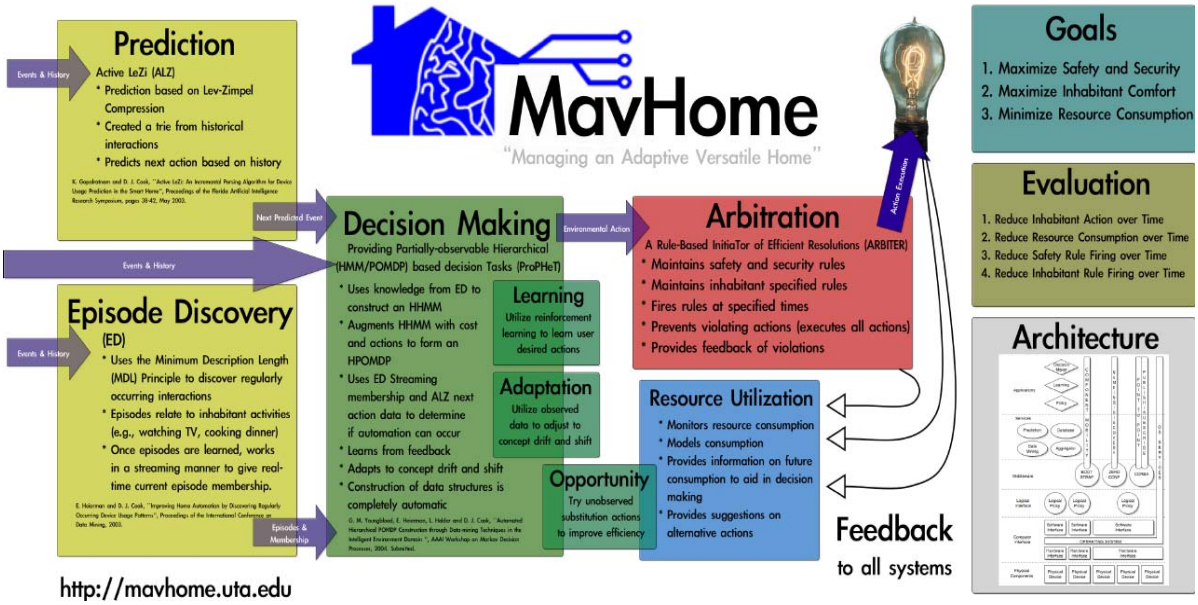

Figure 2: Integration of AI techniques into MavHome architecture.

able compression amount are reported. The window size is determined automatically according to the size that achieves the best compression performance over a sample of the input data.

To evaluate candidate episodes, ED uses the Minimum Description Length principle. The MDL principle targets patterns that can be used to minimize the description length of a database by replacing each instance of the pattern with a pointer to the pattern definition. Our MDL-based evaluation measure thus identifies patterns that balance frequency and length. Periodicity (daily, every other day, weekly occurrence) of episodes is detected using autocorrelation and included in the episode description. If the instances of a pattern are highly periodic (occur at predictable intervals), the exact timings do not need to be encoded and the resulting pattern yields even greater compression value.

In this way, ED identifies patterns of events that can be used to better understand the nature of inhabitant activity in the environment. As the following sections show, the results can also be used to enhance performance of predictors and decision makers that automate the environment.

3.2 Predicting Activities Using ALZ

To predict inhabitant activities, we borrow ideas from text compression, in this case the LZ78 compression algorithm [9]. By predicting inhabitant actions, the home can automate or improve upon anticipated events that inhabitants would normally perform in the home. Well-investigated text compression methods have established that good compression algorithms also make good predictors. According to information theory, a predictor with an order (size of history used) that grows at a rate approximating the entropy rate of the source is an optimal predictor. LZ78 incrementally parses the input sequence into phrases and stores them in a trie.

In our work, the input sequence consists of observed inhabitant actions. Our Active LeZi (ALZ) algorithm [3] enhances the LZ78 algorithm by recapturing information lost across phrase boundaries. Frequency of symbols is stored along with phrase information in a trie, and information from multiple context sizes are combined to provide the probability for each potential symbol, or inhabitant action, as being the next one to occur. As a result, ALZ gradually changes the order of the corresponding model that is used to predict the next symbol in the sequence.

In our experiments, ALZ proved to be a very accurate sequential predictor. However, accuracy is further improved when the task is restricted to only perform predictions when the current activity is part of a sequential pattern identified by ED. These patterns are more regular and predictable. Using one month of MavLab data collected with 6 student inhabitants, ALZ performance increased 14% when enhanced by ED mined results.

3.3 Decision Making Using ProPHeT

Work in decision-making under uncertainty has popularized the use of Partially Observable Markov Decision Processes. Recently, there have been many published hierarchical extensions that allow for the partitioning of large domains into a tree of manageable POMDPs [7, 8]. Although the Hierarchical POMDP is appropriate for an intelligent environment domain, current approaches generally require *a priori* construction of the HPOMDP. Given the large size of our domain, we need to seed our model with structure automatically derived from observed inhabitant activity data.

Unlike other approaches to creating a hierarchical model, our decision learner, ProPHeT, actually automates model creation by using the ED-mined sequences to represent the abstract nodes in the higher levels of the hierarchy. Lowest-level nodes correspond to an environment state representation together with an ALZ-supplied prediction of the next inhabitant action. To learn an automation strategy, the agent explores the effects of its decisions over time and uses this experience within a temporal-difference reinforcement learning framework to form control policies which optimize the expected future reward. The current version of MavHome receives negative reinforcement when the inhabitant immediately reverses an automation decision (e.g., turns the light back off) or an automation decision contradicts ARBITER-supplied safety and comfort constraints.

Before an action is executed it is checked against the policies in the policy engine, ARBITER. 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 actions such as turning the heater to 120^oF or from violating the inhabitant's stated wishes (e.g., a standing rule to never turn off the inhabitant's night light).

The vertical transition vector values between layers of the hierarchical POMDP are assigned using episode occurrence information supplied by ED. Horizontal transition matrix data between abstract nodes on the same level is captured by repeating the episode discovery process on the discovered episodes on each level in order to learn the abstractions of the next higher level. This can be repeated until no abstractions for a level are found, in which case this is the root level. Each abstract state is partially represented by the observation sequences it contains in its child nodes. Due to overlap in these observation sequences between parent abstract nodes, abstract nodes can be grouped into hierarchies. After this process a n-tier hierarchical model is automatically created from learned data. An example hierarchical model constructed from MavHome test data is shown on the left in Figure 3.

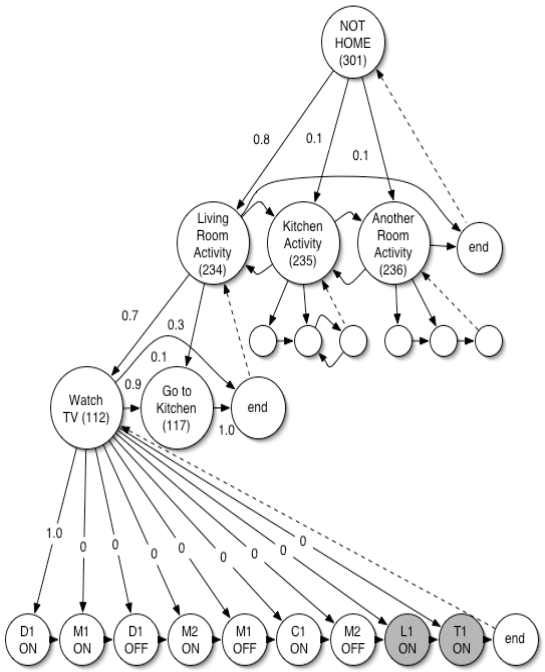

Figure 3: Hierarchical model built from MavHome data.

4 Methods for Evaluation

MavHome systems have the goal of maximizing the comfort of the inhabitants, minimizing the consumption of resources, and maintaining safety and security. We have established metrics to measure the accomplishment of these goals. Inhabitant comfort is measured by the number of inhabitant interactions performed with devices that can be automated. The system should continually strive to reduce the number of inhabitant-initiated interactions. Resource consumption is measured by the utility metering of the environment. The utility consumption should decrease if the system is properly minimizing resource consumption. Safety and security are measured by the number of safety interventions from the policy engine. The system should minimize the number of policy engine interventions.

All of the algorithms described here are implemented in MavHome and are being used to automate two environments, shown in Figure 4. The MavLab environment contains work areas, cubicles, a break

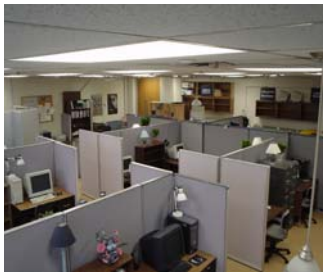

Figure 4: The MavLab (left) and MavPad (right) environments.

Figure 5: MavLab composed of zeroconf objects.

area, a lounge, and a conference room. MavLab is automated using 54 X-10 controllers and the current state is determined using light, temperature, humidity, motion, and door/seat status sensors. The MavPad is an on-campus apartment hosting a full-time student occupant. MavPad is automated using 25 controllers and provides sensing for light, temperature, humidity, leak detection, vent position, smoke detection, CO detection, motion, and door/window/seat status sensors.

A ResiSim 3D simulator has been developed that simulates the environment layout and inhabitant activities. Many current paradigms do not lend themselves to simulation of dynamic physical environments. Changes in environments are usually preplanned. For our research, a simulator is useful because data can be generated from the simulator and used to build models for automation. Similarly, collected real data can be visualized in the simulator and potential automation strategies can be viewed and evaluated.

We simulate our environments as a discrete event simulation with dozens of objects. Each object should be a self-contained, accurate simulation of the item it represents and be seamlessly replaceable by the actual object, bridging the gap between reality and virtual reality. As in the real-world environment, objects can be introduced or removed at any time.

Using ResiSim, representative inhabitant scenarios can be created and tested for automation performance in order to improve the underlying MavHome algorithms. The interface to the simulator is identical to the physical environment, so that either environment can be controlled by the learning algorithms. Figure 5 shows a ResiSim version of MavLab, and Figure 6 shows a ResiSim simulation of MavPad. Each simulation object is modeled as a ZeroConf entity. As a result, objects can notify the simulator of changes in their status without interrupting the execution flow.

Figure 6: MavPab composed of zeroconf objects.

ResiSim consists of three main parts: logical proxy objects, core simulation objects, and user interface objects. The logical proxies represent a physical object in the real world (e.g., lamp or chair). The core simulation objects include zeroconf-enabled virtual inhabitant dynamic objects, simulation objects, interface objects, and an observation object. A user interface object provides an interactive environment to the user as seen in Figures 5 and 6. The interface can communicate with the ResiSim server over the Internet, which requires minimal configuration to locate the server.

5 Results from Case Study

As an illustration of these techniques, we have evaluated a week in an inhabitant's life with the goal of reducing the manual interactions in the MavLab. The data was generated from a 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. We trained ALZ and ED on real data and then repeated a typical week in our ResiSim simulator to determine if the system could automate the lights throughout the day in real-time.

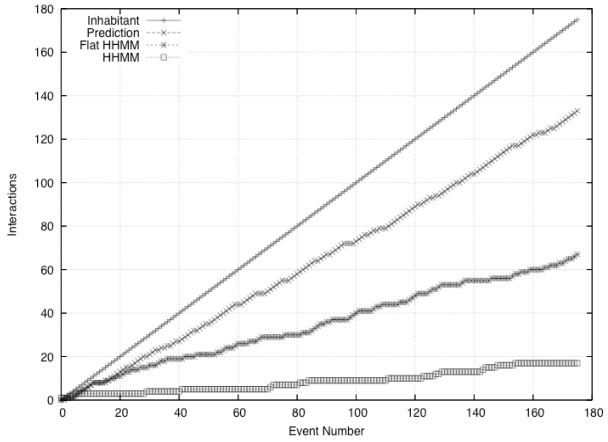

Figure 7: Interaction reduction.

ALZ processed the data and converged to 99.99% accuracy after 10 iterations through the training data, and accuracy was 54% on test data. When automation decisions were made using ALZ alone, interactions were reduced by 9.7% on average. Next, ED processed the data and found 3 episodes to use as abstract nodes in the HPOMDP. The hierarchical Markov model with no abstract nodes reduced interactions by 38.3%, and the combined-learning system (ProPHeT bootstrapped using ED and ALZ) was able to reduce interactions by 76%, as shown in Figure 7.

Experimentation in the MavPad using real inhabitant data has yielded similar results. In this case, ALZ alone reduced interactions from 18 to 17 events, the HPOMDP with no abstract nodes reduced interactions by 33.3% to 12 events, while the bootstrapped HPOMDP reduced interactions by 72.2% to 5 events.

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. Problems in the automation decisions appear around interactions that occur within a short timeframe and are currently under investigation.

6 Conclusions

In this research we have shown that with careful systems engineering, AI algorithms can work together to accomplish a large task such as automating an intelligent environment. We are currently enhancing our work to adapt to multiple inhabitants and accommodate a greater range of environments and automation tasks.

References

[1] S K Das, D J Cook, A Bhattacharya, E O Heierman, and T-Y Lin. The role of prediction algorithms in the mavhome smart home architecture. *IEEE Wireless Communications*, 9(6):77–84, 2002.

[2] Armando Fox, Brad Johanson, Pat Hanrahan, and Terry Winograd. Integrating information appliances into an interactive space. *IEEE Computer Graphics and Applications*, 20(3):54–65, 2000.

[3] K Gopalratnam and D J Cook. Online sequential prediction via incremental parsing: The Active LeZi algorithm. *IEEE Intelligent Systems*, 2005.

[4] IETF Zeroconf Working Group. Zero configuration networking, 2005. http://www.zeroconf.org.

[5] Cory D Kidd, Robert J Orr, Gregory D Abowd, Christopher G Atkeson, Irfan A Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings*, 1999.

[6] Michael Mozer. An intelligent environment must be adaptive. *IEEE Intelligent Systems and their Applications*, 14(2):11–13, 1999.

[7] J. Pineau, N. Roy, and S. Thrun. A Hierarchical Approach to POMDP Planning and Execution, 2001. Workshop on Hierarchy and Memory in Reinforcement Learning (ICML).

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

[9] J Ziv and A Lempel. Compression of individual sequences via variable rate coding. *IEEE Transactions on Information Theory*, IT-24:530–536, 1978.