A Distributed Multi-Agent Framework for Intelligent Environments

Charles HANNON

and

Lisa BURNELL

Crescent Lab for Intelligent Systems
Department of Computer Science
Texas Christian University (TCU)
Fort Worth, Texas 76129, USA

ABSTRACT

Biologically inspired models (BIM), i.e. complex models for organization and communication based on biological systems, consider the heterogeneous and dynamic nature to which entities must adapt. This makes such models appropriate for the design and implementation of smart home systems. The Gold Seekers Project, a BIM developed by one of the authors, has been used to study the mechanisms for realizing intelligence in natural and artificial systems. Two of the project components, Goal Mind and Alchemy, provide a framework in which to develop distributed multi-agent architectures. Goal Mind is used to define the application architecture, including agent structure and communications needs. Reasoning within the agents is multi-modal, with current support for rules, semantic networks, neural networks, and procedures. Alchemy provides the mechanism to distribute the system. We describe Goal Mind and Alchemy and the smart home application being created with these components.

Keywords: distributed multi-agent systems, intelligent environments, biologically inspired computing models, smart homes.

INTRODUCTION

To achieve the "Pervasive Intelligence" [1] needed for intelligent environments, such as smart homes, new approaches are needed in the organization and deployment of systems. Ethnographic studies [8] and other reports [7] clearly reveal that just because a smart home technology is created does not mean that it will be widely adopted. Technologies must provide a clear benefit, fit into the social culture and architecture of the home, be aesthetically acceptable, and consistently work properly.

Additionally, the complexity of the sensors, actuators, and control systems within even a modest smart home environment will require a multi-vendor solution where the system must be forged from conflicting and overlapping standards. This requires the control systems to adapt to not only inhabitants' behavior, but also the availability and utility of its interfaces to the environment.

One way to realize such adaptive control is to use an agent-based strategy that relies on something like a servant analogy. Since such an analogy is hierarchical, these agents can be designed to adapt to each other as well as the environment.

Like the best human workers, our smart home agents must arrive with general knowledge about what work needs to be done, and how to do it. They must be able to make decisions based on this knowledge and the context or environment. They must know when to take the initiative to perform a task and when to ask for assistance. They must be able to continue their work in spite of missing information or other resources. And finally, they must be able to learn the goals, needs, and preferences of the particular beings they serve and adapt their efforts based on this knowledge.

There are different tasks and priorities inhabitants will desire from a smart home. One household may just want to easily view media programs on demand without having to search large program guides. Another may need help in managing household tasks related to maintenance and other routine chores, like keeping the kitchen pantry stocked. A caregiver may desire a system to aid in the supervision of an elderly parent. Occasionally perhaps, a household may desire every possible feature available. Even then, the acquisition of components may occur over an extended period of time and usage may fluctuate as the inhabitant's interests and other activities change.

Biologically inspired models (BIM), i.e. complex models for organization and communication based on biological systems, consider the heterogeneous and dynamic nature to which entities must adapt. This makes such models appropriate for the design and implementation of smart home systems. In this article, we begin by describing a BIM development framework that has been used for both the scientific study of natural intelligence and for developing complex applications systems. Next, we show how this framework is being used to iteratively develop major portions of a distributed, multi-agent smart home system. Following an overview of the smart kitchen agent and event handling in the system, we conclude with current status of the system and future work.

GOLD SEEKERS

The Gold Seekers Project aims to study the mechanisms for realizing biologically-inspired reasoning methods in natural and artificial systems. The relationship of this project to other cognitive and computer science research is provided in Figure 1.

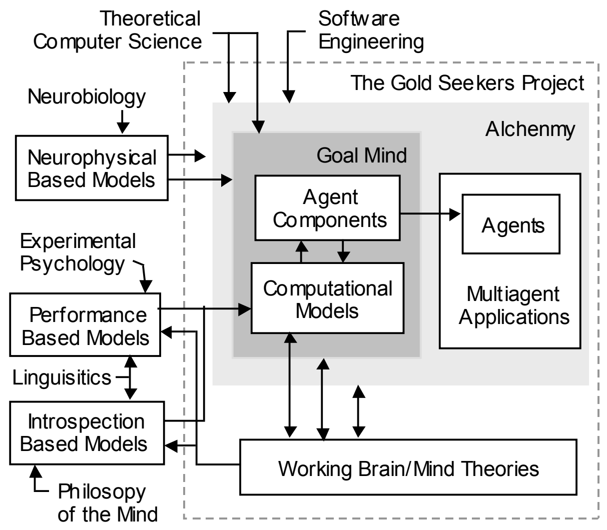

Figure 1. Computational models are used to produce the agent components making up agents within a multiagent application. Components' design, construction, testing and operation are supported by Goal Mind. The distribution, migration and control of component processes and the resulting agent multi-processes across multiple processors are supported by Alchemy.

The Project seeks to allow a direct connection between research in cognitive modeling, parallel/distributed processing, and multi-agent architectures; and thus, allow the direct use of cognitive task fusion for application development, and improvement of the component models by exploring their utility in solving to real-world problems. To support its goal, Gold Seekers provides a set of tools for both general distributed application and multi-agent intelligent system design and implementation.

The two main tools of Gold Seekers are Alchemy and Goal Mind. Alchemy is a distributed processing environment which supports: 1) the asynchronous processing model needed by our cognitive-based approach, 2) a GUI-driven dynamic generation, operation and testing environment, and 3) a multi-level security facility for safe operation over the Internet or other public networks [3, 4]. Goal Mind is the next generation of our AMEBA architecture [5]. While AMEBA used an integrated processing model, Gold Mind is built on top of Alchemy.

Based on a brain model, Alchemy architecture divides an application into a set of asynchronous processing elements that process messages received from other processing elements via either a client-side or server-side connection. Being inherently distributed, concurrency is the only mechanism by which an application can be implemented. Each connection instance in an Alchemy application runs in its own process thread within a heavyweight container called a node.

Alchemy is totally dynamic. Each node can support any number of server and client connections and any number of message handlers that perform the event-driven processing of the application. Thus, the number of nodes in an application, the number of server and client connections maintained by a node, and the mapping between handlers and connections can all be changed at any time during the application's life.

Goal Mind attempts to capture the explanatory force of a connectionist neural model while allowing the use of the better-understood representation and reasoning methods of symbolic AI [6]. Goal Mind models draw their explanatory depth from the environment's ability to support hierarchical cognitive processing. Using adaptive distributed processing and generalized inter-process communication, cognitive functions can be modeled at different levels of abstractions without changing the logical relationship between these functions. Thus, a function like the conceptual reasoning about the world and self can be simulated with a reasoning and knowledge storage system that has far less capacity than that of a real human. This allows us to preserve the overall model's explanatory depth, as long as we preserve explanatory relationships between cognitive components.

From a system perspective, Gold Mind provides processor transparency within a parallel system and a flexible method of process and knowledge management. The key element that supports these requirements is the etheron which supports: 1) a standard way to load and store knowledge, 2) interfaces to a set of predefined management tools and 3) a generalized set of communication channels for talking with other etherons. Since Goal Mind applications runs within the Alchemy environment, the number of etherons and the way they are connected can be dynamically reconfigured; and thus, the application can completely change its functionality while running. Since, Alchemy supports multiple applications running at the same time, a Gold Mind application can similarly affect other executing Gold Mind applications.

Applications built using Gold Seekers fall into two basic groups, those that are primarily designed to explore cognition and those that attempt to use the resulting knowledge of this exploration to address real-world problems. Using both Alchemy and Goal Mind, models of the first group have been developed for language use and leaning in young children, the effect of semantic/spatial concept mapping on automaticity of tasks, attention/arousal mechanisms in animals, and the emotional control of attention and arousal.

Gold Seeker applications that attempt to address real-world problems are supported by a loose set of tools and concepts called Eldorado. The Eldorado framework is suitable for applications as diverse as semantic web search engines and autonomous robots. One major focus of Eldorado is the development of an architecture for a smart home.

For the smart home application, Goal Mind provides development tools, inference engines, and database support for cognitive modeling. The Alchemy environment in which Goal Mind models execute supports distributed platform design, implementation and testing. Eldorado provides a smart home simulator called SimHouse.

GOLD MIND AGENTS

Unlike some computational theories that see a biological entity as a society of agents, Gold Mind views the whole entity as an agent made up of relatively independent components. This agent view is based on the cognitive theory of Hebbian brain areas, in which cognition can be viewed as composed of interacting functional components at varying levels of decomposition. However, both the inference within a component and the basic interface method between components is allowed to be symbolic. The components making up an agent

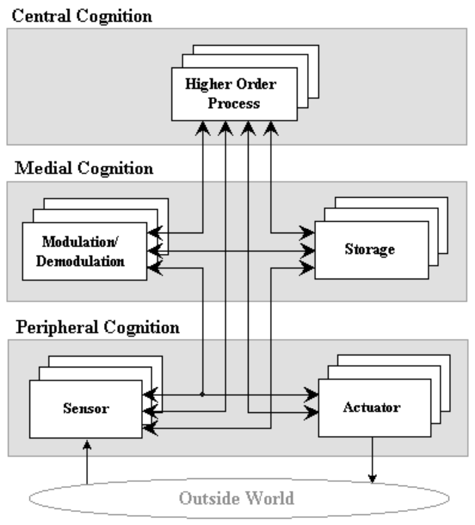

Figure 2. The 3 layers of a Goal Mind agent.

are logically subdivided into three layers: perceptual, medial, and higher order (Figure 2).

An agent in Goal Mind is a collection of components realized as Alchemy nodes. Goal Mind components, also known as etherons (Figure 3), are specializations of general-purpose Alchemy nodes. The etheron model places restrictions on the way its underlying node can be connected and how it can function. Special interface components provide inter-agent communication and the perceptual interface to the outside world. Other component types support a standardized way of intra-agent communication using a stimuli routing network that attempts to capture, at a high level, the way the brain controls communication between neurons and brain areas. Reasoning within agents is supported by other etherons that support multi-modal methods including reasoning via rules, semantic networks, neural networks, and procedures in C++.

Interaction between modeled entities and their surroundings are implemented as agents within a distributed multi-agent model. There is no limit to the number of agents that can exist within a Gold Mind application, but most of the complexity within an application is normally contained within the number of components making up each agent, not in the number of agents being modeled. Agents are free to communicate with each other using a formal language like KQML, but most of the existing work with Gold Mind has focused on using an agent communication language developed to support to natural language research.

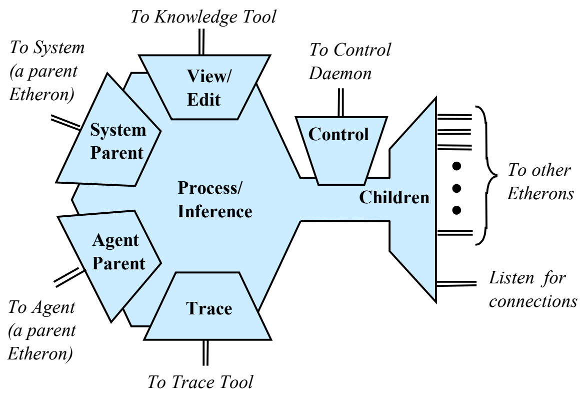

Figure 3. An *etheron* is a Goal Mind component.

Agent
Decision Layer
Processing (planning, learning, deciding…)
Information Layer
Data storage, active DB.
Communication Layer
Interface to other agents, inhabitants
Physical Layer
Interface to physical devices

Figure 4. Abstract Structure of an Agent.

To build a Gold Mind application, the designer defines the number and relationship between agents using a Graphical User Interface (GUI). The component structure of each agent can then be defined using the same GUI. While the user is free to define new component types, normally an agent is created using pre-defined components that are simply loaded with the required knowledge using the appropriate format for the type of reasoner supported by that component. Once this is done, the Alchemy support layer is responsible for distributing the application and balancing its load with all of the other applications running on a defined cluster of processors connect via a LAN or a WAN.

SMART HOME ARCHITECTURE

A smart home can be modeled as a collection of intelligent agents that perform tasks to improve the safety, comfort and economy of inhabitants. Each agent acts with autonomy whenever possible, yet has the ability to communicate with other agents to coordinate actions or acquire information. Agents must be distributed to consider communications requirements and processor loading.

The abstract structure of an individual agent is layered (Figure 4). The physical layer exists only in those agents that need to interface to physical devices such as light sensors, thermostat controls, or bar code readers. The communications layer handles the formatting, transmission and receipt of messages to or from other agents as well as agent registration. The information layer maintains local storage needs and, where needed, an active database. The decision layer exists in agents that need to perform processing beyond the basic queries and event-condition-action rules of the active database in the information layer. Planning, learning, decision-making, conflict resolution and other complex reasoning tasks are implemented in this layer. Examples of tasks the smart home needs to perform are:

- Generate a shopping list of kitchen inventory items that are below their reorder point.
- Retrieve a recipe that can be made from items available in the kitchen.
- Display a list of current or previously recorded TV programs that an inhabitant might be interested in.
- When the inhabitant resets the wake-up time, adjust the start timer for the coffee maker, hot water heater and towel warmer.

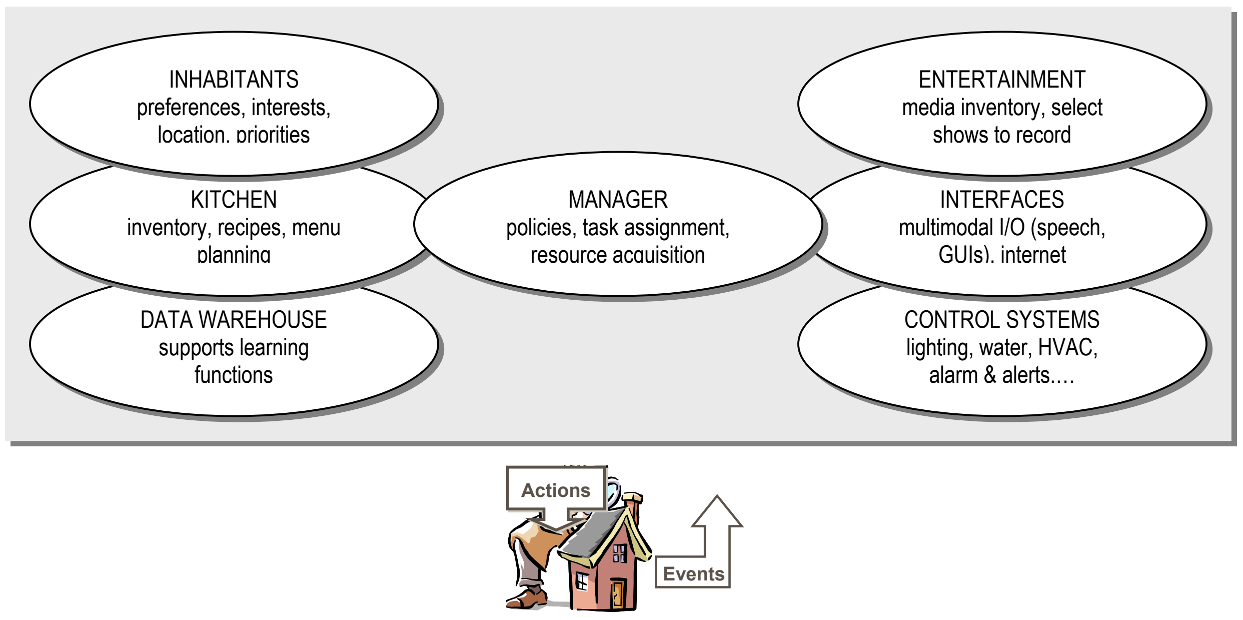
INHABITANTS
preferences, interests, location, priorities
ENTERTAINMENT
media inventory, select shows to record
KITCHEN
inventory, recipes, menu planning
MANAGER
policies, task assignment, resource acquisition
INTERFACES
multimodal I/O (speech, GUIs), internet
DATA WAREHOUSE
supports learning functions
CONTROL SYSTEMS
lighting, water, HVAC, alarm & alerts….
Actions
Events

Figure 5. Package diagram for the Goal Mind smart home architecture.

The layers of the agents map onto the Goal Mind processing categories of higher order, medial and perceptual processing. These correspond to the decision layer (higher order), information layer (medial) and the communication and physical layers (perceptual). Processing occurs asynchronously through the layers. Each agent is defined as a collection of one or more nodes, with one or more special message handler nodes. Messages are defined for an application and the message handlers are written to exchange and process those messages. Broadcast or multicast communication is supported. A message handler accepts only messages it understands and has time to handle. Written in C++, these handlers can be integrated with a semantic network, production system, or neural network created with the Goal Mind supported tools for these reasoning methods.

The smart home architecture (Figure 5) is decomposed into collections of agents that perform related tasks. These agents will vary in their complexity based on 1) the amount of information they can acquire about the environment from the existing sensors, 2) the level of control they exert on the environment through existing actuators, and 3) the level of communication and control allowed between agents. As hardware and software components in the environment change, these agents will need to adapt to maximize the use of the available sensor, actuators and network connectivity. Using Gold Mind such changes to the functionality and scope of agents can be added (or deleted) with little or no impact on the system as a whole.

The kitchen agent is responsible for inventory control, recipe management, and menu planning. Communication between these components is frequent. The menu planner must request inventory availability when an inhabitant has requested a menu that can be made from in-stock items. Inter-agent communication is defined as needed. Examples are 1) recipe management requests inhabitant food preferences and allergies, 2) the data warehouse receives messages about inventory usage to use for predicting consumption rates, thus allowing items to be placed on a shopping list before they will likely be needed.

The entertainment agent maintains a media inventory, controls devices such as televisions or DVD players, and learns inhabitants viewing patterns from the interactions that are captured and sent to the data warehouse. Two simple predictors have been created to date. One is based on a simple scoring scheme that counts how often a show is watched; as the time since the last viewing increases, the score is decremented. The second uses a rule-based system that matches contents of the television guide listing to inhabitant interests (retrieved from the inhabitant agent) to predict shows that an inhabitant may like but has not viewed before. These schemes will be replaced with more sophisticated predictors, most likely by integrating with Tivo-like services.

The interfaces agent receives requests for communication with inhabitants and devices outside the system (e.g., the Internet, PDA's). This functionality is decoupled from the other agents because of the expected degree of change of human-machine interaction and the potentially large number of communication modalities. The current system supports interaction through Unix and Windows PC's, a Sharp Zaurus personal digital assistant, a Microsoft speech recognition and synthesis engine, and a wireless bar code scanner used for input of kitchen inventory. We expect future interaction devices to include RFID readers and touch screens. Other agents send messages that describe the data to be output and, when required, the expected return data. The requesting agent also specifies the required or preferred communication mode. The initial interface to the system is a speech-enable avatar created with Haptek's avatar-generator, called Kate, which is currently being integrated with the smart kitchen agent. As inhabitants request various functions on devices that have display capability, the interface changes appropriately, for example to the recipe search screen.

The other agents each perform their specialized tasks. The control systems agent is intended to interact with physical devices (thermostat, hot water heater, etc.) and to monitor conditions for which an alarm (emergencies like fire) or alert (warnings and reminders like changing the air conditioning filter) should be generated. The data warehouse is the primary repository for historical inhabitant, system and device interactions from which the other agents collect data needed for their specific prediction or decision-making tasks. The inhabitant agent's purpose is to keep track of the inhabitants' current location, and to store acquired and learned inhabitant preferences, interests, and priorities. The entertainment agent, for example, could be notified when Sarah enters the den so that her favorite music genre could be played. The manager agent's purpose is to use encoded general policies and heuristics for selecting appropriate actions when conflicts occur between agents and for reallocating resources and tasks among the agents when the environment changes, such as when a device fails.

A complex and critical function within a smart home is event handling. Challenges include recognizing interesting event sequences, long-term transactions, and error handling. None of these challenges have fully been addressed in our system, but the literature suggests mechanisms that can be employed. Recognition of event sequences may be performed by domain independent methods that use pattern matching over strings or using domain-dependent representations of prototypical sequences of inhabitant-device interactions. We have developed a prototype of the later using case-based reasoning. Long term transactions, such as those involving queries over sensors, can be addressed using methods described in [1]. Error handling is designed using contingency-theoretic principles for propagating un-handled errors to specialized agents [2]. In the next section, we illustrate some of the important aspects of event handling as performed in one of the smart home databases we have developed.

Smart Home Event Handling

Within the smart home system, active databases store, filter and analyze data to perform actions when events occur. Four categories of events, each of which may be primitive or composite, include 1) data manipulation events, 2) temporal events, 3) exception events, and 4) behavioral events [11]. Event class diagrams are shown in Figure 6.

Data manipulation events correspond to standard updates in the relational database model. For example, in an application used to predict television-viewing patterns by a particular member of the household, interface nodes collect information about the television viewing session such as channels, duration and genre of what was viewed. This data is recorded in the warehouse agent to support learning performed in the entertainment agent. Another example, from the systems agent, shows the basic format of active rules in a database as being a specification of an event, zero or more conditions, and the action to be performed:

Description:	Lighting upon Inhabitant Entry to House
Event:	One or more inhabitants enter though the house doors
Condition:	Lights luminosity is below a certain point
Action:	Adjust the light luminosity to pre-determined value

Temporal events occur based on a specified pattern with time. This category may include events that occur periodically within a schedule, or events that occur at a specific time. For example, inhabitants may desire to have the environmental temperature in the living room set at 70 degrees at 7:00 AM, when inhabitants awaken every morning. In contrast, an instant event example is the recording of a television show occurring at 8:00 PM on Sunday, February 10, 2003. The first example relies on a specified schedule, while the second event relies on recognizing a specific instance in time. In event-condition-action (ECA) form, a rule that protects the safety of the inhabitants might be

Description:	Unattended Stove
Event:	Stove has remained on for 5 minutes
Condition:	No adult inhabitant is in the room
Action:	Signal alert to an adult inhabitant AND turn burner off if heat threshold is exceeded

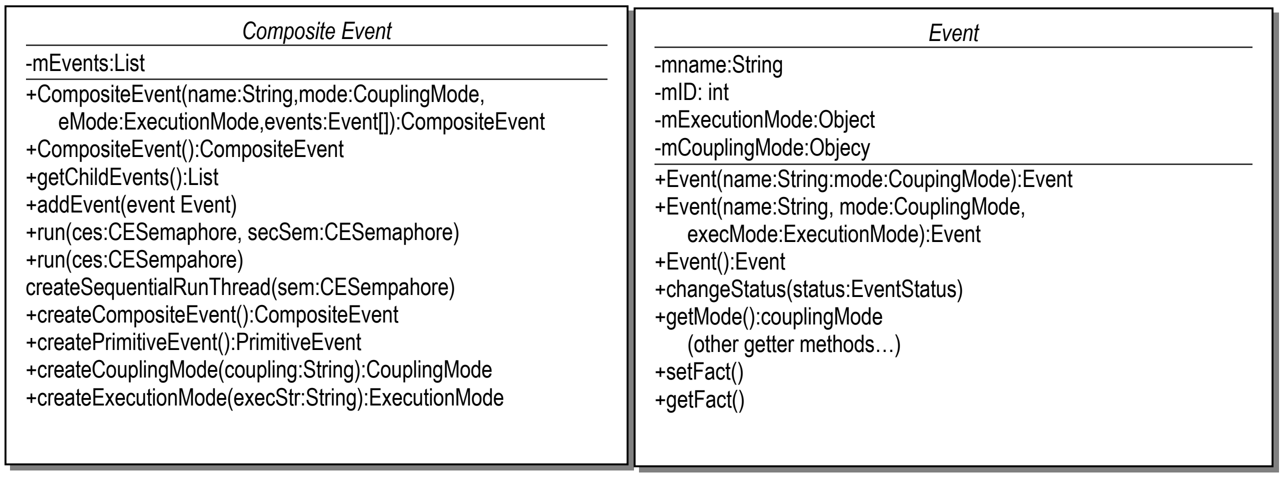
```
Composite Event
-mEvents:List
+CompositeEvent(name:String,mode:CouplingMode,
    eMode:ExecutionMode,events:Event[]):CompositeEvent
+CompositeEvent():CompositeEvent
+getChildEvents():List
+addEvent(event Event)
+run(ces:CESemaphore, secSem:CESemaphore)
+run(ces:CESempahore)
createSequentialRunThread(sem:CESempahore)
+createCompositeEvent():CompositeEvent
+createPrimitiveEvent():PrimitiveEvent
+createCouplingMode(coupling:String):CouplingMode
+createExecutionMode(execStr:String):ExecutionMode
```
```
Event
-mname:String
-mID: int
-mExecutionMode:Object
-mCouplingMode:Objecy
+Event(name:String:mode:CoupingMode):Event
+Event(name:String, mode:CouplingMode,
    execMode:ExecutionMode):Event
+Event():Event
+changeStatus(status:EventStatus)
+getMode():couplingMode
    (other getter methods…)
+setFact()
+getFact()
```

Figure 6. Abstract and composite event class diagrams

Exception events deal with failures or unexpected behavior and actions must be carefully designed to handle these conditions. One example of an exception event is a power failure. What actions should the system take when power is restored? Anticipated to be the most difficult event category, these are not yet modeled in the current implementation. Autonomic computing [8] concepts and techniques appear to be appropriate and will be the evaluated in future work.

Behavioral events represent actions taken by a particular inhabitant. Events in this category frequently include events that require recording for interpretation by the decision layer. For instance, the system may record when the children arrive home from school, or when dinner for a particular household is served.

Events can originate from three sources: as an external event in the physical environment, through the firing of a matched rule and through a message received from the interfaces agent. The event-condition-action rules are implemented using an integrated rule-based expert system. Actions are performed upon an initial rule triggering by an event for which conditions are met. The right-hand-side (RHS) of the rules contains actions that are generally reflection calls to software device controllers. Additionally, retraction or modification of facts is another operation performed in the RHS of rules.

FUTURE WORK

While the SimHouse simulator provided by Eldorado allows us to test many of the concepts of the smart home research with a very realistic environment, we plan to take an iterative approach to improving our multi-agent framework that relies on both simulation and implementation in a real environment. One aspect of this approach is the 'pushing down' of the framework into a microcontroller based processing environment. We are currently developing a SX-based microcontroller network for analyzing how much of the framework can run in the firmware level of processing that would most likely exist in a smart home environment. This work will give us some useful data in scoping exactly what type of computational resources will be needed for high-level smart home control and monitoring.Another area of work is the way in which conflict resolution is best addressed in a fully connected smart home where the data of every sensor can be accessed by every agent and multiple agents can control the same set of actuators. Improvements to our agent communication and control approach will first be tested using SimHouse, and then tested in a physical smart home environment. While we expect this work to require minor changes to the framework, the complexity of existing Gold Mind models demonstrates that these changes are well within the capacity of the modeling tools to support.

CONCLUSIONS

We have described a distributed, intelligent multi-agent system framework that is suitable for implementing intelligent environment applications. Building on the adaptive nature of biological systems, this framework is being used to develop a smart home system in which capabilities can be added incrementally. The current system supports kitchen activities, entertainment, a virtual human interface, and a lighting control sub-system. Active databases record events, perform actions, and aid the learning/adaptation of inhabitant preferences.

ACKNOWLEDGEMENTS

This work was sponsored in part by the National Science Foundation under grant EIA-0203499. Special thanks to the student research assistants, especially Edwin Wong, Steven Johnston, Nihir Patel, and Russ Tribble.

REFERENCES

[1] P.Bonnet, J.Gehrke, and P.Seshadri. "Towards Sensor Database Systems". 2nd Int. Conference on Mobile Data Management, Hong Kong, Jan. 2001, ww.cs.cornell.edu/ johannes/papers/2001/MDM2001-sensor.pdf.

[2] J.R. Durrett, L.J. Burnell, and J.W. Priest. "A Virtual Advisor Utilizing Multi-Agent Software Teams and Contingency Theoretic Coordination Models". **Virtual Education: Cases in Learning and Teaching Technologies Part I**, Chapter 4, Fawzi Albalooshi editor, IRM Press, Hershey PA, Spring, 2003, pp 50-63.

[3] C. Hannon, "A Geographically Distributed Processing Environment for Intelligent Systems", **Proc. of the 15th Int. Parallel and Distributed Systems Conference**, Sept. 19-21, 2002 pp. 355-360.

[4] C. Hannon and R. Rinewalt, "Addressing Security Issues in Geographical Distributed Systems**", 4th Mexican Int. Conf. on Computer Science**, Sept. 8-12, 2003 pp. 182-189.

[5] C. Hannon and D. J. Cook. "Developing a Tool for Unified Cognitive Modeling using a Model of Learning and Understanding in Young Children." **The Int. Journal of Artificial Intelligence Tools**, 10 (2001): 39-63.

[6] C. Hannon, "Emotion-based Control Mechanisms for Agent Systems", **ISE 2003**, July 20-25, 2003 pp. 10-15.

[7] L. Keeley, "Smart Homes? A stupid idea". **Context**, Dec. 2000/Jan. 2001, www.contextmag.com/archives/ 200012/thegreatlie.asp.

[8] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing". **IEEE Computer**. Jan. 2003, pp 41-50.

[9] J. O'Brien, T. Rodden, M. Rouncefield, and J. Hughes, "At Home with the Technology: An Ethnographic Study of a Set-Top-Box Trial**". ACM Transactions on Computer-Human Interaction**, 6(3), Sep. 1999, pp 282-308.

[10] D. Servant and A. Drogoul, "Combining Amorphous Computing and Reactive Agent-Based Systems: A Paradigm for Pervasive Intelligence?" **AAMAS '02**, July 15-19, 2002 pp. 441-448. .

[11] E. S. Wong, L.J. Burnell and C. J. Hannon, "An Active Architecture for Managing Events in Pervasive Computing Environments", **FLAIRS 2004**, Miami Beach, FL. May 17-19, 2004, CD-ROM