Hyperglue: Designing High-Level Agent Communication for Distributed Applications

Stephen Peters, Gary Look, Kevin Quigley, Howard Shrobe, Krzysztof Gajos
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA, USA
{slp,quig,garyl,hes}@ai.mit.edu,kgajos@cs.washington.edu

ABSTRACT

We[1] are building a new communication model and discovery system which will allow agent-based intelligent spaces to interact with one another. This new infrastructure layer, called Hyperglue, coordinates agent actions at a higher level than most agent communication does, providing an interface for communication at the level of "real-world" entities such as people, places, organizations, and information sources. The resulting structure is one which allows these agent communities to interact, while preserving the privacy, privileges, and preferences of the entities they represent. In this paper we describe the rationale for Hyperglue, and present the initial design as an extension of the existing Metaglue agent framework developed at the MIT AI Lab.

Categories and Subject Descriptors

C.2.4 [**Computer-Communications Networks**]: Distributed Systems—*distributed applications*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents, multi-agent systems*

General Terms

Design

Keywords

distributed agent systems, wide-area networks, service discovery

1. INTRODUCTION

Multi-agent systems for intelligent environments typically involve several smaller components working in concert. As the fields of ubiquitous and pervasive computing have grown in size, more and more of these systems have been brought into being, often integrating widely disparate devices such as hand-held PDAs, Bluetooth-enabled cell phones, laptops, speakers, microphones, huge wall displays, interactive tables, and the like. Most of these systems, including our own previous work in the field, have approached the interface and communication problems from either a device-centered approach, or have limited their focus to tying together many devices into a single environment. These limited scenarios fail to address the issues that arise when trying to coordinate among many spaces and many potential users. We believe that many of the existing multi-agent systems for smart environments do not possess the necessary architecture to handle these situations gracefully.

We begin this paper by presenting a short scenario, and exploring some of the problems inherent in even simple interactions involving multiple spaces and people. After presenting the current state of multi-agent systems in smart spaces, we will describe our approach to extending an existing agent framework with a new communication model that can better address these issues, based on resource management and a simple global discovery system.

2. SCENARIO

Consider a straightforward application of agent systems in augmented environments, similar in appearance to the world envisioned by Weiser [11] or Dertouzos' vision of Project Oxygen [4]:

> Alan is sitting in his office, reading his e-mail and perusing articles over the Internet. Beth, his co-worker, is sitting in one of her office's workrooms, and her current task is shown on one of the room's two large displays. Beth also has her handheld computer with her.
>
> Alan comes across an article that he wants to bring to Beth's attention, so he asks his office's agent system to open a communication channel to her. Beth's communication agents inform her that Alan wishes to talk, and asks if she wants to accept the request. When she agrees to a video connection, it displays Alan's image on the unused display. After a brief discussion, Alan asks his office to forward the article to Beth, at which point the information appears on her handheld.

[1]Both Stephen Peters and Gary Look are currently Ph.D. students at MIT.

AAMAS '03 Melbourne, Australia

3. THE PROBLEMS

Although the preceding scenario seems simple, it gives rise to complex questions about how such agent systems communicate, especially when Beth might not be sitting in a nice workroom, but in her office, in her car, or even taking a pleasant stroll through a neighborhood park with a PDA in her pocket. Alan's desire to communicate and transfer information to Beth does not change based on her location, although the methods by which information is conveyed may change based on the devices currently available to the two of them. In order for an agent operating on behalf of Alan to communicate to Beth, several questions must be addressed:

1. *What devices are available in Beth's vicinity?* Several methods have been proposed for answering this question, but they often fall broadly into two categories. In systems that operate on a single-request model, such as MIT's Intentional Naming System [1], all devices that are in some way associated with Beth advertise that association to a global network, so that a single request can retrieve all of the appropriate recipients. Alternately, the methods can use two or more requests to get the information – Beth's software constantly advertises her location to the network, and Alan's agents must first look up her location, and then perform a separate search to find a set of candidate devices. Both of these methods require that all devices be advertised globally, and that Beth's location be advertised and updated to reflect Beth's current situation.

2. *How can Alan and Beth share control of the interaction?* Once the agents working for Alan have located a device for the communication, we still need to make sure that Beth, or agents working on her behalf, can maintain some control over the interaction. Just as we accept an incoming phone call by picking up the receiver (or by pushing a button in this more cellular age), Beth must have the ability to decide whether or not to accept an intrusion into her work. This ability should also extend to the point of being able to automatically refuse all interruptions for a period of time. In a device-centric model of the world, Beth would need to broadcast her preferences to the local devices, so that they would know whether to allow an incoming connection.

3. *How do we avoid conflicts with Beth's work?* Note in the scenario that Alan's image appears on the *unused* display, so as not to overwrite the work that Beth is doing, and thus allowing Beth to make the transition from work to the conversation at her own pace. In addition, when Alan transfers the news item, it gets sent to the last remaining display in Beth's locale, her handheld. But how can Alan's agents discover which display is available for use? If the agents are using a global directory service to find the available display devices, then Beth's display devices must be constantly updating their availability status so that Alan's agents can choose the correct resource. And what if Beth's handheld is unavailable? How can Alan's agents decide where to display the news item?

4. *Who decides how information is presented to Beth?* She may have strong preferences about how she processes information; for example, a short text sentence might best be conveyed as spoken utterances rather than through a display. If she's in her car, she may prefer that all information be emailed to her destination where she can peruse it once she's arrived safely. Alan's agents would need to be able to discover these preferences, and alter their behavior accordingly. Doing so would require that Beth make those preferences publicly available to any agent system making communication requests.

5. *How does the agent system take into account the environment's access control issues?* Irrespective of what the agents for Alan and Beth wish to do, the shared workroom will need to have mechanisms in place to prevent unauthorized access to the devices by malicious agents. Imagine the intelligent environment's equivalent of junk mail in which marketers are perfectly content to pop up advertisements on any available display space. The agent system needs to be able to filter out any unwanted requests, while at the same time giving Beth the necessary privileges to do her work and accept connections and data from Alan.

   Privilege is a double-edged sword; the data Alan is sending might be confidential or otherwise protected, and thus should not be displayed on the walls of a public, shared workspace like Beth's. In this case, Alan's agents must be able to mark the information as being removed from public consumption, and then have that request respected by the devices on the far side.

As can be plainly seen, there are numerous issues that need to be addressed in even the most simplistic pervasive computing application. Many of these issues can be addressed within existing agent infrastructures that rely on a global directory service or shared blackboard architecture. However, to do so requires that nearly all of the above information must be registered in the directory, and then updated constantly to adapt to changing conditions. In addition, such an approach could have rather substantial privacy and security implications, as seen above. Finally, any centralized directory will eventually need to overcome scalability issues when it is extended to large numbers of people and spaces, since the directory would have to maintain information on every person, space, display, PDA, projector, speaker system, and basically any other device that can be integrated into the agent network. Sifting through this list to find the best fit for a given situation becomes more and more expensive as the magnitude of the directory grows.

4. EXISTING MULTI-AGENT SYSTEMS

The Open Agent Architecture [9] one of the more venerable agent systems available, is designed specifically for agent-based interactions in smart spaces. Within OAA, all communication is funneled through a central agent called the *facilitator*, which acts as a coordinator for cooperative problem-solving. Agents register their capabilities with this central agent, and make requests in the form of "tasks," which the facilitator can break down into subtasks and farm them out to client agents that can perform the required actions. OAA lacks mechanisms for resource reservation or for arbitration among conflicting requests, which could make it hard for an OAA system to address the issues inherent in avoiding clashes with ongoing work.

The SmartPlatform infrastructure is currently being used by the Smart Classroom project [12]. This project originally was based on OAA, but recently moved to a new approach, featuring a hybrid communication scheme: short messages are sent through a central broker (the "DS", or directory service) using a publish/subscribe mechanism, but peer-to-peer connections can be created when higher bandwidth requirements are necessary. SmartPlatform consolidates all messages into two kinds of speech acts: "inform" and "query", for asynchronous and synchronous messages respectively. SmartPlatform itself does not contain facilities for handling or processing resource conflicts, and is beginning to work on gateways to other agent systems (such as Metaglue, described below), to help address some of the scalability issues.

The IHome project at UMass [8] was a simulated agent architectures for intelligent environmental control, emphasizing resource coordination. High-level agents were capable of recognizing resource conflicts, and prioritizing allocations according to the user's preferences. IHome's agents, however, are designed to work with one space, and there does not seem to have an easy way of coordinating operations across multiple environments or users, making it problematic for use as a communication substrate between intelligent spaces.

one.world [7] is a system architecture developed at the University of Washington, designed to provide programmers with services for writing pervasive applications. The distributed components use remote event passing for communications, and perform discovery operations to locate resources for the components. The discovery server is centralized, but is elected in an ad-hoc manner from all the participating nodes, with the elections waited to favor candidates with the best response times. Once an election has taken place, all the nodes pass service information into the discovery server, and use this single point for most communication handling. one.world includes support for low-level resource allocation, but does not appear to have a generic way for modeling higher-level resources or handling conflicts. Because all components are also sharing the discovery server, there is a requirement for participating components to share information so that they can be discovered properly.

4.1 Metaglue

Metaglue [3] is our multi-agent system implemented in Java, developed and used by the Intelligent Room project. It also contains some of the problems of the previous approaches, in that all agents must use a centralized directory service called the *catalog* in order to locate other agents and communicate with them. Metaglue is based on peer-to-peer method calls between remote agents, using Java's RMI (Remote Method Invocation) mechanism. All agent-to-agent communication is further channelled through special proxy objects [10], which can be used to handle failover conditions when an agent goes down or moves to another location in the network. The looser coupling of agents that feature provides allows agents to handle changing network situations, without requiring all agents to explicitly support that feature.

In addition to these synchronous method calls, Metaglue also provides mechanisms for publish/subscribe communication with many of the same features provided by the proxies, persistent storage, multimodal input and output, user customization, and automatic restarting of failed components. Metaglue also provides the means to segment agents into small communities called "societies," which can serve as a convenient way to divide large agent networks into smaller, separate groups of agents coordinating on a task. As we see below, we use this capability to separate agents according to the real-world "entity" on whose behalf they operate, whether that entity be a person, a place, or even a less concrete notion like an organizational group.

Metaglue's architecture also allows for a "plug-in" resource management system, under which agents can forward all agent requests through a broker that can translate the request and locate a service agent or set of agents that satisfy the request. Resource managers can determine the best agent for the request, and even put an service on "reserve" so that other agents can't make requests of it. They can also make use of the agent-swapping ability mentioned earlier to alter the resources that an agent is using (for example, switching an agent's output connection from spoken utterances to text on a display.) The resource management system has proven to be a real boon to agent development, allowing agent authors to worry only about the broad interfaces required by their agents, and less about the details about which agent will be retrieved by a given request.

Since the resource management system is only loosely coupled to the agents, different agent societies can use different resource management subsystems. Resource managers can even coordinate together by forwarding an agent request on to another resource management system, and then merging the result with its own agent decisions before passing them back to the original requester. Resource management systems can also provide the basis for a strict access control model, since they can coordinate with a local security agent when making decisions, without introducing unnecessary complexity in the agent code.

The Metaglue system has been deployed in a number of locations around the MIT AI Lab, including a small conference room, a living room, and a set of student and faculty offices. Trying to integrate these rooms together, so that users could have control over their environment while dealing with issues of scalability, privacy, personal preference, et cetera, raised a number of issues with the Metaglue communication architecture, whose centralized catalog made it difficult to separate out these concerns properly. Indeed, it was this need for integration that was the impetus behind the effort to extend Metaglue into the Hyperglue framework.

5. HYPERGLUE

The Hyperglue communication model provides a communication interface between agents, situated at the level of "real-world" entities such as people, places, and organizations. In this section, we illustrate how an agent that is a member of one agent community might use resource management systems and Hyperglue's entity discovery capability to communicate with an agent in a different agent community. Subsequent sections will provide a more detailed description of Hyperglue's discovery mechanism.

Take two agent societies, which need to communicate with each other (see Figure 1). These can be operating on behalf of two users trying to share information, or a person trying to gain access to the devices in a shared environment, or any of a number of other scenarios. When an agent in one society needs to request an agent in another society, it

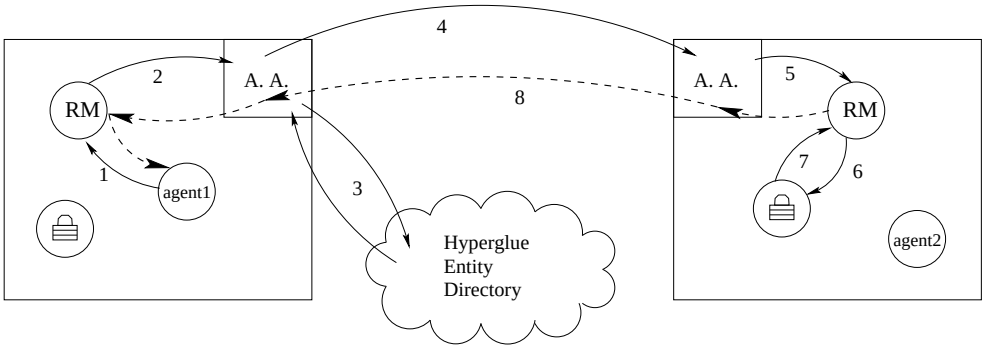

Figure 1: Hyperglue in operation, coordinating between two societies. 1. An agent (Agent1) makes a request for a service agent that is unavailable in the current society. 2. The resource manager (RM_A) receives the request, and forwards it on to the local Ambassador agent (AA_A). 3. AA_A queries the Hyperglue Entity Directory (HED) to find the remote society's Ambassador (AA_B), and receives a handle for communicating to it. 4. AA_A forwards the request to AA_B. 5. AA_B then sends the request on to its local resource manager (RM_B). 6. RM_B takes into account all agents that can handle the request, taking into account any preference information, and also consulting with the local security manager. 7. If the security and preference systems authorize the connection, RM_B returns a stub to AA_B. 8. AA_B returns the stub to the AA_A, which can then pass it on to the original requester.

first will contact a local resource manager. Note that even though the request is for an agent in a different society, the local agent still views the remote agent as merely another necessary resource, and can delegate the task of finding the appropriate component to the local resource manager.

The resource manager will then determine that the request involves the remote society (perhaps simply because it doesn't have any information on the kind of agents being requested), and will then forward the request on to the Ambassador for the local society. The Ambassador agent acts as a proxy for its society to other agent societies, and is registered on startup with a global directory system called the Hyperglue Entity Directory (HED). Like real-world ambassadors, the Ambassador's function is to represent the agents in its society to other societies of agents. This includes sending out requests for handles of agents in other societies as well as receiving requests for handles to agents in its society.

When the local Ambassador receives the request, it consults the HED for the location of the Ambassador agent for the remote society. It can then contact the remote Ambassador and pass on the request. The remote Ambassador can then pass the request on to its own resource manager, which can make the determination about how to fulfill the requirements (if at all), and pass back a handle for any found resources to the local side through the Ambassadors' connection, and finally back to the original requester.

One of the nice features of this approach is that agents treat inter-society requests the same way that they treat requests within the society, easing agent programming greatly. The entire process of requesting remote agents is hidden behind the call to the local resource manager, so the Hyperglue system's inner workings are behind the "abstraction barrier" presented by the resource management framework. In addition, the inner workings of agent societies are encapsulated into the higher-level societies, easing the burdens put on the HED during the discovery phase.

5.1 Benefits and Applicability

Let us briefly return to our earlier scenario, and the questions asked in section 3. In our scenario, when Alan requests a communication channel to Beth, his software agents first reserve the necessary devices in his office so that he can be heard and seen. They then request from Alan's resource manager a connection to Beth, preferably one with audio and video capabilities. Alan's resource manager will then forward the request to the Ambassador, which will perform a discovery operation and locate Beth's Ambassador. The Ambassador agents forward the request to Beth's resource manager, which can then return any agents that are suitable (taking into account Beth's preferences). If it's allowable, Beth's resource manager can return a handler to a communication manager operating in Beth's society.

Alan's agents can then talk to Beth's communication manager agent, which in turn would talk to Beth's resource manager to request audio and video devices for Beth. Her resource manager, realizing that it only has a PDA with limited audio and video bandwidth within its own society, could then take advantage of the fact that it has knowledge of Beth's location, and forward the device request to the workroom's society. The workroom's resource management system can make similar determinations about Beth's access rights, and eventually decides to pass a handler for the remaining free display back to Beth's agents.

Later, when Alan is trying to show the news item to Beth, her resource manager would again check with the workroom for available resources, but be unable to receive any since both displays are now taken. In this case, Beth's resource manager might decide that the limited video on the handheld was suitable for the information and use it.

We can now turn to the previous set of problems, and examine how well this approach handles them:

1. *What devices are available in Beth's vicinity?* Alan's agents now do not need to have this knowledge at all. All his agents need to do is discover the entry-way

into Beth's society; once this is accomplished it is the responsibility of Beth's agents to discover where Beth is located and the various devices that are available to her.

2. *How do Alan and Beth share control of the interaction?* Since the request for an interaction now goes directly from Alan's agents to Beth's, the agent software operating on their behalf has total control over the connection. Beth can tell her agents to ignore all incoming requests for a period of time, or her agents can respond to any incoming interaction request with a confirmation request, all according to Beth's preferences. Beth does not need to broadcast a "hold all calls" request to her current environment.

3. *How do we avoid conflicts with Beth's work?* The resource management systems oversee the devices that Beth is currently using, and can use this information to decide which devices are available and decide on the best option for Beth.

4. *Who decides how information is presented to Beth?* Again, this ability is mostly in Beth's hands (or the "hands" of her agents, rather). Her resource manager can decide whether to use her personal devices for displays, or whether to contact the environment for other possibilities. The communication agent in her society can decide whether to accept the communication with Alan, and even decide how to handle the communication – perhaps by using only an audio connection instead of both audio and video. In addition, her agents could decide whether the news item was displayed, or could have decided to just speak the text aloud if that were preferable.

5. *How does the agent system take into account the environment's access control issues?* Alan has no connection to the workroom's devices; they are all processed through Beth's agents and requested on her behalf, not Alan's. The workroom's resource manager and security manager can decide on its own whether to accept the request without having to worry about whether Alan should be given access, since only the trusted Beth is performing requests.

5.2 Current Implementation Details

The current implementation of Hyperglue uses the Intentional Naming System for its entity discovery component. INS is a global resource discovery system, in which participating resources advertise capabilities and identity information as opposed to just network addresses. For example, a printer could advertise its location, basic capabilities, and current queue length to the INS, which can then use this information to route queries such as "find me a lightly-loaded color printer nearby." The system should, in principle, be able to scale past a single organization; however, the version of INS we were using advertises all devices throughout the entire network, and thus raised privacy and scalability concerns.

As noted before, INS is not viable as the sole resource discovery mechanism, but is successful in its use here as a dynamic, wide-scale system for locating the ambassadors of entity-based societies.

The resource management system being engineered for Hyperglue is Rascal [6], a knowledge-based system that is capable of making high-level arbitration decisions among service requests. It includes a constraint satisfaction engine (CSE), so that once a service has been requested by an agent, it is capable of maintaining that connection until it is either canceled or until a higher-priority request overrides it.

When Rascal starts up in a given society, it loads information about all the society's local devices and services into its knowledge base. This information is dynamically updated as more resource information becomes available. Agents that provide services can also describe what other resources they, in turn, will need – for example, the `Messenger` agent which can deliver information to the local user needs to have one or more agents capable of providing text output (through speech, LED screens, cell-phone displays, or whatever else is available). Agents can also specify startup needs as well, such as noting that the `PowerPointDisplay` agent is going to need a computer with the appropriate software available.

Rascal uses its knowledge base to find all possible candidate agents for the requests, and then uses the CSE to decide which of those requests is "best" for the current situation. Part of this decision is performed by assigning ordered values for a service's utility (how useful it is for the given request) and for its cost to others. This is a simple model, sufficient for the current incarnation of Rascal, but could eventually be replaced with a market-based system (such as that described in [2]) if the need arose. The value calculations depend on the assumption that each candidate resource has a utility to the requester, based on how badly the requester needs the request fulfilled, and on how well the candidate matches the request. Since resources that are already allocated have similar utility values to their own requesters, there is a proportional cost associated with switching the resource to another agent, which gets factored into the constraint satisfaction system when evaluating the candidates.

5.3 Future Work

We are currently working on implementing the second version of the Hyperglue Entity Directory service. This will entail the plan described above, including slots available for resource management and security, filling out the Hyperglue trio. There are several organization models currently under consideration.

1. The Intentional Naming System was used in the first version of Hyperglue with mixed results. It would be good at providing the society discovery component robustly. By only providing the societies' locations, or more specifically the location of each society's Ambassador, we can avoid the problem of resources being broadcast without restriction.

2. A hierarchical structure modeled after the Domain Name System (DNS) used in associating the name of a computer with the IP address. This has been shown to be both robust and widely scalable. With this architecture, each sub-level Hyperglue layer would appear as a single entity (similar to a society) to the current Hyperglue level. Such a stacking would produce a tree structure that is easily traversable.

3. A data-driven model where the requests for resources and communication are labeled with descriptions about

the intended recipients and Hyperglue would route them through the network. *Data-centric networking* is described in papers on the Portolano project [5].

Each of these models has strengths and weaknesses so it may be that none alone can provide a robust and rich enough environment to support Hyperglue. INS requires that the societies constantly refresh the alive status or be dropped which is a waste of computational resources for a static structure like a Hyperglue enabled building. A DNS-like lattice would provide a more static structure which in turn would have trouble modeling social groupings which change frequently as people move and groups form and break. It should become apparent that several models (graph based, tree based, publish-subscribe) will make the grouping of societies easier. This does increase the complexity by having multiple parent nodes to query, but as long as the correct society is found, it doesn't matter which node returns the information first.

An intriguing result of the current design for Hyperglue is that the details of the agent society are abstracted away through the Ambassador agents. Conceivably, it should be straightforward to integrate a network utilizing a different agent system, like one.world, OAA, or the like into the Hyperglue model, simply by creating an appropriate ambassador that can consult the HED for remote societies and translate requests and agent handles into the local framework. Such a setup would allow totally different agent networks to still communicate with each other. An effort is currently underway to allow a CORBA interface into the agent model, which would simplify any effort along these lines.

6. CONTRIBUTIONS

We have pointed out a number of ways in which many existing architectures for agent systems in smart environments have difficulty in fulfilling their promise. In particular, we have noted the problems associated with privacy, access rights, and personal preferences that arise when these systems take too much of a device-centric focus, or focus too much on the needs of a single-user, single-environment model.

In addition, we have proposed a framework that can be built on top of existing agent systems, and can, we submit, be used to address these problems. The framework design is based on the use of resource management and a simple high-level discovery system for locating agent networks that operate on behalf of real-world entities such as people or environments. Our current work in developing this system suggests that it is a viable approach to these systems, and we will be continuing to develop and refine this project in the future.

7. ACKNOWLEDGMENTS

We'd like to thank the members of the AIRE project at the MIT AI Lab for their ever-present assistance and constructive suggestions, and in particular Krzysztof Gajos, who helped describe many of the necessary attributes of scalable agent systems.

8. REFERENCES

[1] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *17th ACM Symposium on Operating Systems Principles (SOSP)*, Kiawah Island, SC, December 1999.

[2] J. Bredin, D. Kotz, D. R. R. T. Maheswaran, Çagri Imer, and T. Basar. A market-based model for resource allocation in agent systems. In F. Zambonelli, editor, *Coordination of Internet Agents*. Springer-Verlag, 2000.

[3] M. Coen, B. Phillips, N. Warshawsky, L. Weisman, S. Peters, and P. Finin. Meeting the computational needs of intelligent environments: The Metaglue system. In *Proceedings of MANSE'99*, Dublin, Ireland, 1999.

[4] M. Dertouzos. The future of computing. *Scientific American*, 282(3):52–63, August 1999.

[5] M. Esler, J. Hightower, T. Anderson, and G. Borriello. Next century challenges: Data-centric networking for invisible computing. In *Mobile Computing and Networking*, pages 256–262, 1999.

[6] K. Gajos. Rascal - a resource manager for multi agent systems in smart spaces. In *Proceedings of CEEMAS 2001 (LNAI 2296)*, pages 111–120. Springer-Verlag, 2001.

[7] R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall. Programming for pervasive computing environments. Technical Report UW-CSE-01-06-01, University of Washington, Department of Computer Science and Engineering, 2001.

[8] V. Lesser, M. Atighetechi, 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 (Agents99)*, Seattle, WA, 1999.

[9] D. L. Martin, A. J. Cheyer, and D. B. Moran. The Open Agent Architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1-2):91–128, January-March 1999.

[10] N. Warshawsky. Extending the Metaglue multi agent system. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

[11] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991.

[12] W. Xie, Y. Shi, G. Xu, and Y. Mao. Smart Platform – a software infrastructure for smart space (SISS). In Submission.