SANDIA REPORT

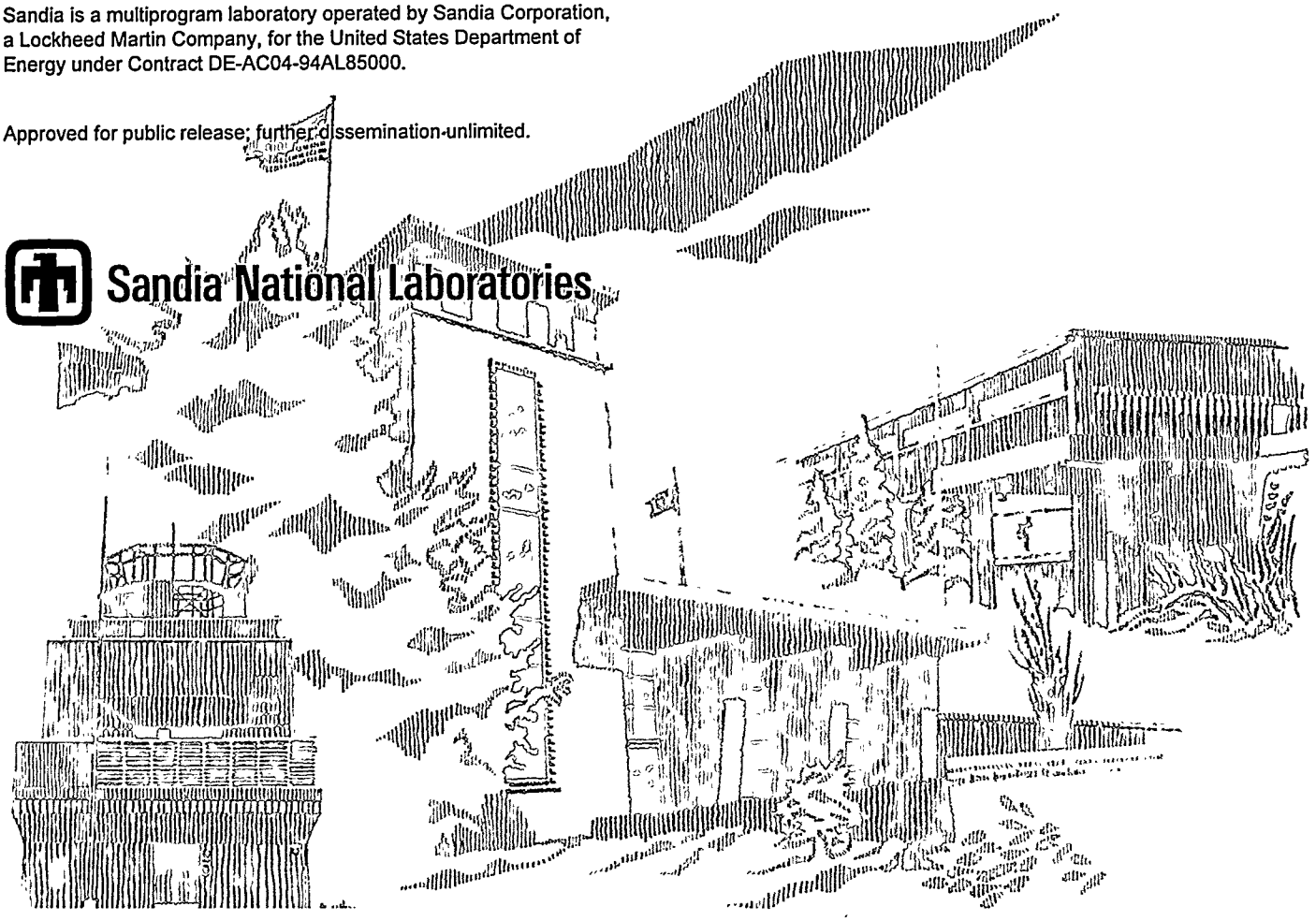
SAND98–8668
Unlimited Release
Printed December 1998

Agent-Based Enterprise Integration

N. M. Berry and C. M. Pancerella

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SAND98-8668
Unlimited Release
Printed December 1998

Agent-Based Enterprise Integration

Nina M. Berry and Carmen M. Pancerella[1]
{nmberry, carmen}@ca.sandia.gov
Sandia National Laboratories
P.O. Box 969, Mailstop 9012
Livermore, CA 94551-0969

Abstract

We are developing and deploying software agents in an enterprise information architecture such that the agents manage enterprise resources and facilitate user interaction with these resources. Our enterprise agents are built on top of a robust software architecture for data exchange and tool integration across heterogeneous hardware and software. The resulting distributed multi-agent system serves as a method of enhancing enterprises in the following ways: providing users with knowledge about enterprise resources and applications; accessing the dynamically changing enterprise; intelligently locating enterprise applications and services; and improving search capabilities for applications and data. Furthermore, agents can access non-agents (i.e., databases and tools) through the enterprise framework. The ultimate target of our effort is the user; we are attempting to increase user productivity in the enterprise. This paper describes our design and early implementation and discusses our planned future work.

[1] This work was supported by Sandia Corporation under Contract No. DE-AC04-94-AL85000 with the U.S. Department of Energy.

Table of Contents

1 INTRODUCTION

Industries must adapt to the distributed information age. The Internet is ubiquitous, the amount of electronic information is growing daily, software systems are more complex than ever before, resources and information are distributed, and the economy is global. People are working across geographic and organization lines, often for just a short-term project, such that data and tools are shared across a virtual enterprise. In such an enterprise there are diverse users (e.g., suppliers, contractors, customers, managers, and partners), and heterogeneous tools, resources, and databases, distributed across a wide area. Furthermore, the enterprise is dynamic.

Given the dynamic and distributed nature of an enterprise, there are demands placed on the enterprise architecture. Specifically, it must support a variety of platforms, operating systems, and programming languages. It must support rapid and easy customization, integration, and reconfiguration. The architecture must allow legacy software to be integrated. The software must be easy to deploy throughout distributed facilities. A wealth of information and a plethora of tools must be accessed throughout the enterprise. Intelligent software components must be developed to manage the various resources in the enterprise. Intelligent components must be able to determine how and where to get information and tools to users. Users must be able to locate and combine resources and data. Users must be able to work together as teams across time and space. Finally, knowledge must be exchanged between various domains and organizations.

Enterprise integration architectures or frameworks have been developed, and these typically focus on interoperability, tool integration, and scalability rather than user integration, intelligent management of resources, and knowledge exchange. One such framework, the Product Realization Environment (PRE) framework [WhFD98] developed at Sandia National Laboratories, has been developed for integrating the numerous heterogeneous electronic resources across Sandia's laboratories and other Department of Energy (DOE) facilities. PRE is enterprise software that has been deployed for use in production. This framework provides solutions to many of the challenges listed previously but does not address issues of adaptability, knowledge exchange, and true integration of distributed resources and end-users.

We believe that in order to enhance existing enterprise architectures, agents can be developed and integrated into the architectures. Software agents are adaptive and intelligent software components; they can respond to users and the state of an environment, act on behalf of users, and can coordinate users and other software components in order to accomplish a goal. Agents can monitor resources, access data, and present the dynamically changing enterprise to end-users. They provide a viable solution to the missing pieces in existing enterprise architectures. In fact, [Paru97] lists autonomous agents as a core technology for virtual enterprises.

We expect that the marriage of enterprise frameworks and intelligent agents will provide robust enterprises with rich user interaction possibilities. Software agents can build upon the capabilities of distributed enterprise frameworks (like interoperability, security, and object-oriented abstractions) but add unique agent properties, like autonomy, reasoning, and agent coordination, to the enterprise.

We are currently in our first phase of the system development. We are building our communicating agents on top of PRE and are using a subset of KQML [FiWe94] as an agent communication language. This paper focuses on our system design decisions, agent architecture, and the expected benefits of our agent-based enterprise integration. Since we are working within a functional

enterprise at Sandia, it is important that we justify our design decisions, apply good software engineering techniques for enterprise systems, and address many of Wooldridge and Jenning's pitfalls of agent-oriented systems [WoJe98].

In Section 2 we review enabling technologies and related work in enterprise integration, including a short overview of the PRE framework. In Section 3 we present our agent architecture, and in Section 4 we give specific details on the agents we are developing. In Section 5 we show various scenarios that are possible in the agent-based enterprise. In Section 6 we discuss some of our agent implementation decisions. In Section 7 we review our initial experiences with the enterprise agents. Finally we summarize our contributions and discuss future work in agent-based enterprise integration.

2 BACKGROUND

We draw upon work in software engineering for agent-based systems, enterprise architectures and multi-agent systems.

2.1 Software Engineering Issues in Enterprise Agent-Based Systems

We are using the integration issues in [Panc98] and many of the pitfalls of agent-based systems [WoJe98] to motivate our design decisions. [Panc98] lists software integration issues that should be considered when developing an enterprise architecture; this list includes heterogeneity, support for distributed applications, interoperability, extensibility, integration with existing enterprise software, agent coordination, and agent communication protocol. Wooldridge and Jennings enumerate a number of pitfalls of agent-oriented development and discuss the pragmatics of agent-based development in [WoJe98]. We use a number of these pitfalls throughout the paper to guide our design decisions. In particular, we are attempting to avoid pitfall 3.4 *confusing prototypes with systems*, pitfall 5.1 *not exploiting related technology*, and pitfalls 4.3 and 4.4 *forgetting that this is distributed software development* in our development.

2.2 Related Work in Multi-Agent Systems

Most multi-agent systems to date address problems at the domain level (e.g., [CuEF93], [PaHF95], [FrCu96], and [WePr98]) and not at the enterprise level, and some systems which have attempted to address enterprise systems [PaTe92] have been prototype demonstrations and not production quality software.

One of the earliest agent frameworks for enterprise computing was defined by Pan and Tenenbaum [PaTe92] in 1991. Though we share the vision and goals of this framework, this framework does not consider the interaction of agents and non-agents, i.e. tools and databases; is only a prototype system (and not a production system); and does not discuss how agents can be developed on top of an existing enterprise framework. Our work addresses these issues.

KAoS [BrDB97] is an open distributed architecture for software agents. Like our approach, KAoS is designed to take advantage of commercial distributed object technologies such as CORBA. KAoS is not, however, built on top of an enterprise framework.

Barbuceanu and Fox [BaFo94] discuss the design of an information agent to be used in collaborative enterprise architectures. Other researchers have explored information agents in dynamic, information-rich environments, for example, InfoSleuth [BaBB97] and SIMS [KnAm97]. These

projects did not address many of the enterprise issues. We will build upon the experiences of these information agents as we design our information agents, resources agents, and agent communication.

2.3 PRE

The PRE framework is a software architecture developed at Sandia for tool integration and data exchange. PRE is a lightweight framework for a broad variety of electronic resources (for example, databases, product data management systems, modeling codes, and engineering advisors). PRE is built on top of CORBA [CORB95] and available vendor-developed CORBAServices. Applications are "wrapped" for use in the PRE framework as reusable components. PRE has a common API which all applications share; hence, in order to "wrap" an application into PRE, a developer simply implements the framework-compliant API in order to expose the functionality of the application.

The PRE architecture consists of several major pieces including uniform data objects and transport, a trading service, security, a conversion broker, integrated or wrapped applications, and user interfaces. Table 1 gives a quick definition of each component, and Figure 1 is a picture of this same architecture.

Table 1. Major components of PRE.

Component	Description
Uniform data objects and data transport	PRE provides a persistent, uniform data container for structured information. Data/information is exchanged between PRE applications and services via data objects. PRE has a file manager to handle very large data files efficiently.
Integrated applications	PRE provides a standard application API for wrapped applications. An application's functionality is accessible to PRE clients and other PRE applications through this wrapper.
User Interfaces	User interfaces can be developed as stand-alone applications, applets, or web front ends to PRE core services and applications.
Trader Service	PRE provides a 'yellow pages' registry for the location and general information about the applications integrated into PRE.
Conversion broker[2]	This PRE service employs a reasoning algorithm to determine the data translation steps necessary to convert data from one format to another, given a registry of PRE converters.
Security	PRE security supports a variety of security models, selected by plug-able library modules.

[2] The conversion broker is the only component in PRE that behaves like an agent.

The data factory, distributed file manager, trader, conversion broker, and security are part of the PRE core services. At this time, there is no federation of conversion brokers or traders; thus, a single trader knows about a subset of the enterprise resources, since there may be many traders in an *extended enterprise*. To date many applications have been wrapped as PRE servers. These include financial and employee databases, product data management systems, modeling and simulation tools, and many data converters which convert to and from various data formats. All of these resources will be available to agents in our system. The strength of building upon the PRE framework is that all PRE servers will be accessible to agents through a common application interface.

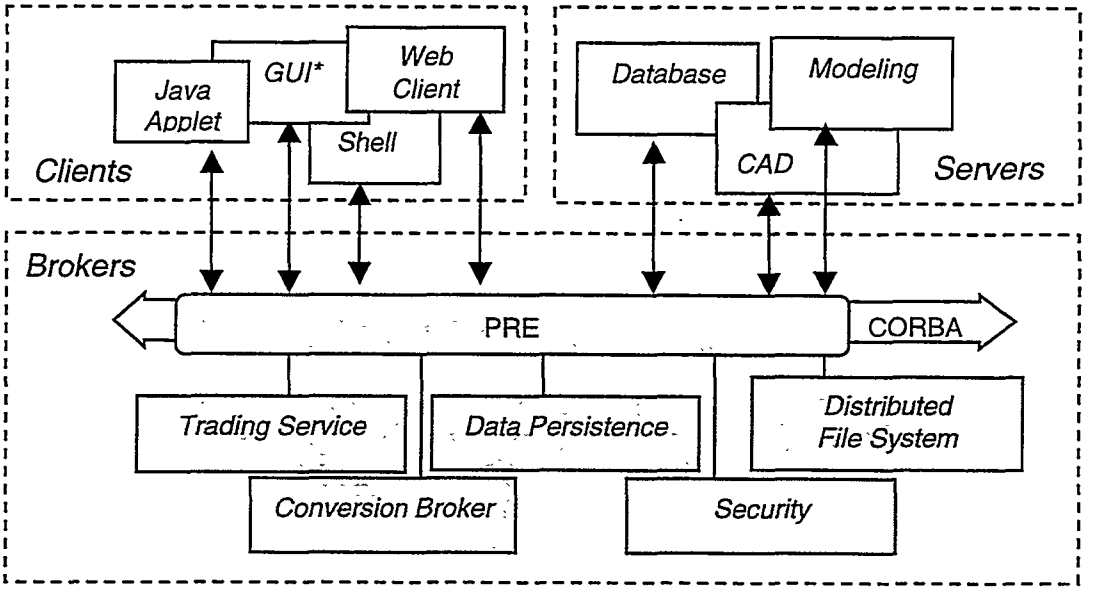


Figure 1. PRE architecture.

Software agents can be built on top of the PRE framework to solve the complex and dynamic problems that arise in a truly integrated enterprise. The result is a powerful enterprise infrastructure. While the PRE framework itself cannot be considered an agent-based system, the infrastructure provided by PRE does provide many underlying requirements necessary for agent-oriented systems to function correctly.

3 AGENT ARCHITECTURE

The first phase of our agent architecture development is illustrated in Figure 2. We view this architecture as a non-intrusive layer of information-centric agents that operate on top of the existing PRE infrastructure and PRE applications. Agents can communicate with other agents in an agent communication language (in our case, KQML), can utilize PRE core services, and can access other electronic resources in the enterprise framework. We note that FIPA has a standard for the communication between agents and non-agents; their approach is to "wrap" non-agents in a FIPA-compliant wrapper agent [FIP397]. Our approach differs in the sense that many resources have already been "wrapped" as enterprise applications, and hence, agents can connect to these non-agents through their PRE-compliant wrappers. Hence, we are eliminating the need for another "wrapper" by using the existing enterprise infrastructure. Our philosophy is that there is no need to make each of these electronic resources an agent. Where it makes sense, we will create agents.

The result of our agent architecture is that the enterprise is enhanced with location of resources and data across extended enterprises, extraction of knowledge about these resources, and better integration of users and knowledge in the underlying enterprise framework. The agent layer is built on top of the existing PRE enterprise framework, thus permitting the agent architecture to utilize the existing infrastructure. All networking and distributed systems issues are addressed through the enterprise framework. This approach avoids committing pitfall 7.4 *spending all your time implementing infrastructure* [WoJe98].

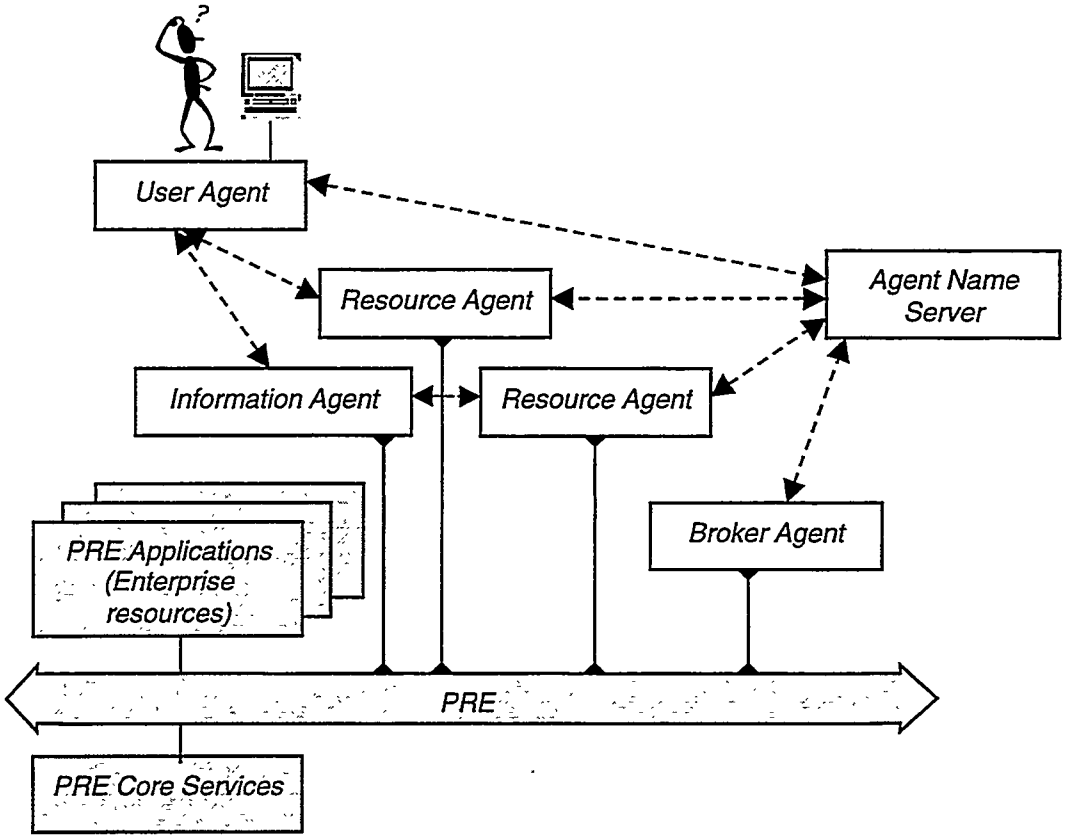


Figure 2. Agent architecture.

As shown in Figure 2 users have access to the enterprise framework through some GUI, possibly within a browser. As users request information and navigate through the enterprise, personalized *user agents* are interfacing directly with the users through this GUI. A user agent assists the user with his/her request. The user agent can also provide personal access to the information, data, and tools that are available in the enterprise (e.g., employee databases, data in a PDM, modeling codes, converters, etc.). User agents typically communicate with resource agents, information agents, and/or broker agents in order to accomplish their goals, but they can also access data sources directly and synthesize data from multiple sources. The other agents in the agent layer are resource agents, information agents, and broker agents. While many textbooks and papers use the term *resource agent* as an agent that simply interfaces to one or more electronic resources, we use this term instead to categorize agents that reason and maintain information about collections of electronic resources (e.g. modeling codes, conversion codes, etc.). Similar to a resource agent, an *information agent* is an agent that reasons about data in the enterprise. A *broker agent* is a special type of agent that is designed to

operate with enterprise core services across several enterprises. Broker agents dynamically gather information about core services in an extended enterprise (e.g., several trading services or several conversion brokers) and provide other agents with this information. The functionality and interactions of user agents, resource agents, information agents, and broker agents will be discussed in greater detail in the following sections. We imagine that the number and type of agents in our enterprise architecture will change in time; this paper describes our first phase.

3.1 Agent Name Service

Upon startup, each agent must register and advertise its capabilities/services with the agent name service (ANS) utility. Likewise when an agent leaves the system, it must unregister with the ANS. The ANS gives agents a method of locating other agents indirectly based on their capabilities or name without the knowledge of a specific network address. To locate the available electronic resources in the enterprise, an agent directs queries a trading service, an enterprise core service. By utilizing an ANS, the amount of repetitious and inconsistent information being propagated within the agent layer is limited. The concept of using an ANS is common among many agent architectures, for example, Retsina [RETS98], JATLite [JATL98], and the FIPA standards [FIP197]. Our decision to use an ANS is driven by our desire to avoid pitfall 8.2 *ignoring de facto standards* [WoJe98]. We are adopting techniques and practices commonly used in other agent development projects.

Our particular ANS is more than simply a router as in JATLite. It functions as a matchmaker or facilitator does in other agent architectures [GeKe94] [DeSW97].

As mentioned earlier, agents communicate with each other and the ANS in KQML. The communication process begins with a sending agent asking the ANS to direct an enclosed message to an agent with a given capability. The ANS checks its registry and the message is delivered to a receiving agent. Each message begins with a KQML performative, examples of agent messages include registering an agent's location, advertising an agent's capabilities, querying another agent, and general information exchange between agents. Our decision to use KQML further avoids committing pitfall 8.2 *ignoring de facto standards* [WoJe98], since KQML has been deployed in some fashion in numerous other agent projects. We discuss our layered implementation in Section 6.

3.2 User agent

Each user has an individual user agent, which is an intelligent interface to the resources of the enterprise. A user agent can communicate with other agents or can connect directly to the resources of the enterprise. The user agent acts as a buffer between the user and the enterprise, thus controlling all interaction between these entities. Employing users agents facilitates (i) intelligent reasoning to accomplish the user's goals; (ii) translating user queries into appropriate agent messages and PRE method calls to core services and enterprise resources; and (iii) presenting the relevant components of the enterprise to the user. The interactions between the user agent and elements of the enterprise include (i) communication with other agents; (ii) direct queries and calls to the enterprise resources; and (iii) usage of PRE core services.

3.3 Resource agent

Resource agents establish relationships between the enterprise resources. For example, one such resource agent will be a chemical modeling agent; this agent collects knowledge about all chemical modeling tools in the enterprise. If a user agent queries a resource agent for this knowledge, this can

be presented to the end-user, and the user can then use appropriate tools for his/her problem. As resources are added to the enterprise and are registered with the trading service, resource agents learn about these new resources. Resource agents can query resources to learn about their capabilities. Since relationships between enterprise resources are established by agents, some traditional complexities associated with capturing knowledge about the resources in a dynamic enterprise are eliminated. In this first phase of our development, the resource agents deal with other resource agents, other agents, electronic resources, and PRE core services. The interactions between these three entities collectively address the following issues: (i) reasoning about the inter-relationships of resource; (ii) communicating with other agents to exchanging information; (iii) using core PRE services; and (iv) propagating relevant information about resources to other agents.

3.4 Information agent

While resource agents perform the task of resource discovery (finding the proper source to query or search), information agents extend this by reasoning about specific data contained in the enterprise resources and core services. Information agents can perform the tasks of database querying, information retrieval, and information fusion (merging results in a meaningful manner). An information agent can build up a knowledge base of how the data contained in enterprise resources and core services are related. For instance, if a collection of several databases (i.e. DB_1, DB_2, DB_3...) are connected to the PRE enterprise, the resources agents will provide the enterprise with possible relationships between DB_1 and DB_3 based on the capabilities of these databases, for example, what type of data is contained in the database? However, the information agents will provide knowledge about the informational aspects of the databases DB_1 and DB_3 such as (i) are there identical query structures between the two databases?; (ii) are there similar records?; and (iii) what is the relationship(s) between their fields? To address these types of queries the information agent will reason about data contained within resources; refer to its knowledge base; communicate with other agents; use PRE core services; and update enterprise information as appropriate.

3.5 Broker agent

The broker agent is a type of "middleware agent" designed to improve the robustness of the existing enterprise core services. We do not use the term broker agent as a middle agent which matches requesting agents and providing agents as in [DeSW97] or [BaBB97]. Our ANS serves this role. Instead, a broker agent in our architecture will gather resource and data capabilities across multiple enterprise core services. The broker agents reason about the extended enterprise capabilities and exchange this information with other agents. By developing broker agents, we can federate enterprise systems in PRE without modifying the original core services. With extended enterprise information, agents can solve a wider range of problems for a user. The broker agents currently residing in the agent layer are designed to support the PRE conversion brokers. To achieve this behavior the broker agents communicate with other agents and exchange information; queries similar PRE core services; reasons about information provided by core services; generates a dynamic views of the enterprise; and alleviates inconsistent information within agent layer.

4 AGENT MODEL

All agents residing in the agent layer consist of the following three general components: *communication interface*, *problem solver*, and *agent knowledge base*. Each of these general

components decompose into several smaller modules, based on the functionality of the individual agent. The diagram in Figure 3 illustrates the components and the flow of information/actions within the agent model. Each agent exhibits the basic characteristics of autonomy, responsiveness, and sociability, as defined in [JeWo98] and [JeSW98]. Each agent is *autonomous*, with control over its own actions and internal state. An agent is *responsive*, perceiving its environment and reacting to changes in the environment; further, agents are *proactive* in the sense that they exhibit goal-directed behavior. Furthermore, each agent is part of a *social community* of agents, interacting with other agents in order to complete its problem solving.

The communications, interactions, and cooperation between agents are achieved through the sociability of the agent model. The communication interface is the central component of sociability where the agent (i) sends/receives messages; (ii) observes activities; and (iii) translates KQML performatives and builds new KQML messages. Once a message is received by an agent, the communication interface parses the message and passes the translated message to the problem solver component. This causes the problem solver to act on the message content.

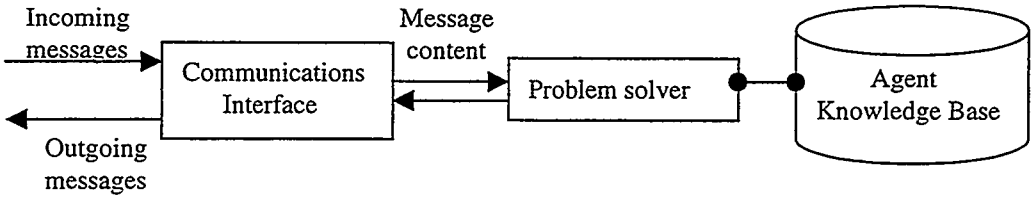


Figure 3. High-level agent model.

The problem solver component permits the agent model to achieve autonomy and responsiveness. The problem solver is responsible for (i) reasoning about how to accomplish the requested; (ii) scheduling activities; (iii) reasoning/accessing/maintaining/creating agent knowledge; (iv) inferring new knowledge from existing information, through deduction; and (v) collaborating/negotiating with others to accomplish problems. We are achieving some of the reasoning aspects of the agents with the use of the expert system shell JESS [Frie97]. By using JESS we are able to capture the varied cognitive reasoning processes needed in the problem solver of each agent to successfully accomplish (i), (iii), and (iv). The agent knowledge base holds information, world models, etc. that assist the agent in meeting its goals; the problem solver component retrieves and updates the knowledge base.

5 EXAMPLES OF ENTERPRISE AGENTS

We envision many new scenarios with the addition of agents into the PRE framework. These scenarios require agents to interoperate with PRE applications, PRE services, and of course, other agents. We detail two scenarios which are currently possible with the first enterprise agents we have developed and deployed.

5.1 Selecting from Similar Resources in the Enterprise

A user wants information about a set of similar modeling codes available in the enterprise. The user wants to view information about the codes (e.g., location of modeling code, hardware platform on

which modeling code executes, specific capabilities of modeling code, quality of modeling code, etc.) prior to selecting a particular modeling tool for use. Figure 4 shows a picture of this scenario where several modeling codes are registered with a PRE trader, and the user wishes to select a single one for his/her use. For simplicity this diagram only shows one resource agent and one trading service, though there could be many of both of these entities.

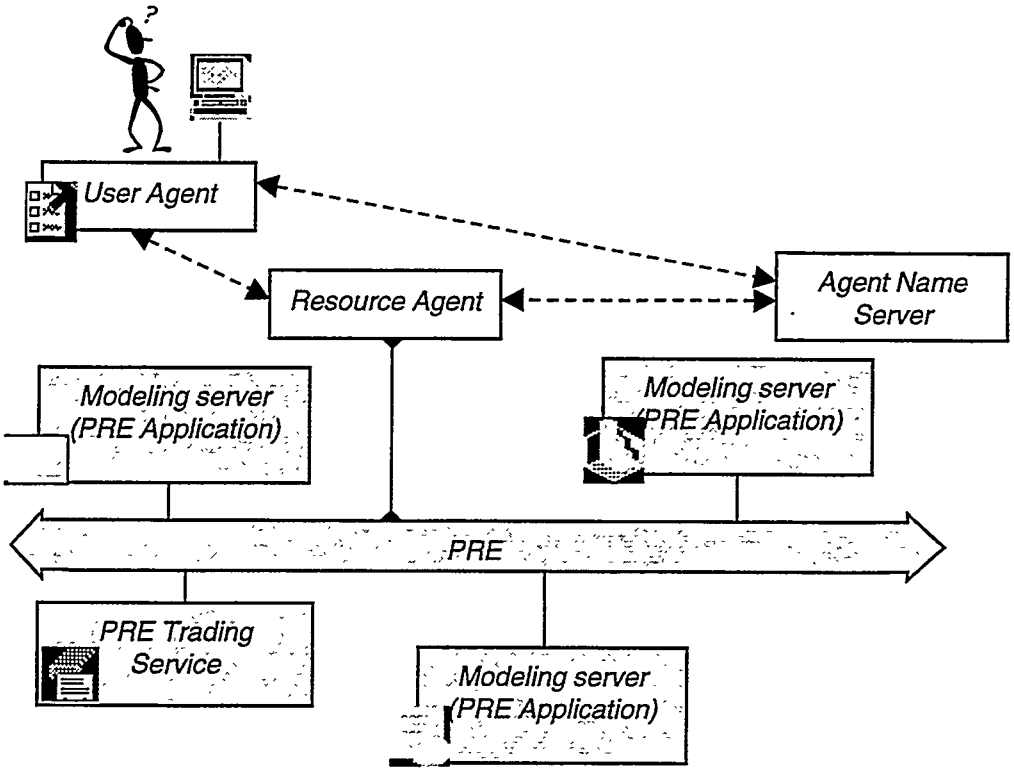


Figure 4. Locating the Desired Enterprise Application.

The sequence of steps which is required to make this happen are the following:

1. The user interacts with his/her personal user agent through a web browser interface and selects a class of modeling codes. The user agent processes this request.
2. The user agent contacts the ANS to locate one or more resource agents that advertise this capability.
3. The ANS directs the query to one or more resource agents.
4. A resource agent processes the request by querying a PRE trader in order to determine the location of the available modeling servers.
5. The trader returns the locations of these modeling codes and some preliminary information about these servers.
6. The resource agent then refers to its knowledge base regarding other data for the desired modeling capabilities. This step may require coordination with other resource agents.
7. The resource agent replies to the user agent.
8. The user agent may repeatedly query the resource agent to collect additional information about these codes/resources.
9. The user agent refers to its knowledge base prior to presenting the information to the user. A sophisticated user agent may be able to prioritize the modeling codes based on past user requirements or some other knowledge about its user.
10. The information requested is presented to the user in an appropriate format so he/she can select a particular code for execution. Again, a sophisticated agent may make recommendations based on past history.

This example has interaction between a human and a user agent, agent-to-agent interaction, and interaction between a resource agent and the enterprise core services. An agent processes queries based on its past knowledge and on new knowledge obtained from the current status of the enterprise. At this time our user agent and resource agents are quite simplified, but we are exploring learning techniques in order to build more sophisticated agents. In particular, we intend to develop advanced user agents, as we believe these agents will greatly enhance the user's productivity as he/she works in the enterprise.

5.2 Agents Reasoning Across a Wider Enterprise

As shown in Figure 5, an extended enterprise can contain several trading services, and several conversion brokers. The addition of broker agents can coordinate multiple trading services. This can have a significant effect on another core service, the conversion broker. Currently within PRE, each conversion broker reasons over a single trader in order to effect multi-step data conversions (i.e., in order to convert from A to C, you can first convert from A to B and then convert from B to C). We propose to add a broker agent to extend the capabilities of a conversion broker across several trading services. With one or more broker agents, the conversion broker capabilities can be extended to reasoning across a larger set of data converters by linking traders dynamically.

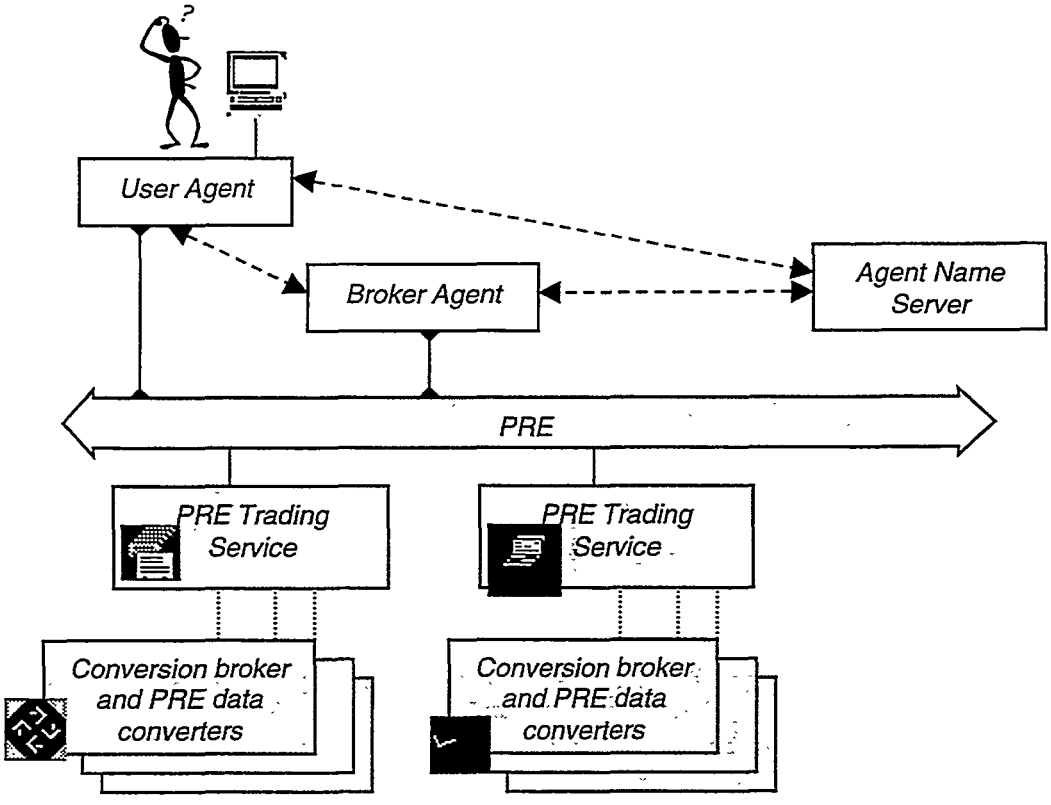


Figure 5. Extending conversion capabilities across an extended enterprise.

We will briefly describe a specific scenario. A user requests that a file of type XX needs to be translated into a file of type YY. This request is directed from the user agent to a PRE conversion broker. If the conversion broker cannot respond with a conversion, the user agent contacts the ANS to find a broker agent. This broker agent can query many individual conversion brokers across the enterprise. If the conversion request cannot be answered by a single conversion broker, the broker agent queries individual traders to determine the convert capabilities of all applications registered with the traders. The broker agent can then build a multi-step conversion across trading services.

A broker agent solves the problem as follows: (1) evaluate its knowledge base; (2) determine viability of information; (3) request updated information about data converters from known PRE traders across the enterprise; (4) reason about the information received on converters to determine solution; (5) if a solution cannot be reached, consult other broker agents with this same capability. Broker agents can coordinate in order to solve multi-step conversions. For example, suppose the broker agent B_1 is unable to solve the entire conversion from XX to YY; however, it has found a conversion from XX to any of the following: VV, WW, or ZZ and from WX and XY to YY. B_1 must now seek help from other broker agents to provide a complete solution, given the partial solutions that are possible. B_1 asks the ANS to send a message to any other broker agents with the capability of locating data translators on other PRE networks. The ANS locates another broker agent B_2 and directs the message from B_1. B_2 takes this message and goes through a similar process as B_1 to solve the problem. B_2 determines it does know about a translator that can solve the problem, because it can convert from data format WW to YY. B_2 sends a message back to B_1 with this part of the solution path. B_1 receives this message and informs the user agent of the required translation path.

In time, the conversion brokers will be modified to collaborate with a collection of broker agents to determine how file translations across traders can be achieved. At this time, however, the underlying enterprise framework is not being modified, but rather enhanced by the agents.

6 AGENT ARCHITECTURE IMPLEMENTATION

Similar to other agent architectures such as JATLite and InfoSleuth, we have created an agent template (or agent shell) to facilitate the creation of agents in our enterprise. To create a new agent, we simply subclass the agent template, inheriting functionality from the template, and implement agent-specific code into the detailed agent layer. We are employing a layered approach with well-defined interfaces between layers. This allows us to modify layers easily. We briefly discuss each layer here.

Transport Layer

The underlying PRE infrastructure will be used as a transport device for the sending/receiving of agent messages. This framework does not alter the syntax or semantics of KQML, in essence, it just allows CORBA (and TCP/IP) to be the transport method for KQML messages, separating the agent communication layer from the transport layer.

Message Layer

The message layer is the layer at which KQML is implemented. At this layer, the syntax of KQML is checked to ensure that all messages sent and received are valid KQML messages.

Protocol Layer

The protocol layer is the layer at which agent conversations and negotiations take place. At this layer, an agent determines which KQML performative to use when sending a message or replying to a request. During the first phase of our development, agents engage in very simple conversations.

Agent Layer

This layer defines the common modules for the agent model discussed in Section 4. In particular, each agent has a problem solving unit and a knowledge base. Furthermore, this is the level at which all agents are aware of the underlying PRE infrastructure, specifically how to contact PRE core services or PRE applications.

Detailed Agent Layer

The detailed agent layer is unique for each agent. This is the application-specific functionality that is required by the agent.

7 INITIAL EXPERIENCES

The first phase of our design and implementation has explored the incorporation of agents to provide an architecture where knowledge is managed and created at an enterprise level. The main strength of our approach is the ability to capture, create, and use the traditionally unused/inaccessible knowledge in the dynamic enterprise. This process is achieved through the collective efforts of resource, information, and broker agents to reason and integrate knowledge into enterprise solutions. The user agent allows a user and his/her preferences to be integrated into a working enterprise.

The ultimate target of our architecture is the end-user. Users have more comprehensive knowledge about enterprise resources and applications, can access a dynamically changing enterprise, can intelligently locate enterprise applications and services to fit their needs, and have improved search capabilities for applications and data. Even the simple user agents that we have implemented in this first phase of our project are beneficial to users who are using enterprise resources and data. Without a user agent which interfaced to a resource agent, there was no way to easily navigate and search through a class of electronic resources. We are accomplishing our goal of enhancing a user's experience in the enterprise, and as agents become more complex and evolve, we expect higher payoffs.

Our agent architecture is supported by a strong infrastructure based on middleware and distributed object technology. This alleviates many of the complex issues associated with distributed systems and gives us a reliable infrastructure on which to develop agents. Given the enterprise framework and the agent template we are developing on top of it, we have found that developing enterprise agents is much easier than if we were developing them from scratch. Furthermore, we believe that our agent architecture is robust, adaptable, and enduring because of both the strong enterprise infrastructure and the template approach to developing agents.

8 CONCLUSIONS AND FUTURE WORK

We have presented our agent-based enterprise, which is built upon Sandia's PRE enterprise framework. It is unique in that we have enhanced an existing enterprise architecture by building agents that add capabilities beyond what the enterprise architecture provides. In this enterprise, agents communicate with each other in an agent communication language, access enterprise resources, and utilize core services of the enterprise framework. Our design decisions are motivated

by the practice of good software engineering techniques, and this is critical to the development of future enterprise systems.

In future phases of our agent architecture we will concentrate on more mature agent-to-agent and human-to-agent relationships, advanced conversations between agents, improved agent intelligence, true proactive behavior, development of the ontologies necessary for agent messages, and the addition of new types of agents. We are beginning to investigate the social aspects of the agent-to-agent relationship by introducing coordination and collaboration protocols for conflict resolution, prioritization, and negotiation. As we do this, the agent conversations will get more complex. We are researching advanced agent conversations in other multi-agent systems and will apply appropriate techniques in our system. The improved human-to-agent relationship hinges on the incorporation of agent learning, information presentation, and the further investigation of FIPA standards for human/agent interactions [FIP898] and related work. To develop the analytical ability of agents, we will employ deductive reasoning techniques. As the architecture and its complexity increase, we envision the addition of agents to translate ontologies, mediate, and plan.

In general, enterprise systems and extended enterprises are complex. The agent society that we are building on top of an existing enterprise has the potential to be complex as well. Therefore, we are careful and deliberate in our design and development and are not "throwing together" a multi-agent system. Our approach is echoed by Wooldridge and Jennings and their plea to avoid pitfall 7.5 *your system is anarchic* [WoJe98].

We believe that software agents will be an important part of enterprise integration architectures in the future. As agents become more common in enterprises, we expect that approaches such as ours will be used.

9 REFERENCES

[BaFo94] M. Barbuceanu and M. S. Fox, "The Information Agent: An Infrastructure Agent Supporting Collaborative Enterprise Architectures", *Proceedings of the Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '94)*, Morgantown, West Virginia, April 1994.

[BaBB97] R. Bayardo, W. Bohrer, R. Brice, *et. al.*, "InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments", *Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD)*, Tucson, Arizona, May 1997.

[BrDB97] J. M. Bradshaw, S. Dutfield, P. Benoit, and J. D. Woolley, "KAoS: Toward An Industrial-Strength Open Agent Architecture", in *Software Agents*, J. M. Bradshaw, editor, The MIT Press, Cambridge, Massachusetts, 1997.

[CORB95] The Common Object Request Broker: Architecture and Specification, OMG Technical Document PTC/96-03-04, Object Management Group, Framingham, Massachusetts, July 1995.

[CuEF93] M. Cutkosky, R. Engelmore, R. Fikes, T. Gruber, M. Genesereth, W. Mark, J. Tenenbaum, and J. Weber, "PACT: An experiment in integrating concurrent engineering systems", *IEEE Computer*, January 1993.

[DeSW97] K. Decker, K.Sycara, and M.Williamson, "Middle-Agents for the Internet", *Proceedings of International Joint Conferences on Artificial Intelligence 97*, Nagoya, Japan, January 1997.

[FIP197] Foundation for Intelligent Physical Agents (FIPA), FIPA 97 Specification, Part 1, Agent Management; `http://drogo.cselt.stet.it/fipa/spec/fipa97/fipa97.htm`, October 1997.

[FIP397] Foundation for Intelligent Physical Agents (FIPA), FIPA 97 Specification, Part 3, Agent Software Integration; `http://drogo.cselt.stet.it/fipa/spec/fipa97/fipa97.htm`, October 1997.

[FIP898] Foundation for Intelligent Physical Agents (FIPA), FIPA 98 Draft Specification, Part 8, Human/Agent Interaction; `http://drogo.cselt.stet.it/fipa/spec/fipa98/fipa98.htm`, July 1998.

[FiWe94] T. Finin, J. Weber, *et. al.*, "Specification of the KQML Agent-Communication Language", The DARPA Knowledge Sharing Initiative External Interfaces Working Group, February 9, 1994.

[FrCu96] H. R. Frost and M. R. Cutkosky, "Design for Manufacturability Via Agent Interaction", *Proceedings of the 1996 ASME Design for Manufacturing Conference*, Irvine, California, August 1996.

[Frie97] E. J. Friedman-Hill, "JESS, The Java Expert System Shell", Sandia National Laboratories, Sandia Report SAND98-8206, November 1997.

[GeKe94] M. R. Genesereth and S. P. Ketchel, "Software Agents", *Communications of the ACM*, Volume 37, Number 7, July 1994.

[JATL98] Java Agent Template, `http://java.stanford.edu`.

[JeSW98] N. R. Jennings, K. Sycara, and M. Wooldridge, "A Roadmap of Agent Research and Development", *Autonomous Agents and Multi-Agent Systems*, Vol. 1, No. 1, p. 7-38, 1998.

[JeWo98] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents", in *Agent Technology*: *Foundations, Applications, and Markets*, N. R. Jennings and M. J. Wooldridge, editors, Springer-Verlag, Berlin, Germany, 1998.

[KnAm97] C. A. Knoblock and J. L. Ambite, "Agents for Information Gathering", in *Software Agents*, J. M. Bradshaw, editor, The MIT Press, Cambridge, Massachusetts, 1997.

[McKW93] J. G. McGuire, D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, and G. R. Olsen, "SHADE: Technology for Knowledge-Based Collaborative Engineering", *Journal of Concurrent Engineering: Applications and Research (CERA),* 1(2), September 1993.

[OlCT94] G. R. Olsen, M. Cutkosky, J. M. Tenenbaum, and T. R. Gruber, "Collaborative Engineering based on Knowledge Sharing Agreements", *Proceedings of the 1994 ASME Database Symposium*, September 11-14, 1994, Minneapolis, MN.

[PaTe92] J. Y. Pan and J. M. Tenenbaum, "An Intelligent Agent Framework for Enterprise Integration", in *Artificial Intelligence Applications in Manufacturing*, A. Famili, D.S. Nau, and S. H. Kim, editors, AAAI Press/The MIT Press, Menlo Park, CA, 1992.

[PaHF95] C. M. Pancerella, A. J. Hazelton, and H. R. Frost, "An autonomous agent for on-machine acceptance of machined components", *Proceedings of Modeling, Simulation, and Control Technologies for Manufacturing*, SPIE's International Symposium on Intelligent Systems and Advanced Manufacturing, Philadelphia, Pennsylvania, October 1995.

[Panc98] C. M. Pancerella, "The Use of Agents and Objects to Integrate Virtual Enterprises", Sandia National Laboratories, Sandia Report SAND98-8226, January 1998.

[Paru97] H. V. D. Parunak, "Technologies for Virtual Enterprises", to appear in *Agility Journal*, 1997.

[RETS98] Reusable Environment for Task Structured Intelligent Network Agents, `http://www.cs.cmu.edu/~softagents/retsina/retsina.html`.

[WePr98] D. Wenger and A. R. Probst, "Adding Value with Intelligent Agents in Financial Services", in *Agent Technology: Foundations, Applications, and Markets*, N. R. Jennings and M. J. Wooldridge, editors, Springer-Verlag, Berlin, Germany, 1998.

[WhFD98] R. A. Whiteside, E. J. Friedman-Hill, and R. J. Detry, "PRE: A framework for enterprise integration", *Proceedings of Design of Information Infrastructure Systems for Manufacturing 1998 (DIISM '98),* Fort Worth, Texas, May 1998.

[WoJe98] M. Wooldridge, N. R. Jennings, "Pitfalls of Agent-Oriented Development", *Proceedings of the Second International Conference on Autonomous Agents (Agents '98),* Minneapolis, Minnesota, May 1998.

DISTRIBUTION:

1 Mr. Chris Murray, Managing Director
Streamline Consulting Pty Limited
Level 30, Optus Centre
P.O. Box 1663
North Sydney NSW 2059 Australia

1 Mr. Mark Volpato, Consulting Director
Streamline Consulting Pty Limited
Level 30, Optus Centre
P.O. Box 1663
North Sydney NSW 2059 Australia

1	MS 9001	T. O. Hunter, 8000 Attn: J. B. Wright, 2200 D. L. Crawford, 5200 M. E. John, 8100 L. A. West, 8200 W. J. McLean, 8300 P. N. Smith, 8500 P. E. Brewer, 8600 T. M. Dyer, 8700 L. A. Hiles, 8800
1	MS 9003	D. R. Henson, 8909
1	MS 9007	R. C. Wayne, 8900
1	MS 9011	P. W. Dean, 8903
1	MS 9011	P. R. Bryson, 8910/8980
5	MS 9011	N. M. Berry, 8920
1	MS 9011	J. Meza, 8950
1	MS 9011	M. Rogers, 8960
1	MS 9011	J. A. Larson, 8970
1	MS 9012	J. E. Costa, 8920
5	MS 9012	C. M. Pancerella, 8920
1	MS 9012	S. C. Gray, 8930
1	MS 9019	B. A. Maxwell, 8940
3	MS 9018	Central Technical Files, 8940-2
1	MS 0899	Technical Library, 4916
1	MS 9021	Technical Communications Department, 8815/ Technical Library, MS 0899, 4916
1	MS 9021	Technical Communications Department, 8815 For DOE/OSTI