AAMAS'02, pages 578-586

An Open Agent Architecture for Assisting Elder Independence

Karen Zita Haigh, John Phelps, Christopher W. Geib
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
{khaigh,jphelps,geib}@htc.honeywell.com

ABSTRACT

We are building an agent-oriented system to aid elderly people to live longer in their homes, increasing the duration of their independence from round-the-clock care while maintaining important social connectedness and reducing caregiver burden. The Independent LifeStyle Assistant™ (I.L.S.A.) is a multiagent system that incorporates a unified sensing model, probabilistically derived situation awareness, hierarchical task network response planning, real-time action selection control, complex coordination, and machine learning. This paper describes the problem, our reasoning for selecting an agent-based approach, and the architecture of the system.

CATEGORIES & SUBJECT DESCRIPTORS: I.2.11 [Artificial Intelligence] Distributed Artificial Intelligents – Intelligent agents, Multiagent systems; K.4.2 [Computers and Society] Social Issues – Assistive technologies for persons with disabilities.

GENERAL TERMS: Algorithms, Human Factors

1. INTRODUCTION

The Minneapolis Star Tribune ran an article on April 8, 1999 headlined "Woman, 89, says relocation violates her rights." The woman sued her nephew in an effort to remain in her legacy home rather than move to a nursing home. This headline exemplifies the strong feelings elicited by elder care issues. When it becomes apparent that a loved one can no longer safely take care of themselves, a nursing home is often the only option, in spite of the financial and emotional strain placed on the family.

Historically, 43% of Americans over the age of 65 will enter a nursing home for at least one year. With this demographic growing rapidly – the American Administration on Aging estimates that it will double to 69.4 million, 22% of the population, by 2030 – the economic strain assumed by the nation will increase dramatically.

We are developing an automated monitoring and caregiving system called *Independent LifeStyle Assistant*™ (I.L.S.A.) that will be a better alternative. Researchers and manufacturers are developing a host of home automation devices that will soon be available. I.L.S.A. will be the integration of these individual functions, augmented with advanced reasoning capabilities to create an intelligent, coherent, useful assistant that helps people enjoy a prolonged independent lifestyle. I.L.S.A. will provide this integration through a unique, knowledge-based approach to situation assessment and interaction generation. That is, I.L.S.A. will coordinate device inputs and outputs intelligently through holistic situation tracking. I.L.S.A. shall:

- Provide accurate situation assessment for unconstrained, unstructured environments, using a network of low cost sensors.
- Provide intelligent, accurate, safe, and acceptable user interaction generation for potentially technophobic users with varying capabilities and constraints
- Provide easy-to-use, low cost installation and configuration aids, and on-going intelligent adaptation to support situation assessment and response.

By providing intelligent, affordable, usable, and expandable integration of home automation devices, I.L.S.A. will support daily activities, facilitate remote interaction with family and caregivers, provide safety and security, and otherwise assist the elderly or disabled, potentially deferring nursing home care for years.

In order to support this desired functionality, we have decided to use an agent-oriented programming paradigm. We chose this paradigm to support coarse grained functional modularity that we see as critical to the rapid creation of a system that exhibits intelligence and flexibility in its interactions with users, as well as in its own growth. This paper presents the arguments and thought process behind our decision, and then presents the architecture itself.

2. REQUIREMENTS

Every I.L.S.A. installation will (1) be in a home with a different layout and suite of sensor and actuator capabilities, and (2) supporting a technophobic client with unique capabilities, needs and care-giving support network. I.L.S.A. thus must be rapidly deployable, easy to configure and easy

AAMAS'02, July 15-19, 2002, Bologna, Italy.

to update as the client ages and technology changes. I.L.S.A. shall facilitiate the evolution of a particular installation by providing an open architecture into which new devices and reasoning modules may be plugged. The system will require at least the following:

Modularity. Due to the high complexity of the I.L.S.A. domain, distributed and encapsulated expertise will be critical to I.L.S.A.'s success. This modularity will directly enable the extendability goal by carving the problem into smaller logical units that can be added or changed to extend, refine or adapt functionality.

Dynamic discovery of capabilities. Discovery is a procedure that enhances and/or replaces the hardcoding of infrastructure references in system configuration files or code. As a result, it is an essential capability of an open system. A component being installed in an I.L.S.A. network needs to be able to discover what infrastructure capabilities are available to it.

A public ontology. To enable a given component to reliably integrate and communicate with other components, it is necessary that they share a communication protocol. I.L.S.A. has opted not to use a blackboard architecture in order to make implementation of I.L.S.A. more feasible on home hardware. Nevertheless, I.L.S.A. will still require the development of specific communication protocols (vocabulary and syntax) to enforce "I.L.S.A. Compatibility" across modules.

We generally tolerate redundancy (read, inefficiency) in order to facilitate adaptability, ease of fielding and tuning.

3. ARCHITECTURAL PARADIGM

Given the requirements described above, we performed an analysis of the infrastructure services that would support an I.L.S.A. system. We outlined an operating scenario of several components, detailing the computations, communication and coordination required to provide the desired functionality. Given that scenario, we then decided on an agent-oriented programming paradigm, and evaluated several agent architecture alternatives, finally selecting the Java Agent Development Environment *JADE* [2] for I.L.S.A.

3.1 Scenario

We have repeatedly found it useful when analyzing agent-oriented systems to try to concretely capture one scenario in which most of the agents participate. Table 1 shows a sample of the components we analyzed, their functions and the most likely components they will interact with. A more detailed description can be found in [20].

In this scenario we track a hypothetical I.L.S.A. client, whom we'll name Lois, through a morning of normal behavior. When Lois wakes up, a mobility and client activity agent tracks her getting out of bed. If Lois falls, and I.L.S.A. believes she was hurt, the mobility agent should raise an alarm. The diagnosis model that this agent uses can be arbitrarily complex, and so can the conversations that it has during diagnosis with the other agents in the system, including notably the task tracking agent.

A medication agent monitors and adjusts a medication schedule established by Lois' caregiver. It discretely reminds Lois to take her medication at times that are convenient to her. For instance, it would keep track of which medications go with food and at what times in the day her medications should be taken.

An I.L.S.A. environment agent learns Lois's home environment preferences, from the temperature to the level of noise in rooms over the course of a day, and in the context of a given activity, occasion, or season. The context can be part of a recognized plan or known cultural knowledge or learned information. Intelligent behavior is highly dependent on understanding the context of a situation.

This sample highlights the need for components to interact with numerous devices, both local and remote, for both sensing and actuation. The reasoning required in each component should not affect reasoning in other components. They each have separate goals and privacy concerns. Interactions might get quite complex, as each component can provide information of interest to other components (e.g. the environment agent may wish to know whether the client is capable of turning off the stove). The system will need a rich method for communicating and reasoning about concepts and for discovering capabilities of other components.

In order to provide a baseline "failsafe" functionality, the components need to be independent enough from each other that they do not *require* each other. Certain functionality will be fundamental to the system, for example the home security system and the panic button: even if I.L.S.A. loses connectivity or power, it needs to be able to use the telephone to raise emergency alerts. Other services will be fundamental to the system, such as device subscription services.

The distributed devices, the failsafe function, and the potential growth of the reasoning system highlights the need to support distributed computation.

3.2 Agent Architectural Paradigm

We have adopted an agent-oriented approach for two main reasons: (1) multiple independent computational threads, and (2) the task-centered model of computation that it encourages [49].

Multiple independent computation threads directly support the distributed computation model our task requires. By encouraging a task-centered model of computation, we benefit from the natural byproduct of decoupled areas of computational responsibility. The multithreaded computation model enhances this decoupling by supporting a system design that makes use of the different levels of granularity that a problem presents. Thus, we have an agent or agents responsible for various components essential to good system performance available at several levels of computational responsibility, from device control to client task tracking.

One of the benefits we receive from the agent-oriented paradigm is *reflection*. Reflection is the process of reasoning about and acting upon oneself [32, 44]. Reflection is present at both the individual and social levels of properly constructed agent systems. Reflection at the single agent level primarily means to us that it can reason about the importance of its goals and commitments in a dynamic environment; it is apparent in explicit models of goals, tasks, and execution state. An agent's ability to reason about goals and commitments in the context of an agent system is provided by a common, interchangeable task model [8].

The model is expressed in an ontology [10, 14] and Agent Communication Language [15, 28] that form a common language to describe the domain. This ontologically mediated intercommunication provides an additional benefit: it gives system components the ability to discover services provided by other agents, often through the services of a matchmaker [7]. Discovery directly provides the opportunity for

Components	Functions	Interactions
Eating	• Monitor food quality • Planning meals and keeping track of expected expiration dates, forming a grocery list • Shopping for groceries via the Internet • Raise alerts of low food consumption	• Device Control Component (appliances, water) • Caregiver UI Component (CG-UIC) • Client UI Component (C-UIC) - reminders, questions • Secure Internet Information Broker (SIIB) for nutritional info, recipes and plans
Medication	• Schedule and remind administrations • Monitor effects, and raise alerts	• Device Control Component (Smart Medication Cabinet) • C-UIC - reminders, questions, schedule maintenance • CG-UIC
Mobility	• Tracks entry and movement of all mobile entities in home • Detects falls • Detects lack of activity • Raises alerts	• Device Control Component • C-UIC • CG-UIC • SIIB for tracking data given Lois's (and her friends, pets?) physical and mental parameters (could be done locally as well)
Environment	• Learn device setting preferences • Coordinate resource consumption • Decide whether safety conditions have been violated (fire, CO, temperature, etc) • Raise alerts	• Device Control Component • Mobility • C-UIC - questions, preferences • CG-UIC

Table 1: Sample components, functionality, and likely communication interactions.

an independent agent to expand its range of knowledge without radically changing its control focus. As a result, discovery allows the overall system to grow at run-time without adversely affecting functionality.

The agent-oriented approach gives us modularity, independence, distribution, and discovery, thereby meeting the requirements outlined above.

3.3 Agent Architecture Selection.

We evaluated several popular free agent development frameworks, including AgentTool [1], JADE [2], DECAF [6], FIPA-OS [11], JAFMAS [23], MadKit [31], OAA2 [36], and Zeus [51].

Our evaluation criteria included ease of deployment, stability, protocols, support tools, security and support from the development team (documentation, responsiveness, etc).

We also considered using communication facilitation packages such as Universal Plug 'N' Play (UPnP) [33], JINI [24] and JAFMAS [23] as a basis for developing our own agent architecture. UPnP was especially appealing because it is likely that in the future the devices will communicate with UPnP. However, these protocols lack many of the tools that an existing multiagent infrastructure supports already, including visualization, ontology support, and hierarchical, asynchronous behavior definition.

JADE turned out to be one of the easiest, best supported (although informally), and cleanest implementations that we examined. Further, It also met most of our multiagent infrastructure needs, so we decided to use JADE as our I.L.S.A. agent system prototype development framework.

4. ARCHITECTURE

At the highest level we have designed an architecture for I.L.S.A. that both allows modularity of design as well as enabling interaction. This architecture is defined as a federated set of *agents* that define *agent interfaces*. We will discuss each of these ideas in turn.

4.1 Agents

We define an I.L.S.A. *agent* as a software module that is

- designed around fulfilling a single task or goal, and
- provides at least one *agent interface*.

An individual I.L.S.A. agent is intended to perform a single (possibly very high level) task. Examples of the agent's task include interaction with a client or caregiver, preventing fires in the kitchen, or interfacing to a medication monitoring device. We see the agent as the basic delivery and compositional unit of I.L.S.A. the architecture. As such, different software vendors will produce agents for installation in an I.L.S.A. system to provide new functionality.

While we anticipate a small set of agents will be present in every installation of I.L.S.A., the breakdown of I.L.S.A. functionality into agents is designed to allow a flexible modularity to the system construction. Choosing agents on the basis of provided functionality will allow the end user to customize the system to provide only those functions they want to have without requiring the adoption of functionality they are not interested in.

4.2 Agent interfaces

Agent interfaces provide the inter-agent communication and interaction. Each agent must make available at least one agent interface. In contrast to the task-organized functionality provided by agents, the agent interfaces are designed to allow the agents to provide functionality to each other. They provide for and foster specific kinds of interactions between the agents by restricting the kinds of information that can be provided through each interface. In I.L.S.A. we define three kinds of agent interfaces, hereby termed the SRA interfaces:

1. **Sensor Agent Interface:** Interfaces of this kind answer questions about the current state of the world,

such as "is the stove on or off?", "has Lois taken her medications for the day?", "is Lois in the house?" These interfaces allow others to interact with the agent as though it is just a sensor. An example of this kind of interface is a Kitchen Fire Safety agent that allows other agents to know the state of the stove.

2. **Actuator Agent Interface:** Interfaces of this kind accept requests for actions to change/modify the world, for example: turning the stove on or off, calling Lois on the phone, or flashing the lights. These interfaces allow the agent to be used by others as a simple actuator. Note that the monitoring of an action to verify that it has been done would be carried out by the agent implementing the actuator agent interface rather then by the agent requesting the action.
3. **Reasoner Agent Interface:** Interfaces of this kind answer questions about the future state of the world like, "will Lois be home tonight?", or "Can Lois turn off the TV?" These interfaces are designed to allow the agent to perform reasoning for other agents.

We assume that in general an agent will have more than one interface and may even provide multiple interfaces of the same type. For example the Kitchen Fire Safety agent might provide a sensor agent interface for the state of the stove and a similar but separate one for the toaster oven.

When an agent is registered as part of the I.L.S.A. system it will register the agent interfaces that it makes available. Other agents that wish to make use of these interfaces can be informed of the availability and be reconfigured accordingly.

4.3 Functional Layering

To facilitate description of functionality, there are four main categories of capability that fit into a layered heirarchy.

1. **Sensing.**
2. **Situation Assessment.**
   (a) **Clustering.** Combine multiple sensor reports into a single event. For example, the three-sensor sequence "<pressure-mat hall> <pressure-mat kitchen> <motion-sensor kitchen>" are probably reports of the same event, namely entering the kitchen.
   (b) **Validating.** Increase confidence of patterns, eliminate false positives, weigh competing hypethesized patterns. Essentially, tweak the likelihood of events based on sensor reliability.
   (c) **Situation Assessment.** Based on evidence, predict ramifications. Includes monitoring response plans; for example if the client does *not* respond to the question "Are you OK?" then the detected fall is likely to be more serious.
   (d) **Intent Inference.** Put multiple events together, infer goals of actors. For example, going into the kitchen, opening the fridge, and turning on the stove probably indicate that the client is preparing a meal.
3. **Response Planning.** Based on situation, create general response plan – what to do, who to talk to, how to present it, on what device.
4. **Response Execution and Actuation.** Talks to devices (displays and actuators). Formats the presentation, sends it to the device, and monitors for its successful delivery.

Layers provide a framework in which to describe an agent's capability, rather than a strict enforcement of code.

There are some agents that reside outside this framework, notably because they are not part of the 'reasoning chain' in quite the same way. These would include, for example, customization and configuration (where a user modifies the I.L.S.A. databases), trending (where I.L.S.A. reasons over history), and the log manager.

4.4 Vizualization

The architectural organization is sketched in Figure 1. Devices (both sensor and actuator) will reside in the device layer, communicating with a standard device communication protocol. Agents will communicate within an agent infrastructure. There will be one agent that functions as an adapter to translate device messages.

Devices in the device layer can directly write to the log. Agents must go through a log manager that selectively returns only the requested information. It remains to be determined whether any other software will have direct write-access to the log (for example, must a tool in a doctor's office talk to I.L.S.A., or can it write to the log directly). There are numerous other databases drawn in the picture; accesses into these are currently mediated by manager agents, but similarly we will have to determine whether to open the communications.

Within the agent layer, we show agents and a layering of those agents (described above). Agent-components (small dots) provide basic modular capabilities, including:

- the stove-monitor reasons about whether the stove is on or off, and whether it is actually being used,
- the location-tracker reasons about where the client (or other object of interest) is currently located,
- model managers maintain interactions with the models, checking authenticity, logging transactions, etc.

Agent-components are bundled into agents (larger ovals) according to functional groupings, e.g.

- The Home agent keeps track of every situation going on in the home
- IDS reasons about which display device to utilize for a given task
- The Eating agent provides all the functionality related to the client's eating habits, including monitoring what and when the client is eating, monitoring the freshness of food, creating menus and grocery lists, and raising alerts when necessary.

Communication between agents is through the SRA interfaces (see Section 4.2), implemented in JADE, and describing their capability with the I.L.S.A. ontology. Within an agent, agent-components may communicate using whatever mechanism they choose, including the extremes of (1) choosing to be one piece of monolithic code that requires no communication, or (2) using their own proprietary communication method, or (3) choosing to use the I.L.S.A. ontology and communication protocols. (For reasons of modularity and consistency across vendors, we recommend the latter.) If a vendor decides to sell a particular device or reasoning module, the vendor may decide what information to keep proprietary or private, and what information to make public.

While it is unlikely that an agent or agent-component residing in the response planning layer will want or need to

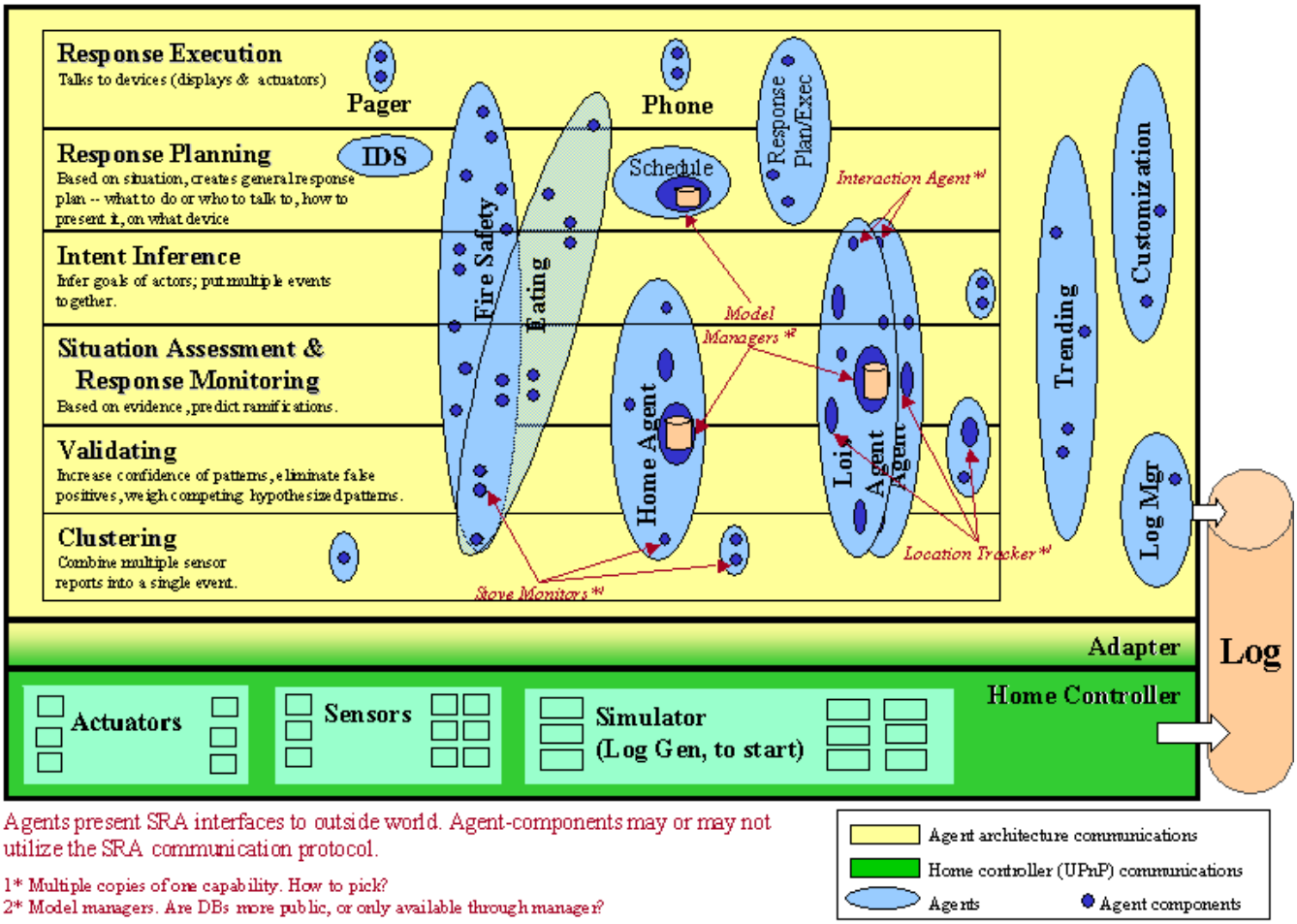

Figure 1: I.L.S.A. reasoning architecture.

access an agent in the pattern matching layer (i.e. skipping layers), we do not restrict this information flow. They must only maintain 'ontological purity' in their communications with other agents. The same holds true for agents that can reason over multiple layers in the reasoning architecture. 'Ontological purity' means that the ontology defines concepts that can be shared or inspected between agents, and those concepts exist within a level of the reasoning architecture. Concepts can be used within or across levels. Concepts must be maintained across agents.

The major issue that arises from this architecture is how to handle multiple copies of a particular agent or agent-component. (Note that a single agent may/will publish multiple capabilities, and a single capability may be published by multiple agents.) There may be cases when

1. we want to create a general I.L.S.A. 'operating system' that has a set of generic agents everyone can use,
2. multiple agents need the same agent-component (not in the base system), and we don't want dozens of identical agent-components running at the same time,
3. a specific vendor wants a specialized capability (either agent or component) for their application only, or
4. a specific vendor wants to sell a new-improved agent or agent-component.

We will have to handle the resolution and discovery of these capabilities. Allowing multiple views of 'stove use' means either overhead in knowing which one is right, or overhead in discovering which one you want to use. A precise ontology is probably an effective way to solve this problem.

5. IMPLEMENTATION

We are building an I.L.S.A. implementation that demonstrates functionality. The implementation demonstrates the complete cycle of I.L.S.A. interactions, from sensors to reasoning to alerts and home control. This implementation will also be used as a field study to determine commercial viability of the product. Features include:

- Active Monitoring: panic button
- Passive Monitoring: toilet use, basic mobility, medication compliance, sleeping, modes (on/off)
- Cognitive Support: reminders, coordinating multiple caregivers
- Alarms, Alerts and Notifications: auto contacting caregivers (by telephone or email)
- Reports: summary reports of client behaviour
- Remote access to information (allowing users to monitor or interact with the client/home)

Other high-priority features to add in future releases include fall detection, environment monitoring (temperature, fire, etc), security, eating, and to-do lists. We conducted an in-depth knowledge acquisition effort to identify potential features for an I.L.S.A. system; the final list contained apporximately 300 capabilities of interest to elders, their caregivers (formal or informal), and other interested parties

(e.g. insurance).

For this field study, we are using the Honeywell Home Controller system as the backbone communications infrastructure for I.L.S.A. (a product would likely use a different, less expensive, platform). Sensing devices use a simple XML schema to record events. JADE provides the agent communication layer.

The system has been installed in the homes of four of the project engineers, with 12-20 sensors per home. These sensors include motion detectors, pressure pads, contact switches (door and cabinet open), flush sensors, a medication caddy, and a panic button. Interaction devices for this implementation are email, WWW browsers and telephones.

Currently there are 16 agents in the system, including device controllers, behaviour recognizers, response planners, and system management. Specific examples include:

- **SensorFilter**: remove noisy signals, cluster signals from one device
- **IntentRecognition**: probabilistic task tracking [13, 17, 12] to infer the goals of the actors
- **Ontology**: parse ontological components and their relationships
- **Medication**: monitor use of medication caddy, raise alerts and generate reminders
- **Mobility; calculate statistics about the elder's mobility, raise alerts**
- **Panic Button**: monitor alarm signals from the body-worn panic button
- **ResponseCoordinator**: suppress and merge alters and reminders as appropriate, select contactee; see [48] for more details
- **PhoneAgent**: format alert for presentation and manage communication with contactee

We have been collecting data from this prototype system since July 2001. Our main focus for the first six months was on configuring the hardware (the response capability was disabled). We had numerous sensor difficulties – door sensors for example, are designed for security systems to raise an alarm when the door is opened. Since doors are often left open for long periods of time, the sensor starts 'shouting' and drowns out the signals of other sensors, and is hence not appropriate for a a continuous monitoring environment.

We have started the initial phases of a field study in which I.L.S.A. will support 20 elders in their homes for a period of six months. We have partnered with an assisted living facility in Minneapolis, and the University of Florida's International Center for Technology for Successful Aging will install and administer 10 systems. At this stage, we are waiting for IRB approval for human subjects studies, and have specified the capabilities of participants. We hope to install by mid June.

Beyond work to enhance the capabilities of these existing agents, adding additional sensors, actuators and interaction devices, we are also developing agents that

- capture the required configuration knowledge for a given user need
- use vision to track the location of people [37],
- use an interactive design system, e.g. [38, 40], to format interations on-the-fly based on the current task and the person's capabilities (e.g. hearing or sight impaired).
- use machine learning to identify frequent observed behaviours [19, 18]

We envision certain agents as providing a service to the community, while others meet a user's specific needs. For example, the ontology, the task tracking, and each of the device agents each provide services. Individual agents are constructed depending only on the capabilities their designer wishes to impart, and through which tools those capabilities are rendered. For instance, the medication agent needs to understand the intent of the client with regards to taking their medication, so it subscribes to messages from the task tracker. The mobility agent, on the other hand, only calculates statistics about the elders' motion, and hence does not uilize the services of the task tracker.

6. RELATED WORK

Over the last ten years, numerous efforts have been made to create monitoring systems for elders.

Togawa *et al* [46, 50] was one of the first projects to use passive sensing of everyday activities to monitor subjects. Their main focus is to monitor physiological parameters, but they also monitor, for example, sleep hours, toileting habits, body weight and computer use. The systems collect data for analysis by a caregiver, and do not raise alarms or automatically respond to the data in any way.

Celler *et al* [3] collects data for measuring the behaviour and functional health status of the elderly, and assessing changes in that status. Data analysis is off-line, and reports are generated for participants who have demonstrated a consistent change in functional health status.

Inada *et al* [22] was perhaps the first system to incorporate the capability to contact emergency personnel whenever there is a sudden change in the patient's condition, and the patient initiates the call. The system collects biological information, physical activity, and subjective information such as complaints.

Richardson and Poulson [41, 42] describe installments of assistive home control technologies for supporting independent living. The main focus of this work was to make devices more supportive and easier to use by creating a common framework for controlling and monitoring devices, both from within the home and externally. One of the installed bases includes medical monitoring devices and raises appropriate alarms, and they call for systems that raise alarms for all appropriate 'supportive' purposes.

Glascock and Kutzik [16] similarly aims at using non-intrusive monitoring to detect functional activities of daily living. This system does not respond to the collected data in any way; the data is logged and later analyzed off-site. Their patent [27], however, covers the capability of generating a control signal in response to the collected information.

Chan *et al* incorporate the results of machine learning to control environments and automatically raise alarms. A neural network is used to learn the habits of this group of people (temperature and location) [4]. The network is trained over a given period, and then used to control the temperature of a room based on expected occupancy. The authors extend this work to recognize behavioural changes and raise alarms [45].

Sixsmith [43] describes and evaluates results from an intelligent home system installed in 22 homes. The system raises alerts for "potential cause for concern" – namely when the current activity is outside a activity profile based on the average patterns of activity. The system was well-perceived by the elders and their caregivers.

Leikas *et al* [29] describe a security system for monitoring the activities of demented people at home. Vigil [47] has a similar concept, and has fielded over 2000 sites in assisted living facilities.

The paucity of installed systems is most likely due to the complexity of this domain. Vigil's product focusses on a tiny subset of the problem (essentially bedwetting and door alarms). It is our belief that the main reason more complex systems have not been fielded is a direct result of a weak reasoning framework. Prior approaches have focused on the hardware and networking capabilities of the system, and rarely focused on the reasoning or inferencing component. Systems are unable to integrate the information, assess the situation and communicate it in an appropriate fashion. In an effort to find richer reasoning systems, we turn to the intelligent environments community.

Huberman and Clearwater [21] built a agent-based market-based temperature controller. Chatterjee [5] built an agent-based system with three device agents (TV, phone, stereo), and tried to find correlations between the interactions of those agents. The Intelligent Home project [30] researches multi-agent systems in the context of managing a simulated intelligent environment. The primary research focus is on resource coordination, e.g. managing the hot water supply.

The Neural Network House [35] also used neural networks to 'self-program' a home controller. The system learned the users preferred environmental settings, and then controlled the house to meet those settings and optimize for energy conservation.

The Georgia Tech Aware Home [9, 26], MIT's House_*n* [34], University of Washington's Assisted Cognition [25] and the University of Pittsburgh's Pearl [39] are current research projects in this area. All of these projects have similar goals to I.L.S.A., but have taken very different approaches. The AwareHome and House_*n* are essentially platforms for researchers, and projects tend to be unrelated to one another. Assisted Cognition is less than one year old, and Pearl is primarily a robotics project.

7. CONCLUSION

As a necessary step to achieving our aim of an open architecture, we will be publishing our ontology and API shortly. The ontology is currently undergoing review with our subcontractors and interested device manufacturers.

I.L.S.A. is an ambitious agent-oriented development project. It is advancing the feasibility of agents as the appropriate software abstraction for systems with many computationally complex, interacting processes. We hope to help demonstrate the value of using on off-the-shelf agent architecture, rather than developing our own. We also aim to demonstrate the feasibility of integrating multiple, disparate AI technologies in one cohesive system.

Our project is focussing on the reasoning and inferencing components of the system. We hope that this effort will lay a strong foundation for a viable product that meets the needs of elders and their caregivers.

ACKNOWLEDGEMENTS

This work was performed under the support of the U.S. Department of Commerce, National Institute of Standards and Technology, Advanced Technology Program, Cooperative Agreement Number 70NAN80H3020.

8. REFERENCES

[1] AgentTool, 2001. http://en.afit.edu/ai/agentool.htm.

[2] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: A FIPA-compliant agent framework. In *Proceedings of The Practical Applications of Intelligent Agents and MultiAgent Technology (PAAM)*, pages 97–108, 1999. http://sharon.cselt.it/projects/jade/.

[3] B. G. Celler, W. Earnshaw, E. D. Ilsar, L. Betbeder-Matibet, M. F. Harris, R. Clark, T. Hesketh, and N. H. Lovell. Remote monitoring of health status of the elderly at home. a multidisciplinary project on aging at the University of New South Wales. *International Journal of Bio-Medical Computing*, 40:147–155, 1995.

[4] M. Chan, C. Hariton, P. Ringeard, and E. Campo. Smart house automation system for the elderly and the disabled. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1586–1589, 1995.

[5] S. Chatterjee. SANI: A seamless and non-intrusive framework and agent for creating intelligent interative homes. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 436–440, 1998.

[6] DECAF, 2001. http://www.eecis.udel.edu/~decaf/.

[7] K. Decker, K. Sycara, and M. Williamson. Matchmaking and brokering. In *Proceedings of the Second International Conference on Multi-Agent Systems*, 1996.

[8] K. S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, Amherst, Massachusetts, 1995.

[9] A. K. Dey, D. Salber, and G. D. Abowd. A context-based infrastructure for smart environments. In *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99)*, pages 114–128, 1999.

[10] S. Falasconi, G. Lanzola, and M. Stefanelli. Using ontologies in multi-agent systems. In *Tenth Knowledge Acquisition For Knowledge-Based Systems Workshop (KAW'96)*, 1996.

[11] FIPA-OS, 2001. http://fipa-os.sourceforge.net/.

[12] C. W. Geib. Problems with intent recognition for elder care. In *Proceedings of the AAAI Workshop "Automation as Caregiver"*, 2002. To appear.

[13] C. W. Geib and R. P. Goldman. Probabilistic plan recognition for hostile agents. In *Proceedings of the FLAIRS 2001 Conference*, 2001.

[14] M. R. Genesereth. An agent-based approach to software interoperability. In *Proceedings of the DARPA Software Technology Conference*, pages 359–366, 1992.

[15] M. R. Genesereth and S. P. Ketchpel. Software agents. *Communications of the ACM*, 7(37):48–53, 1994.

[16] A. P. Glascock and D. M. Kutzik. Behavioral telemedicine: A new approach to the continuous nonintrusive monitoring of activities of daily living. *Telemedicine Journal*, 6(1):33–44, 2000.

[17] R. P. Goldman, C. W. Geib, and C. A. Miller. A new model of plan recognition. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[18] V. Guralnik and K. Z. Haigh. Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI workshop "Automation as Caregiver"*, 2002. To appear.

[19] K. Z. Haigh. *Situation-Dependent Learning for Interleaved Planning and Robot Execution.* PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, February 1998. Available as Technical Report CMU-CS-98-108.

[20] K. Z. Haigh, C. W. Geib, C. A. Miller, J. Phelps, and T. Wagner. Agents for recognizing and responding to the behaviour of an elder. In *Proceedings of the AAAI Workshop "Automation as Caregiver"*, 2002.

[21] B. Huberman and S. H. Clearwater. A multi-agent system for controlling building environments. In *Proceedings of the International Conference on MultiAgent Systems*, pages 171–176, 1995.

[22] H. Inada, H. Horio, Y. Sekita, K. Isikawa, and K. Yoshida. A study on a home care support information system. In *Proceedings of the Seventh World Congress on Medical Informatics*, pages 349–353, 1992.

[23] JAFMAS, 2001. http://www.ececs.uc.edu/~abaker/JAFMAS/.

[24] JINI, 2001. http://www.sun.com/jini/.

[25] H. Kautz, L. Arnstein, G. Borriello, O. Etzioni, and D. Fox. An overview of the assisted cognition project. In *Proceedings of the AAAI Workshop "Automation as Caregiver"*, 2002. To appear.

[26] C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99*, Oct 1999.

[27] D. M. Kutzik, A. P. Glascock, D. L. Chute, T. T. Hewett, and B. G. Hornum. System for generating periodic reports, generating trend analysis and intervention in accordance with trend analysis from a detection subsystem for monitoring daily activity, Nov 1997. US Patent number 5,692,215. Filed December 23, 1994.

[28] Y. Labrou and T. Finin. A semantics approach for KQML - a general purpose communication language for software agents. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*. ACM Press, 1994.

[29] J. Leikas, J. Salo, and R. Poramo. Security alarm system supports independent living of demented persons. *Proceedings of Gerontechnology Second International Conference*, pages 402–405, 1998.

[30] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan, and S. X. Zhang. A multi-agent system for intelligent environment control. In *Proceedings of the Third International Conference on Autonomous Agents*, 1999.

[31] MadKit, 2001. http://www.madkit.org.

[32] P. Maes. Concepts and experiments in computational reflection. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications Conference*, pages 147–155, 1987.

[33] Microsoft and the Universal Plug and Play Forum. *Universal Plug and Play Device Architecture.* http://www.upnp.org/download/UPnPDA10_20000613.htm.

[34] MIT Home of the Future. http://architecture.mit.edu/house_n/web/.

[35] M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *AAAI Spring Symposium on Intelligent Environments*, pages 110–114, 1998.

[36] OAA2, 2001. http://www.ai.sri.com/oaa/distrib.html.

[37] I. Pavlidis, V. Morellas, P. Tsiamyrtzis, and S. Harp. Urban surveillance systems: From the laboratory to the commercial world. *Proceedings of the IEEE*, 89(10):1478–1497, 2001.

[38] R. Penner and E. Steinmetz. Dynamic user interface adaptation based on operator role and task modeling. In *Proceedings of Systems, Man, and Cybernetics*, 2000.

[39] M. Pollack, L. Brown, D. Colbry, C. Orosz, B. Peintner, S. Ramakrishnan, S. Engberg, J. Matthews, J. Dunbar-Jacob, C. McCarthy, S. Thrun, M. Montemerlo, J. Pineau, and N. Roy. Pearl: A mobile robotic assistant for the elderly. In *Proceedings of the AAAI Workshop on "Automation as Caregiver"*, 2002.

[40] M. Raymond. Interaction design system use of xml. In *XML Conference and Exposition*, 2001.

[41] S. J. Richardson, D. F. Poulson, and C. Nicolle. Supporting independent living through Adaptable Smart Home (ASH) technologies. In *Human Welfare and Technology: Papers from the Human Service Information Technology Applications (HUSITA) 3 Conference on Information Technology and the Quality of Life and Services*, pages 87–95, 1993.

[42] S. J. Richardson, D. F. Poulson, and C. Nicolle. User requirements capture for adaptable smarter home technologies. In *Rehabilitation Technology: Proceedings of the 1st TIDE Congress*, pages 244–248, 1993.

[43] A. J. Sixsmith. An evaluation of and intelligent home monitoring system. *Journal of Telemedicine and Telecare*, 6:63–72, 2000.

[44] B. C. Smith. *Reflection and Semantics in a Procedural Programming Language.* PhD thesis, AI Laboratory, MIT, Cambridge, MA, 1982.

[45] F. Steenkeste, H. Bocquet, M. Chan, and E. Campo. La mise en place d'une technologie pour observer le comportement nocturne des personnes âgées en institution. *Innovation and Technology in Biology and Medicine – Revue of Biomedical Technology*, 22:25–30, 2001.

[46] T. Tamura, T. Togawa, and M. Murata. A bed temperature monitoring system for assessing body movement during sleep. *Clinical Physics and Physiological Measurement*, 9:139–145, 1988.

[47] Vigil Health Management http://www.vigil-inc.com.

[48] T. A. Wagner. Achieving global coherence in multiagent cargiver systems: Centralized versus distributed response coordination in i.l.s.a. In *Proceedings of the AAAI Workshop on "Automation as Caregiver"*, 2002.

[49] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[50] A. Yamaguchi, M. Ogawa, T. Tamura, and T. Togawa. Monitoring behaviour in the home using positioning sensors. In *Proceedings of the 20th annual IEEE conference on Engineering in Medicine and Biology*, pages 1977–79, 1998.

[51] Zeus, 2001. http://www.btexact.com/projects/agents/zeus/.