

# A Learning Architecture for Automating the Intelligent Environment

G. Michael Youngblood, Diane J. Cook, and Lawrence B. Holder

Department of Computer Science & Engineering  
The University of Texas at Arlington  
Arlington, Texas 76019-0015  
{youngbld, cook, holder}@cse.uta.edu

## Abstract

Developing technologies and systems for perception and perspicacious automated control of home and workplace environments is a challenging problem. We present a complete agent architecture for learning to automate the 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 by 72.2%.

## 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. These new intelligent environments need to go beyond the basics and assist us in our everyday journey through life. The MavHome Project (Managing and Adaptive Versatile Home) is focused on conducting just such research in the smart home technologies of tomorrow. We take the viewpoint of treating an environment as an intelligent agent (Das *et al.* 2002). We seek to develop and integrate components that will enable these intelligent environments of the future where the goals of these environments are to maximize the comfort of the inhabitants, minimize the consumption of resources, and maintain safety and security.

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

There are many intelligent environment projects producing valuable research. The Georgia Tech Aware Home (AHRI 2003; Salber, Dey, & Abowd 1999; Abowd, Battesini, & O'Connell 2002), the AIRE group at the MIT AI Lab (AIRE Group 2004), the Interactive Workspaces project at Stanford University (Stanford Interactivity Lab 2003), and

the Gaia project at UIUC (Gaia Project 2004) all focus on areas of infrastructure, localization, context-aware computing, and HCI work for intelligent home and workplaces. The Adaptive Home at UC-Boulder utilizes reinforcement learning and neural networks to control the lighting, HVAC, and water temperature in a manner that minimizes operating cost (Mozer 1999). 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 utilizes advanced AI techniques in novel ways (e.g., seeding HHMMs with Data Mining techniques), 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. 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.

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 initial experimental results, and discuss on-going case studies.

## 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, potentially even hot-swappable, in order to create a robust and adaptive system. We present the architecture first in a functional abstract view and then in a detailed concrete form.

The MavHome abstract architecture 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, process mobility, 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, 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, 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 vacuous). 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.

The abstract layers of the MavHome architecture are realized through a set of concrete functional layers. These concrete layers are shown with some example components in Figure 1. 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, input devices, 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 commu-

nication and service discovery. 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.

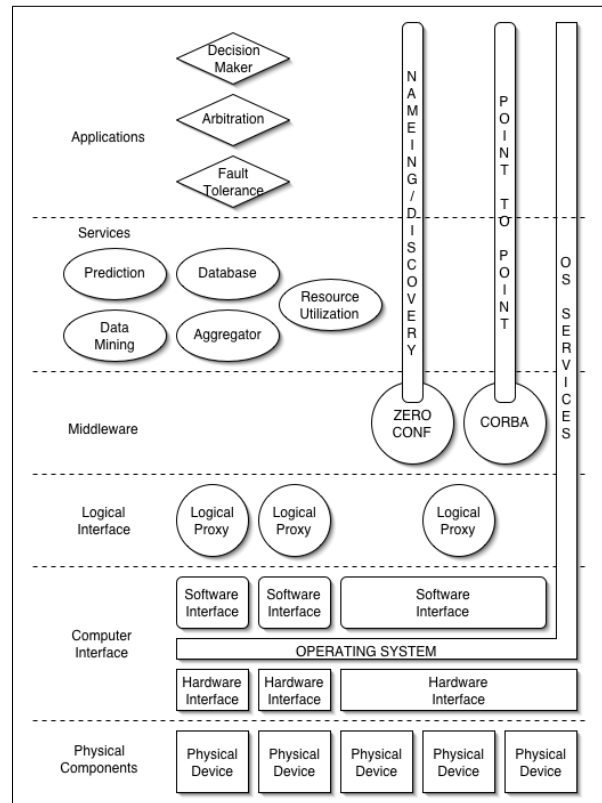


Figure 1: MavHome concrete architecture.

## Implementation and Deployment

The MavHome architecture has been developed and deployed over the last 3 years in the MavLab and over the last year and a half in the MavPad.

Lighting control is the largest effector in most intelligent environments. We currently use X10-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. X10 was chosen because of its availability and low price. Many home users also utilize X10 technology, so immediate benefits to the current home user are possible.

Perception through light, humidity, temperature, 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

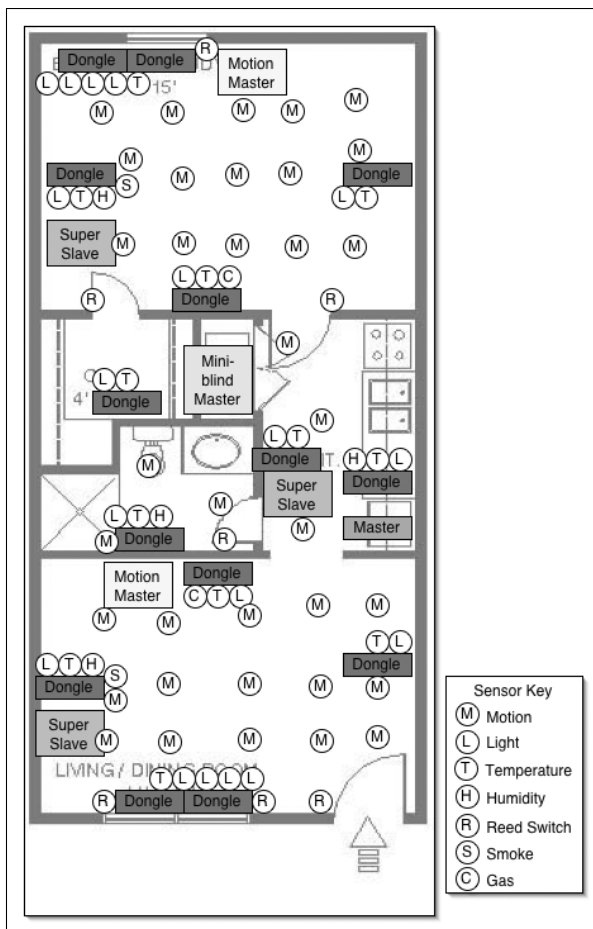


Figure 2: MavPad sensors.

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 4 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 4 mini-blinds. Figure 2 shows the MavPad sensor layout.

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 10 feet from the ground. In order to reduce the sensing area, tubes were placed over the sensors to reduce the floor footprint to a 4 foot sensing circle. Tests in the MavLab show a consistent single inhabitant location detection rate of 95% or better accuracy. Multiple inhabitant studies will require augmenting technology, so our focus is on single inhabitants.

The logical interfaces for all X10 and Argus based components have been written as light-weight configurable modules. The proxies maintain the current state of each device and provide a mechanism for reading and, if applicable, control. The communication protocols for X10 devices and Ar-

gus components are well defined and interface availability is advertised through zero configuration (ZeroConf) technology.

Components desiring to find X10 or Argus components merely need to perform a link-local query for devices that follow the defined MavHome X10 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 shutdown. We have had a high level of success using ZeroConf technology with very few problems once the components are 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.

MavHome 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 for replacing the CORBA naming service and utilizing service discovery are provided by the Apple Multicast DNS responder and adherence to the ZeroConf standard.

## Services

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.

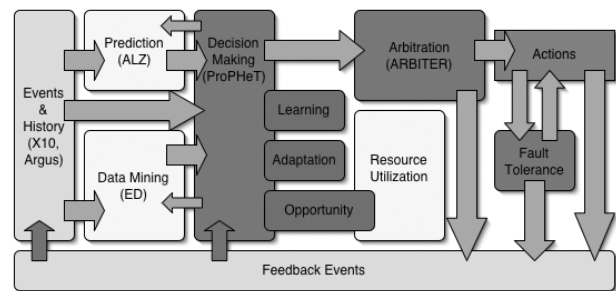


Figure 3: MavCore architecture.

**Prediction** 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 (Gopalratnam & Cook 2003) 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.

In our evaluation and usage of Active-LeZi (ALZ) in the MavLab, we have achieved 100% predictive accuracy on 30 days of real data (200 events/day, no noise) for single inhabitant patterns. The MavHome systems first train the algorithm and then use the predictions in real time. When predictive accuracy drops below a defined threshold, the predictor is retrained over data from a configurable sliding window to adapt for concept drift in the changing patterns of inhabitants.

**Data Mining** The MavHome 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. After discovering the episodes, it is also desirable to be able to classify streamed observations to episodes in real time with the same service.

MavHome uses the *Episode Discovery* (ED) algorithm (Heierman & Cook 2003) 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 (Rissanen 1989) 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 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 reevaluate 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.

## Applications

The application layer components learn, make the decisions, and follow set policies on safety and security. They also maintain a level of fault detection and correction as well as providing valuable feedback to the system. The MavCore architecture illustrating the main services and application layer components is shown in Figure 3.

**Decision Making** Decision making is performed in the ProPHeT (**Providing Partially-observable Hierarchical (HMM/POMDP) based decision Tasks**) component. The world representation at this level is the Hierarchical Hidden Markov Model (HHMM) (Fine, Singer, & Tishby 1998) based upon a hierarchy of episodes of activity mined from stored observations. As review, an HHMM is defined as a set of states  $S$ , which include *production* states (leaves) that produce observations, *abstract* states which are hidden states (unobservable) representing entire stochastic processes, and *end* (child) states which return control to a parent node; a horizontal transition matrix  $T$ , mapping the probability of transition between child nodes of the same parent; a vertical transition vector  $\Pi$ , that assigns the probability of transition from an internal node to its child nodes; a set of observations  $Z$ ; and a mapping of a distribution probability set of observations in each product state  $O$ . For detailed discussions refer to (Fine, Singer, & Tishby 1998; Theocharous, Rohanimanesh, & Mahadevan 2001).

From a collection of data that contains many repetitive patterns, data-mining techniques can automatically discover these patterns and provide statistical data on pattern permutations within a given set of members and over the entire data set. This information can be utilized to create abstract states from the identified patterns and production states from the pattern sequences.

The vertical transition vector values between abstract nodes from the root to the episodes is assigned from episode occurrence information from ED. Horizontal transition matrix data between abstract nodes of 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 HHMM is automatically created



The data was restricted to just motion and lighting interactions which account for an average of 10,310 events per day. There are on average 18 lighting device interactions a day with the remainder being motion information. Using our ResiSim (Residential Simulator) 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 5. 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 4 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 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 5.

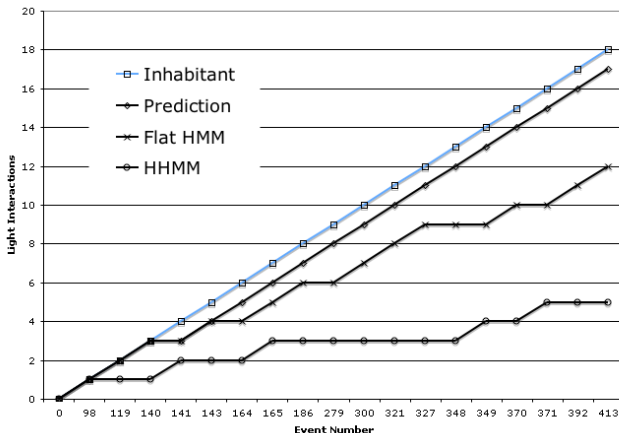


Figure 5: Interaction reduction.

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

## Future Work

Current experimentation continues in the MavPad and MavLab. Our goal is to strengthen our two main contributions of this work: the data-driven automated construc-

tion of a HHMM and the dynamic incremental updating of the structure through learning and additional data-driven changes for the intelligent environment to continually adapt and learn with the inhabitants. We are continuing to refine our work by evaluating increasingly complex patterns, improving automation usage, expanding the sensor resolution, and are investigating reinforcement learning techniques in order to improve and adapt our systems automatically over time.

## Acknowledgements

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

## References

- Abowd, G. D.; Battestini, A.; and O'Connell, T. 2002. The Location Service: A Framework for Handling Multiple Location Sensing Technologies.
- AHRI. 2003. [AHRI] - Aware Home Research Initiative.
- AIRE Group. 2004. MIT Project AIRE – About Us.
- Das, S. K.; Cook, D. J.; Bhattacharya, A.; III, E. O. H.; and Lin, T.-Y. 2002. The Role of Prediction Algorithms in the MavHome Smart Home Architecture. *IEEE Wireless Communications Special Issue on Smart Homes* 9(6):77–84.
- Fine, S.; Singer, Y.; and Tishby, N. 1998. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning* 32(1):41–62.
- Gaia Project. 2004. Gaia homepage. <http://gaia.cs.uiuc.edu>.
- Gopalratnam, K., and Cook, D. J. 2003. Active LeZi: An Incremental Parsing Algorithm for Device Usage Prediction in the Smart Home. In *Proceedings of the Florida Artificial Intelligence Research Symposium*, 38–42.
- Heierman, E., and Cook, D. J. 2003. Improving Home Automation by Discovering Regularly Occurring Device Usage Patterns. In *Proceedings of the International Conference on Data Mining*.
- Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1996. Planning and Acting in Partially Observable Stochastic Domains. Technical Report CS-96-08, Brown University, Providence, RI.
- Mozer, M. 1999. An Intelligent Environment must be Adaptive. *IEEE Intelligent Systems* 14(2):11–13.
- Rissanen, J. 1989. *Stochastic Complexity in Statistical inquiry*. World Scientific Publishing Company.
- Salber, D.; Dey, A. K.; and Abowd, G. D. 1999. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, 434–441.
- Stanford Interactivity Lab. 2003. Interactive Workspaces.
- Theocharous, G.; Rohanimanesh, K.; and Mahadevan, S. 2001. Learning Hierarchical Partially Observable Markov Decision Processes for Robot Navigation. *IEEE Conference on Robotics and Automation*.