Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX 76019

The MavHome Architecture

G. Michael Youngblood, Diane J. Cook,
and Lawrence B. Holder
{youngbld, cook, holder}@cse.uta.edu

Technical Report CSE-2004-18

The MavHome Architecture

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

Introduction

The MavHome Project (Managing and Adaptive Versatile Home) is focused on conducting research in smart home technologies from the aspect of treating an environment as an intelligent agent (Das *et al.* 2002). 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.

The MavHome Project goes beyond just a home environment and encompasses all environments in which sensors can perceive observations, the system can reason about those observations, and actions can be taken to automate features of that environment. We currently conduct research in the MavLab which provides an office type setting and contains the MavDen as shown in Figure 1 and the MavKitchen. We also are establishing a multiple-month experiment with single and multiple inhabitants under observation in an on-campus apartment called the MavPad.

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 (AHRI 2003; Salber, Dey, & Abowd 1999; Abowd, Battestini, & O'Connell 2002). The AIRE group at the MIT AI Lab is engaged in research involving pervasive computing designs and people-centic applications and have constructed "AIRE spaces" in the forms of an intelligent conference room, intelligent workspaces, kiosks, and "oxgenated" offices (AIRE Group 2004). At Stanford University, the Interactive Workspaces Project is exploring work collaboration technologies in technology-rich environments with a focus on task-oriented work such as design reviews or brainstorming sessions. Their experimental facility is called "iRoom" where they are investigating integration issues with multiple-device, multiple user applications, interaction technologies, deployment software, and component integration (Stanford Interactivity Lab 2003). 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 (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.

Figure 1: MavDen in the UTA MavLab.

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, 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.

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 our preparations for case studies. Section 2 starts by describing the MavHome architecture in an abstract form and then discusses the concrete layers that make the system work. Section 3 describes the specifics of our implementation and current deployment. Section 4 presents our evaluation criteria, and section 5 reveals some of our preliminary results.

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.

Abstract Layers

The MavHome architecture shown in Figure 2 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 from the other layers. 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, voice synthesis and recognition 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). 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

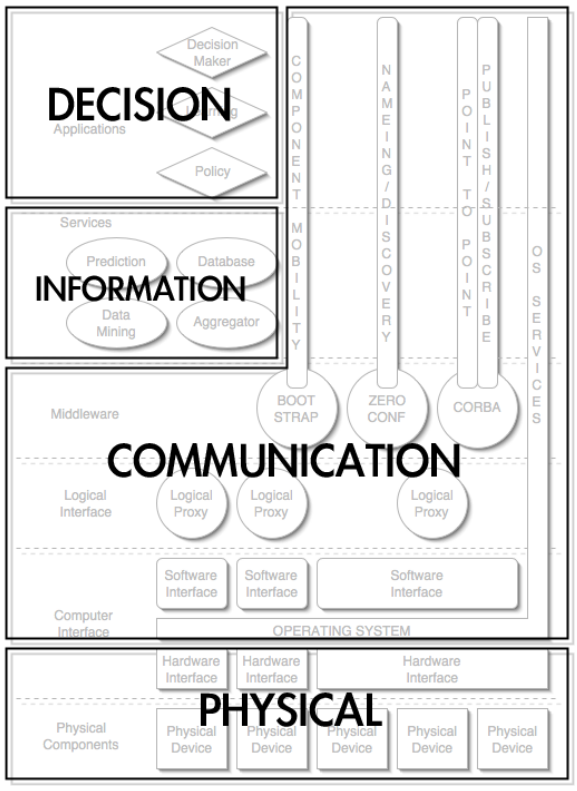

Figure 2: MavHome Abstract Architecture.

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.

Concrete Layers

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 3. 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, touch screens, gesture input devices, cameras, and so forth, with the exception of the computer with which equipment is interfaced. The physical computer(s) 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 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 will become more complex. The *Middleware* layer inserts valuable services to the upper layers of the architecture to facilitate communication, process mobility, and service discovery. The MavHome architecture specifies middleware that provides both point-to-point and publish-subscribe types of communication, naming/service discovery provisions, and a mechanism to move system components between physical computing hardware devices. 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.

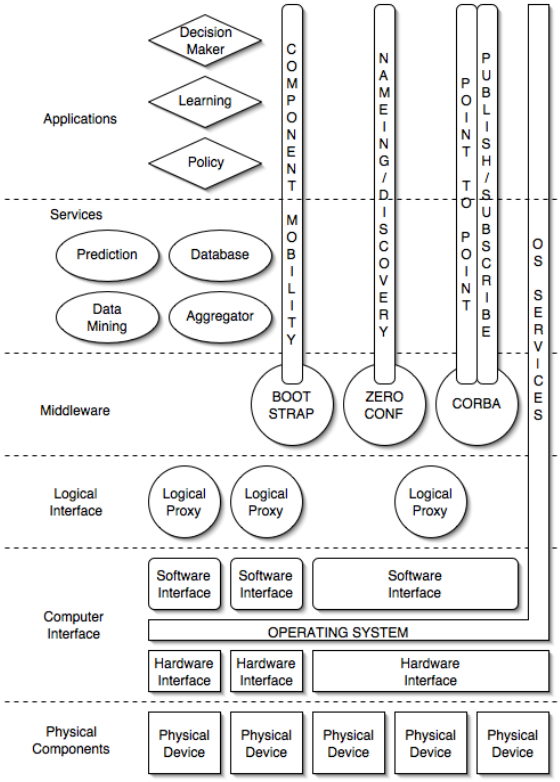

Figure 3: MavHome Concrete Architecture.

Implementation and Deployment

Pieces of the MavHome architecture have been developed and deployed over the last 2 years in the MavLab. We currently have all of the components in the MavHome architecture in some form and are working to fully integrate and improve all of the components to improve the system.

Physical Components

Lighting control is the largest effector in most intelligent environments. We currently use X10-based (X10 Corporation 2004) 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. The MavLab is partitioned into 10 sections {A-J} each containing approximately 6 lamps and a controlling computer. 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.

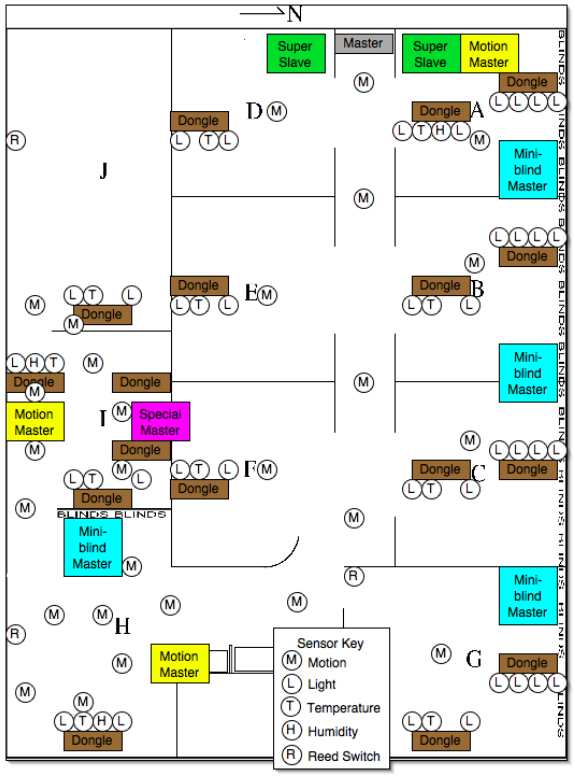

Figure 4: MavLab Sensors.

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 interfaces to the computer via a serial interface and connects up to 99 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 4 shows the MavLab sensor layout.

A key element in perception is inhabitant localization. The Argus Motion Master is used in conjunction with passive infrared (PIR) sensors placed on the ceiling in traffic areas to detect motion. Figure 5 shows part of the motion

sensor network in the MavDen–each detector is shaded and the closeup is on a single sensor. 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 MavDen show a single inhabitant location detection rate of 95% accuracy. Multiple inhabitant studies will require augmenting technology.

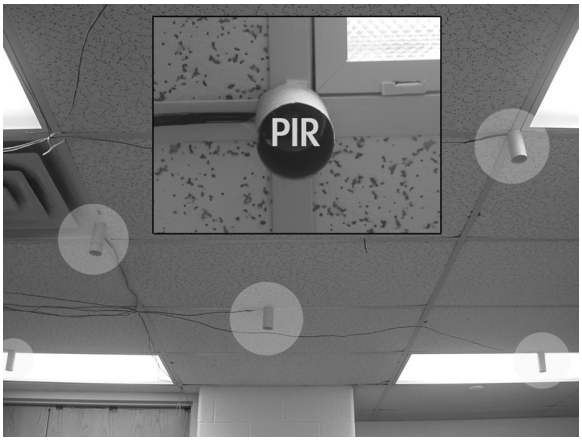

Figure 5: Passive Infrared Motion Network.

Computer Interface

All MavHome components interface through either serial, USB, or firewire interfaces. New PCs rarely come with more than a single 9-pin serial port, so many of our systems have been outfitted with additional serial interface cards. The MavHome architecture and components have been developed on Intel based PCs (Pentium 4) and use Red Hat Linux 9.0 with the stock 2.4.20-8 kernel.

Logical Interface

The logical interfaces for all X10 and Argus based components have been written as light-weight configurable modules with both shared memory and UDP socket-based interfaces. Shared memory is used for local components and the socket interface for remote components. The proxies are written in C++. 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 Argus components are well defined and interface availability is advertised through zero configuration technology (Cheshire 2004) using a multi-cast DNS responder (a.k.a., Rendezvous-enabled).

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.

Middleware

MavHome uses three main middleware packages. Communication between high level components is performed using the Common Object Request Broker Architecture (CORBA) (Object Management Group 2004) 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++ and Java implemented components. OmniORB is used for point-to-point communication and OmniEvents for publish-subscribe (Grisby 2004). Zero configuration technologies for replacing the CORBA naming service and utilizing service discovery are provided by the Howl (Porchdog Software 2004) multi-cast DNS responder and adherence to the ZEROCONF standard (Cheshire 2004). Component mobility is provided by an in-house developed daemon system called Boot Strap. Boot Strap provides a password secure RPC mechanism for starting and stopping processes on remote machines. In conjunction with the requirement that all services and applications have the ability to checkpoint and restore their data and state, a shared NFS system, and the Boot Strap daemon, component mobility is easily achieved.

Services

Implemented services include a PostgreSQL (PostgreSQL Global Development Group 2004) 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). Other services such as speech recognition/synthesis are also currently being developed.

Figure 6: Mavigator Interface.

Interfaces There are a number of interaction mechanisms utilized in our work. Inhabitants can interact with devices through RF remotes, their own motion and actions (e.g., opening doors, sitting on a chair), or through more advanced interfaces. Figure 6 shows our PDA-based interface to the MavLab, called the Mavigator. Mavigator is Macromedia

Flash-based and is displayed on a mobile web browser. A back-end XML-based service handles the interface and interacts with logical proxies to control devices and query information. Another advanced interface is shown in Figure 7. The MavKitchen interface uses a gesture pad and down-projecting image to interact with kitchen inhabitants. The MavKitchen interface is also Flash-based with an XML back-end service.

Figure 7: MavKitchen Interface.

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 in the MavLab, we have achieved 100% predictive accuracy on 30 days of real data for single scenarios. 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 a matter of continued study. Currently, we use a period of 30 days.

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. 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. Learning is performed by extending the HHMM to a hierarchical Partially Observable Markov Decision Process (POMDP) and applying a hierarchical Baum-Welch algorithm (Theocharous, Rohanimanesh, & Mahadevan 2001). Action decisions are made by using the streaming episode membership features of ED to provide input into the current belief state in the HHMM and the Active-LeZi prediction of the next event to chose the appropriate transitional action in the HHMM. Before the action is executed it is checked against the policies in the policy engine. 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° F or from violating the inhabitant's stated wishes (e.g., a standing

order to never turn off the inhabitant's night light).

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.

Experimentation and Preliminary Results

Recently, we conducted some initial trials in the MavDen using the MavHome architecture to determine if the system could learn to automate devices. Data containing three patterns (watching TV, listening to the CD player, and reading) was generated in the MavDen environment and extrapolated to 30 days, patterns occurring at random times during the day. Figure 8 shows the monitor display of the MavDen with a motion sensor localization block showing the inhabitant's location. Each pattern was unique and included an opportunity for the system to automate one or more devices. After setup and training the system was randomly tested in real time with a single inhabitant for the three patterns four times each (12 cases) over a two hour period. The system successfully automated devices in 10 out of 12 episodes (83.3%). Failures occurred due to some minor sensor noise causing the patterns to appear different than the system training data. Additional trials and better training data should fix the errors. This initial trial shows promise for the design of the MavHome architecture. Plans are in progress to fully implement our sensor networks and start more advanced, fully implemented, and integrated system case studies in the MavLab, MavDen, MavPad, and MavKitchen.

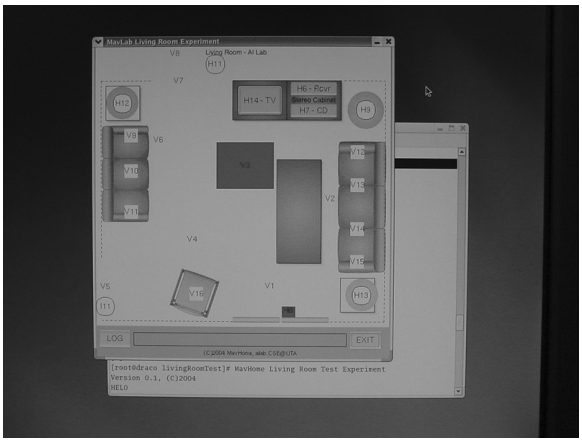

Figure 8: Sensor Interaction Test Display.

Acknowledgements

This work was supported by National Science Foundation grants IIS-0121297 and EIA-9820440. Thank you to Ryan Duryea (TCU) for his work on the initial Mavigator.

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.

Cheshire, S. 2004. Zero Configuration Networking (Zeroconf).

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. http://citeseer.nj.nec.com/fine98hierarchical.html.

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.

Grisby, D. 2004. omniORB.

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*.

Mozer, M. 1999. An Intelligent Environment must be Adaptive. *IEEE Intelligent Systems* 14(2):11–13.

Object Management Group. 2004. Object Managemnt Group.

Porchdog Software. 2004. Howl. http://www.porchdogsoft.com/products/index.html.

PostgreSQL Global Development Group. 2004. PostgreSQL. http://www.postgresql.org/.

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. http://citeseer.nj.nec.com/theocharous01learning.html.

X10 Corporation. 2004. X10.com. http://www.x10.com.