Submitted Aug 18, 2003 *Informatica Jr.* (ISSN 0350-5596) published by Slovene Society Informatika (http://ai.ijs.si/informatica/)

Multi-Agent System Case Studies in Command and Control, Information Fusion and Data Management [1]

Frederick Sheldon[1], Thomas Potok[1] and Krishna Kavi[2]

[1]Oak Ridge National Laboratory
Computational Sciences and Engineering
Oak Ridge, Tennessee 37831-6363
Phone: 865-576-1339/ Fax: 865-241-6275
SheldonFT | PotokTE@ornl.gov

[2]Department of Computer Science
The University of North Texas P.O. Box 311366
Denton, Texas 76203
Phone: 940-565-2767 / Fax; 940-565-2799
Kavi@cs.unt.edu

Abstract

On the basis of three different agent-based development projects, we assess the fitness of software (SW) agent-based systems (ABS) in various application settings: (1) distributed command and control (DCC) in fault-tolerant, safety-critical responsive decision networks, (2) agents discovering knowledge an open and changing environment, and (3) light weight distributed data management (DM) for analyzing massive scientific data sets. We characterize the fundamental commonalities and benefits of ABSs in light of our experiences in deploying the different applications.
Key Words: Intelligent software agents, ontology, information fusion, collaborative decision support.

1 Introduction

Systems whose information-processing structures are fully programmed are difficult to design/evolve for all but the simplest kinds of applications. Changing and dynamic open environments will characterize future real-world software application context. Such systems must be able to modify their behavior by changing their information-procession structures [1]. Software agents (SAs) are the latest advancement in the trend toward small modular pieces of code where each module performs a well-defined, focused task or set of tasks. Programmed to interact with and provide services to other agents, including humans, SAs autonomously with prescribed backgrounds, beliefs and operations. Systems of agents can access and manipulate heterogeneous data such as information available on the Internet [2]. Not all agent systems have to have the above properties but any agent-based paradigm must have the ability to engender agents with some or all of the aforementioned properties.

1.1 Agent Technology, Maturation & Evolution

SW development methods have been transformed over the years from structured analysis methods, where processing and data were kept separate [3], to Object-oriented (OO) methods, where processing and data are combined into SW entities called objects [4, 5] (¶1.6-1.7). Object technology was further enhanced with distributed capabilities, allowing an object on one system to communicate with objects on other systems [6]. Objects may be transmitted across a trusted network and executed on another computer, commonly known as mobile code [7].

Furthermore, component-based software development (CBSD) can be viewed as a similar evolutionary trend, which differs from traditional software development. For example, CBSD includes activities selection and creation of SW architectures, as well as the customization of components, while implementation deals with component integration. Typically, this process involves developing wrappers that bond reusable components into a cohesive system rather than extensive coding "from scratch" construction. Indeed, developers must architect/design extensibility into a system and all of its parts to make components independently producible and deployable. SAs offer a great deal of flexibility and adaptability within this context. Agent-oriented SE provides developer's high-level flexible abstractions from which to represent and conceptualize distributed application systems (e.g., delegation of information search, analysis, negotiation and presentation).

SA systems, to some degree, are characterized by being persistent, mobile, knowledgeable, adaptable, autonomous and collaborative, which facilitates the building and evolving of software systems as technologies and requirements change [8]. Developers use increasingly pervasive message-based middleware and component technologies, Java, Extensible Markup Language, and the Hypertext Transfer Protocol to create agent-based software systems. Mobile appliance-oriented application servers and portal technologies based on these technologies provide a basis for more robust agent-oriented systems. These technologies will make the use of mobile appliances, adaptive content, and SAs quicker and easier.

1.2 Distributed Computing

Distributed or ubiquitous computing envisions devices ranging from super computers to nanoscale CPUs acting in

*This manuscript has been authored by UT-Battelle, a contractor of the U.S. Government (USG) under Department of Energy (DOE) Contract DE-AC05-00OR22725. The USG retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

concert to solve problems. Current distributed computing approaches include the Common Object Request Broker Architecture (CORBA), the Distributed Component Object Model (DCOM), and Remote Method Invocation (RMI) Each provides a way of executing a SW function needed by one computer on a different computer. Remote execution places a number of constraints on the SW. For example, assume that a source object (e.g., program or function) is attempting to execute some function on a target object; the source object must have the capability to resolve the network and computer memory address of the target object. Next, the source object must have detailed prior knowledge of the functions (methods) and parameters available on the target object, as well as return information. There are also assumptions that these remote functions will be accessed synchronously and that the network connections are available and permanent. If any of these assumptions does not hold, then these distributed interactions will fail [9].

1.3 Agent infrastructure

The dynamic interaction of multiple SAs requires an architecture that supports "our definition" of an agent (i.e., is a program P, written in a language l, P_l an agent?), what underlying infrastructure is needed to support agents to interact effectively, and how the agents will utilize the infrastructure to interact. The Oak Ridge Mobile Agent Community (ORMAC) is a communication/mobility framework developed over the course of several agent-based research projects. ORMAC is generic framework providing transparent agent communication and mobility across Internet connected hosts (Fig. 1). This architecture enables an agent community to be quickly created using a set of machines with each machine executing the ORMAC agent host software (SW): (1) SAs migrate among machines as necessary to facilitate communication among agents within the community, and (2) ORMAC SAs can also interact with systems and agents that are not part of the community. Internet mobility is very limited based on enforced Internet security/firewall constraints. ORMAC uses the Foundation for Intelligent Physical Agent (FIPA) compliant agent communication language (ACL) messages. Any FIPA compliant agent can interact with an ORMAC agent [10, 11]. Within an ORMAC community, each agent host has a name server responsible for tracking where agents are currently hosted. In addition, the name server is responsible for answering queries from agents trying to locate other agents in the community. For example, an agent may want to broadcast information to all agents within the community. The name server for each agent host is used to locate all such agents for delivery of said message(s).

Agents migrate among machines by changing *agent hosts*. When an agent is received at an agent host, the agent host provides it with an agent context. This agent context is the agent's only point of contact with the machine it is running on and provides machine specific environments for the agent to work. The agent is not allowed to directly communicate with the agent host or other agents. This provides an architectural layer for security in the ORMAC system (written in JAVA, ORMAC uses Remote Method Invocation (RMI) to communicate among agents).

1.4 Heterogeneous agent interoperability

Ontology-based thesauri have been an important part of research in Natural Language Processing. As the need for distributed software configurations has risen, Ontologies have become increasingly important. Ontologies have evolved as a convenient way to permit agents using diverse vocabularies to specify common concepts. There are two main approaches (1) creating a large general ontology, or (2) many domain-specific ontologies. Most ontologies or thesauri are constructed manually, however, methods have been developed for automated construction of such [2]. In our Virtual Information Processing Agent Research (VIPAR) case study, agents use a flexible RDF (Resource Description Framework) ontology to transform heterogeneous HTML documents to XML tagged documents, and their ability to rapidly cluster newspaper articles that arrive in an asynchronous manner.

Agents move from one machine to another by changing *agent hosts*. The *ontologies* move with the agents. When an agent is received at an agent host, the agent host provides it with an agent context. This agent context is the agent's only point of contact with the machine it is running on and provides machine specific environments for the agent to work. The agent is not allowed to directly communicate with the agent host or other agents. This provides an architectural layer for security in the ORMAC system (written in JAVA, ORMAC uses Remote Method Invocation (RMI) to communicate among agents).

1.5 Independent asynchronous communication

Agent-based architectures provide several advantages over OO technologies where objects communicate through messages. The sender object must know the address of the

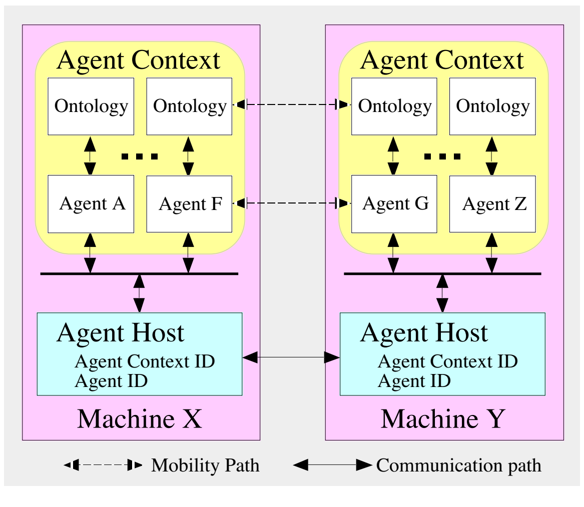


Fig. 1 ORMAC mobility-communication architecture.

receiver object (i.e., public methods). In contrast, the ORMAC framework imposes a communication protocol that allows messages to be sent without having to know the specific address/method(s) of the recipient [12]. This allows agents to migrate among host and still be in connected with other agents via direct or broadcast requests to any number of other agents. ORMAC provides the ability to use an ontology to direct agents through a task. An ontology can act as a script, or rule base for an agent to follow. This difference is perhaps more conceptual than practical because there currently is very little ontology standardization. For example, in our information fusion case study, an RDF ontology is used to describe the characteristics of each Internet newspaper within the system while agents use the ontology to correctly interpret and retrieve the appropriate information (see ¶2.2).

Furthermore, agents can suspend processing on one machine, move to another, and resume processing. In this way, the possibility exists to prioritize agents (tasks) by sending high priority agents to faster resources, and for example, load balance a system depending on the workload of each agent. The priority and/or allocation of agents can be determined cooperatively thereby preempting the need for global scheduling (to avert single-point-failure risk).

1.6 Agents cognitive development framework

Kavi, et. al. [13, 14] present a framework for modeling, analysis and construction of agent-based systems. The framework is rooted in the Belief Desire Intention (BDI) formalism and extends the Unified Modeling Language (UML) to model MAS. Several modeling constructs are introduced including *Agent, Belief, Goal, Plan, FIPA Performative, KQML-Performative*, and *Blackboard*. In addition, the following diagrammatic constructs are introduced: Agent Goal Diagram to model the relationships between the goals and the environment of an agent; Use Case Goal Diagram to model the relationships between use cases and goals; Agent Domain Model to facilitate understanding of domain knowledge of an agent; Agent Sequence Diagram to model interactions within an agent. Similarly, Agent Activity Diagram and Agent Statechart Diagram are introduced. The framework is illustrated by an agent-based intelligent elevator system.

The framework is based on extensions to UML to support multi-agent systems (MAS) development. Their approach is rooted in the BDI formalism [15], but stresses practical software design methods instead of reasoning theories. In particular, we propose to extend UML with modeling constructs called *Agent, Belief, Goal, Plan, FIPA Performative, KQML Performative*, and *Blackboard. Agent* is the super-type for all agent types. Belief, Goal and Plan model the reactive and proactive behaviors of agents. An agent has, among other data types, a collection of beliefs, goals and plans. *Belief*s are the agent's observations and/or sensing of the environment and are updated by sensors or other agents. Changes in an agent's beliefs trigger the re-evaluation of the utility values of goals of the agent. Changes to goals' utility values result in pre-empting some plans and initiating new plans. Execution of plans affects the environment, which in turn changes the beliefs, and so on. Agents communicate with each other through agent communication performatives such as FIPA or KQML, or shared blackboards as in Linda or its extensions. In the conceptual model of our framework the Agent Goal Diagram (AGD) is introduced to model the relationships between the goals and the environment, the Use Case Goal Diagram (UCGD) to relate use cases and goals, Agent Domain Model (ADM) to facilitate understanding of agent domain knowledge, Agent Sequence Diagram (ASD) to model interactions within an agent. Similarly, Agent Activity Diagram and Agent Statechart Diagram are introduced.

1.7 Integration of Mobile Agents/Genetic Algorithms

Papavassiliou et. al. [16], present an agent based approach for building a framework where resource allocation is provided under the control of different and often-competing stakeholders (users, network providers, service providers, etc.). They describe the efficient integration and adoption of mobile agents and genetic algorithms in the implementation of an effective strategy for the development of effective market based routes for brokering purposes (i.e., in the future multi-operator network marketplace). The agent based network management approach represents an underlying framework and structure for the multi-operator network model, and can be used to collect all the required management data. The proposed genetic algorithm provides a kind of stochastic search for optimal resource allocation strategies. [16]

Agent programming was developed in the distributed programming field as a flexible and complementary way of managing resources of a distributed system. Distributing intelligence across the network allows the fast exploitation of advanced services that dynamically adapt to the user's requirements. User requirements are automatically translated into network requirements, and this implicitly assumes the possibility to interact with network equipment. For example, network providers who need application level information to better manage their resources can better satisfy their user needs while minimizing their costs. Moreover, content providers can gain the knowledge of the network resources needed by their services to be properly accessed.

2 Case studies

2.1 Distributed command and control

ABSs are particularly suitable for satisfying both functional and nonfunctional DCC requirements, especially in satisfying application scalability, mobility, and security (SMS) expectations. A general set of DCC SW requirements (SRs) was developed based on needs aligned with current computer science technology and inherent limitations [12]. ABS advantages (i.e., SMS) are enabled mainly through a stronger messaging/coordination (MC) model; however, the impact of key DCC system/functional requirements poses the greatest SW challenge. While information fusion, information summary and analysis, and decision support are only tangential to SW technology advances (see Figure 2). Our analysis indicates six key challenges best-addressed using agent technology to provide:

1. Higher-level interfaces to distributed objects,
2. Asynchronous object interaction,
3. Message support for sporadic network connections,
4. Secure object communication and information system operation,
5. Support for richer peer-to-peer programming models,
6. Accelerated SW development productivity.

ABS is an evolving paradigm that strives to create SW that can mimic certain human behavior. Agents are typically endowed with human-like characteristics. For example, agents are normally considered to be autonomous, adaptable, social, knowledgeable, mobile, and reactive [17]. Lets consider therefore, the comparative benefits of agent technology.

A representative agent architecture by Sycara et al. [18] describes planning, communication and coordination, scheduling, and execution monitoring of agent activities. Agents' access shared information, implemented through a coordination model that can be both domain specific or independent. Griss et al. [19] describes a generalized agent architecture with facilities for locating and communicating with mobile, disconnected agents, and for gathering information about groups of agents. Griss's architecture provides services and support for mobility, security, management, persistence, and the naming of agents.

In general, most agent architectures include support for DCC aspects through a general MC paradigm (i.e., any agent can communicate with one or more agents). This approach encapsulates messages that agents send and receive [17]. OO methods utilize the concept of data encapsulation, which provide for simple SW functions to access an object's data. These functions, not direct data access, are responsible for data retrieval and update. This capability limits the SW (i.e., coupling) that must change due to nonconforming data formats, etc. The agent paradigm extends encapsulation from data to messages sent among agents through an agent coordination model [20]. The model defines how agents communicate among themselves, and can be seen as coordinating communication based on the time a message is sent (temporal) or the names of the target agents (spatial). These models provide the ability for communication that is encapsulated and asynchronous with the use of blackboards, and tuple space models and associated pattern matching, such as Linda [21]. Agents that use a blackboard or Linda type coordination provide a level of indirection for agent communication (i.e., agents post messages to a blackboard, while subscribers to the blackboard retrieve the message). The agent that sent the message may have no idea who actually receives it. This concept allows for asynchronous and encapsulated communication among a collection of connected or disconnected agents, a capability not currently available in non-agent systems.
Messages are written in an agent control language [22] (ACL) such as KQML or the FIPA ACL, which provide a structured means of exchanging information and knowledge among agents. ACLs support a higher-level communication protocol that does not currently exist for distributed objects. On this basis, lets consider how the DCC concept challenges ABS SW development.

SW Requirements / **SW Technology Limitations**	Distributed cmptng	Fault Tolerance	Mobile Code	Security	Information Fusion	Information Analysis Summary	Decision Support	SW Productivity
Higher-level Interfaces	**X**			**X**				
Asynchronous Interaction	**X**							
Sporadic Network Support	**X**	**X**	**X**					
Security			**X**	**X**				
Peer-to-peer Models	**X**	**X**						
SW Productivity								**X**

Figure 2. A mapping of the SW requirements to the limitations of the current SW technology

2.1.1 Higher level interfaces to distributed objects

Agent technology, based on a flexible MC scheme and control language, (conceptually) require agents to be connected to blackboards, not other agents[17]. The encapsulation of messages allows for agent interfaces to change, requiring only minor modifications to a blackboard, not to all calling agents. This capability provides for a more robust interface than is currently available in distributed object systems. Moreover, ACLs provide the ability to pass propositions, rules, actions, and states among agents. In this way, messaging is not merely a way of activating a function on a remote host, but provides a way of sending information to another agent. This information can be used to describe what requirements need to be met for an agent to take action, what states the sender and receiver will be in after the action takes place, or what states the agents will be in when the overall transaction is complete [22]. Information sent from one agent to another may also be informative or declarative thereby causing no agent action.

The challenge of implementing such an agent interface is selecting both an MC architecture and an ACL. Currently, no universally accepted MC architecture or ACL means that for an ABS to take advantage of this high-level interface, there must be very specific and precise specifications on how agents will communicate (i.e., using precise ACL syntax).

2.1.2 *Asynchronous object interaction*

Griss et al. [19] points out that ABS typically have simple interfaces, and derive capability from loose coupling and asynchronous messaging (i.e., messages are sent and retrieved through a loosely coupled temporal agent coordination model). Cabri et al. [20] reference two coordination models that provide asynchronicity. The first coordination model is blackboard-based and provides a shared area where agents' send/retrieve messages. Any authorized agent can read messages posted to the blackboard. Other agents determine whether to retrieve the message based on the sending agent's identifier and therefore knowledge of the agent identifiers is required. The second is based on the Linda coordination model, which defines a messaging protocol, made up of a tuple of information (e.g., a tuple may include the data format, the date of creation, the classification, or a list of keywords). These tuples are placed in a shared area, such as a blackboard. Agents access these messages, not based on agent identifiers, but on a query of the tuple information, (i.e., an agent may retrieve all messages created yesterday with the "Taliban" keyword). This model is asynchronous, and does not require knowledge of the agent identifier.

Both model types are mature and widely used. They provide needed asynchronous behavior but suffer from single-point failure outages. Thus, a single blackboard ABS is exposed to security and performance failures and requires multiple blackboards to provide fault tolerance.

2.1.3 *Message support for sporadic networks*

One main advantage that ABS provide is flexibility (i.e., ability for agents to change location) along with communication path redundancy. Vogler et al. [23] propose a distributed transaction model using a two-phase commit protocol to verify message delivery. The model must support storage of undelivered messages within the agent, or support the ability to rollback the transaction, if synchronous transactions are required. If a transaction has not completed, then various network/graph theory algorithms can be used to determine a viable path prior to reattempting the transaction. Alternatively, agents can move to another location and try again. If a physical path cannot be found then the transaction is not possible.

Both messaging and mobility can be effectively used to communicate over a sporadic network; however, if the network degrades too much, communication becomes infeasible. Distributed transaction protocols are very useful for verifying the success of transactions, and can be used to ensure network security with the caveat that this capability will limit overall system response time.

2.1.4 *Secure communication operations*

As Abadi [24] notes, it is practically impossible to construct a truly secure information system. Communications are secure if transmitted messages can be neither affected nor understood by an adversary; likewise, information operations are secure if information cannot be damaged, destroyed, or acquired by an adversary.

Security in a distributed system can be enforced through system wide policies, which are often static, and difficult to modify and enforce [25] ABS can enforce a security policy defining what must be done and what must not be done when information is moved, stored, created, or destroyed. ABS provide multiple, standalone, persistent processes that can act at high speeds to ensure that all rules are always followed. Encapsulated instructions concerning what actions to take under what circumstances enables agents to execute very complex operations, enabling participation in complex collaborative security protocols (e.g., key updating/multiparty authorization).

2.1.5 *Peer-to-peer programming models*

Fortunately, through the use of blackboard and Linda type coordination models, the programming model of agents can be very general. Any number of agents can send messages to one or many blackboard(s), and any number of agents can receive messages from one or many blackboard(s). Virtually any topology can be created which allows for very broad scalability of the network. Care must be taken in defining the bandwidth, messaging rates, and processing requirements and will require tuning to enhance fault tolerance and performance.

2.1.6 *Increasing SW development productivity*

There are indications that agent technology may provide some SW development productivity improvement [19]. While there exists no empirical evidence to support this, the theory claims that ABS increase the level of SW reuse. Agents are SW components that have their messaging, functionality, and location encapsulated thus increasing productivity. Likewise, if standard MC protocols and ACLs can be defined, the agent development teams may require less communication overhead because the interfaces are far richer than with traditional programming.

2.2 *Information fusion*

In theory, an information SA scours multiple heterogeneous information sources to proactively acquire, semantically understand, process and distribute information and perform other information processing related tasks at the behest and bidding of a specified user. This technology focuses on obtaining a battery of semantic insights from the information-glut/overload that we now face and delivering this semantically digested information in an easy to use/navigate interface.

One so-called Digital Assistant (DA) ABS offers a variety of information gathering/management and processing features where you can: (1) set up a personal watch-list for companies, news and keywords; (2) monitor various online newsgroups and topics of interest; (3) monitor what companies and topics are favored by media; (4) track regular financial data to get a statistical sense of bullish/bearish Sentiment in the market. Results are made available in a decision-ready format (tabular and statistically aggregated percentages) with the flexibility of setting up an email alert containing the digest [26, 27].

The most advanced feature of the DA is an attempt to gauge investor sentiment from various online message boards in the form of an *Opinion Rating*. Various public message boards are scoured to understand what investors are saying about the companies and based on a semantic understanding of these messages a quantitative *Public Opinion* index is formed (assessing the opinion-pulse in the stock markets). Future enhancements could include a news opinion engine that will (at the aggregate level) *understand* what people are saying about a company or how the media is profiling a particular company as well as the ability to query the DA through email. Such enhancements could provide insights into when and by how much market psychology, herd mentality and media exposure has an impact on a stock's price.

The VIPAR project/tool employs ABS technology 1) to utilize the ability for broadcast and peer-to-peer communication among agents, 2) to follow rules outlined in an ontology, and 3) provide persistence (because of the ability for agents to suspend processing on one machine, move to another, and resume processing). These strengths are combined for the purpose of providing an Internet-based DA to support aspects of intelligence, surveillance, and reconnaissance (ISR) in multiple languages[28].

2.2.1 Background

Detailed analysis of large collections of heterogeneous unstructured information is an obvious ISR need[2]. The problem can be viewed in two parts, first how to gather and structure information, and second how to organize and classify information.

2.2.2 Approach

Two broad approaches exist to efficiently gathering and structuring frequently changing heterogeneous Internet accessible information. First, we could obviously use Internet search engines (ISE), which (typically) use programs that recursively traverse links, capturing non-trivial terms on each page. Pages are organized based on the *relevance* of encountered terms enabling a wide variety and number of documents to be categorized according to relevance and made available for further refined searches/reorganization.

ISE weaknesses include 1) existing pages in the system are infrequently re-traversed tending to make the information stale, 2) the Internet pages have no consistent format, and therefore, the semantic content of a page cannot be easily discerned, 3) the documents are organized based solely on the presence of a keyword in a document (regardless of other attributes like timeliness).

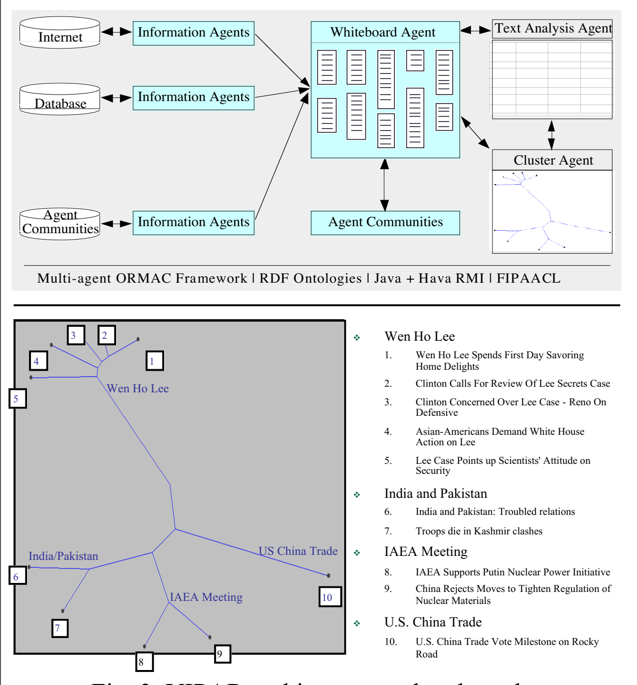
Alternatively, the second approach gathers and structures Internet information using agents. The agents provide various ways to retrieve and organize information, including agents that are capable to access multiple sources, and to filter based on the relevance to the user [18, 29]. Non-cooperating agents perform the information retrieval task, cooperating agents organize the information based on relevance, and finally, adaptive agents deal with uncertain, incomplete, or vague information [30]. Additionally, transforming the inherent and chaotic structure of newspaper articles into a common schema is a difficult problem that must be overcome.

2.2.2.1 VIPAR: unique approach

The VIPAR server uses a set of information retrieval agents to gather news related, non-redundant heterogeneous information from the Internet newspapers, and to format the information using XML (Fig. 3). A whiteboard agent acts as an information-clearing house. Agents submit their articles to the whiteboard agent, who preempts/deletes duplicate articles, archives stale articles (beyond a prescribed age), and feeds articles to agents that have "subscribed" to the whiteboard. A team of cluster agents organizes articles into a vector space model (VSM), then into clusters of articles.

2.2.2.2 VIPAR: information agents

These agents gather and organize information through the transformation of HTML formatted information into XML formatted information. The conversion from HTML to XML is a two-step process. An ontology is defined to provide a common semantic representation and structuring of the heterogeneous information. This ontology embodies the transformation of HTML formatted information to XML formatted information. This ontology is expressed in



Wen Ho Lee
1. Wen Ho Lee Spends First Day Savoring Home Delights
2. Clinton Calls For Review Of Lee Secrets Case
3. Clinton Concerned Over Lee Case - Reno On Defensive
4. Asian-Americans Demand White House Action on Lee
5. Lee Case Points up Scientists' Attitude on Security

India and Pakistan
6. India and Pakistan: Troubled relations
7. Troops die in Kashmir clashes

IAEA Meeting
8. IAEA Supports Putin Nuclear Power Initiative
9. China Rejects Moves to Tighten Regulation of Nuclear Materials

U.S. China Trade
10. U.S. China Trade Vote Milestone on Rocky Road

Fig. 3. VIPAR architecture and end results.

[2] Virtual Information Center (VIC) at US Pacific Command, gathers, analyzes, and summarizes information from Internet-based newspapers on a daily basis (a manual, time and resource intensive process).

an XML variant called the Resource Description Framework (RDF, see http://www.w3.org/RDF/). The RDF syntax allows directed graphs to be expressed in an XML-like format. An Internet site is a collection of linked Internet pages. A site is viewed as a directed graph and RDF provides a way to model the linked pages. Furthermore, our agents understand these RDF instructions. A series of RDF ontologies have been developed for the newspapers accessed by the VIPAR system. Each site ontology describes a newspaper: (1) meta-information about the newspaper, and (2) describes site-specific agent actions (e.g., login, etc.). Based on the ontological description of a newspaper site, the agent monitors and manages the information at the site.

An HTML→XML conversion is completed using the defined ontology. An agent, using the RDF ontology, to understand the site layout/semantics can autonomously retrieve articles of interest, and perform the conversion into a structured XML formatted document. Each converted article contains a rich set of XML tags ranging from the time and date the article was discovered, URL location, to the XML tags that format the article. Each agent monitors the site looking for new articles. Fresh articles are formatted and posted to the whiteboard agent.

The ontological site description (OSD) includes a root URL where the agent begins traversal of the site and from which the agent resolves relative site URLs. The OSD includes a series of regular expressions used to describe the table-of-contents for the site. The site description includes a series of regular expressions that describe article pages of interest along with contextual information (i.e., differentiating the text of an article from the myriad of unimportant information (boilerplate, banners, ads, etc). Meta-information is maintained which includes the newspaper's name and the name of the collection under which VIPAR classifies the newspaper, as well as site-specific actions taken by the agents (e.g., search depth limit [hops from the root URL], minutes to wait between rescanning for new articles, etc.).

Using the RDF ontology agents' monitor/manage each site. They check each link against its ontological criteria to discriminate table-of-contents versus article pages. If an article page of interest is found, the agent requests the whiteboard agent verify that the article is not already posted. If the article is not posted, the agent reads the page, distills out clean article text (i.e., filters the raw text from nonessential/extraneous information). The agent marks up the clean text using XML, tagging the parts of the article (title, author, date, location, paragraphs, etc) depending on the site, and then posts the information to the VIPAR whiteboard agent. The agent continues to monitor the site, posting new information of interest as it becomes available. The VIPAR client is also an ORMAC agent that contains a graphical user interface. The client agent communicates with both the whiteboard and cluster agents to direct/refine searches and clustering.

The whiteboard agent maintains all current articles, ensuring no duplicates, and removing articles beyond a certain age. The cluster agent subscribes to the whiteboard agent and thus is notified when an article is added or removed from the whiteboard. When the cluster agent is notified of a new article (as discussed below), it examines the contents of the article and adjusts its search and clustering tables appropriately. Likewise, the tables are adjusted when the whiteboard removes an article.

2.2.2.3 VIPAR: dynamic article clustering

Two basic steps are taken to organize articles into clusters. The first creates a VSM from the articles. The VSM presumes that newspaper articles and their significant terms (words) can be represented as elements of a multi-dimensional vector space. Within this space, each significant term is represented by a new dimension, and a document is represented as a vector within this multidimensional space [31]. The value of each vector coordinate is an entropy-based function of "local" and "global" frequencies of the word corresponding to this dimension. The cluster agent maintains information containing the frequency of occurrence of terms within a document, called local term frequency, and over the entire set of documents, called global term frequency. These term frequency counts are then used to calculate a weight for each term in each document, which is called the document term weighting.

The second step creates a similarity matrix (SM) that provides a pair wise comparison of each document in the system. We use the dot product (i.e., cosine of the angle between the vector pair) as the measure of similarity between two document vectors. This generates a global SM of size $n \times n$, where n is the number of documents contained in the document collection. Only the upper triangular portion of this matrix is needed because it is a symmetric matrix. Note, when a document is added or removed the VSM *must* be updated. This is due to the changes in the global frequency of words that are contained in this document. The brute-force approach is to re-compute all the document vectors in the document collection (i.e., document term weights of each document vector) as well as a global similarity matrix. However, the time/space complexity is $O(n \cdot d) + O(n^2)$, where d is the document vector space dimensionality. This is very expensive when the collection size grows. An approach is needed to more efficiently update the SMs. A sliding-window-based approach is used.

The whole SM is modeled as a circular array of size $\frac{n(n-1)}{2}$ with a pointer initially pointing to the first array element. When a new document is added or removed from the collection the p percentage of the SM is updated and the pointer is forwarded $p \cdot \frac{n(n-1)}{2}$ steps from its current position, thus pointing to the next stale entry of the array.

A series of experiments was conducted to determine how changes in global term frequency affect the similarity values. Updating 5% of the global SM every time a single document is added or deleted preserves high accuracy. To compare SMs, several measures were made based on the values of their determinants, traces, and x^2 distribution. In other words, it takes 20 document additions or removals to fully update the SM. This method resulted in acceptable dynamic similarity update performance.

Finally, a global SM is used to perform on-demand clustering of the documents of interest (e.g., the documents retrieved in response to a user query). For a set of documents to be clustered, the local SM is constructed by including the cells of the global SM that in turn corresponds to the documents of interest. This local SM is used to analyze the documents of interest based on their closeness in the document vector space. The documents are merged into clusters using an agglomerative hierarchical clustering algorithm [32]. When all of the documents are combined, a Phylips Tree is generated to illustrate the hierarchical tree structure of the clustered documents (see Fig. 3 lower half). The Phylips Tree (or cluster diagram) is a type of dendrogram. The nodes of the tree represent each article while the edges (or links) between nodes represent relationships. In general, the closer (based on distance and hops) two nodes are, the more similar the articles. If links from two nodes share a vertex, then these articles are the closest in the set of articles. The longer links between article nodes indicate greater dissimilarity.

2.2.3 Results

Organization of the acquired information using the VIPAR system was very successful. In an experiment comparing the organization of news articles done manually, versus organized by VIPAR, results favor VIPAR as the preferred method. The experiment involved searching a collection of newspapers for key terms that produced a number of relevant news articles. This collection of articles was then manually organized based on the contents of the articles. Following this manual process, VIPAR was used to organize the same article set and the results from both methods compared. A search was performed on September 21, 2001, using the phrase "nuclear weapons." At the time, five newspapers were in the VIPAR system, (1) Japan Times, (2) Pacific Islands Report, (3) Inside China Today, (4) Russia Today, and (5) Sydney Morning Herald. The results produced 10 articles, with various titles (see Fig 3 lower half).

These results are typical of an average search engine, except that VIPAR targets newspapers only and is timelier because it filters out articles older than a few days. Manually clustering these articles put the same articles into the same category. This articles collection covers four broad areas, 1) the Los Alamos Nuclear Scientist Wen Ho Lee, 2) the India and Pakistan conflict spurring nuclear weapons development, 3) an International Atomic Energy Agency meeting dealing with nuclear material, and 4) U.S. China Trade Policy dealing with nuclear material. To manually organize a small number of articles like these can be done fairly quickly by a knowledgeable person. However, as the number of articles *increases* so does the time required to *manually* organize the articles.

VIPAR clustered articles within a few seconds and produced 4 distinct groupings. Fig. 3 (lower half) shows a comparison of the VIPAR cluster to the manual clustering. The four groups determined by VIPAR match extremely well to the four groups of articles manually organized. VIPAR clusters provide an intuitive (i.e., natural, quick and effective) way to organize and visualize this information.

2.3 Data management

This case study uses SAs to divide and concur massive amounts of distributed data. The SAs, which run on the machines where the data resides, collaborate to produce movies from the requested data, which are sent back to the remote client for display. The quality of the movies can be varied depending on the available network bandwidth.

2.3.1 Background

Simulationists who model physical phenomena commonly deal with massive (terabytes) datasets widely distributed and derived from months of supercomputing. Refining these models and algorithms to maturity requires numerous iterations where the scientist modifies the algorithm, and validates the resulting output. The scientist either examines the candidate dataset in raw form or invests considerable time and effort to analyze the data using highly specialized hardware and software tools.

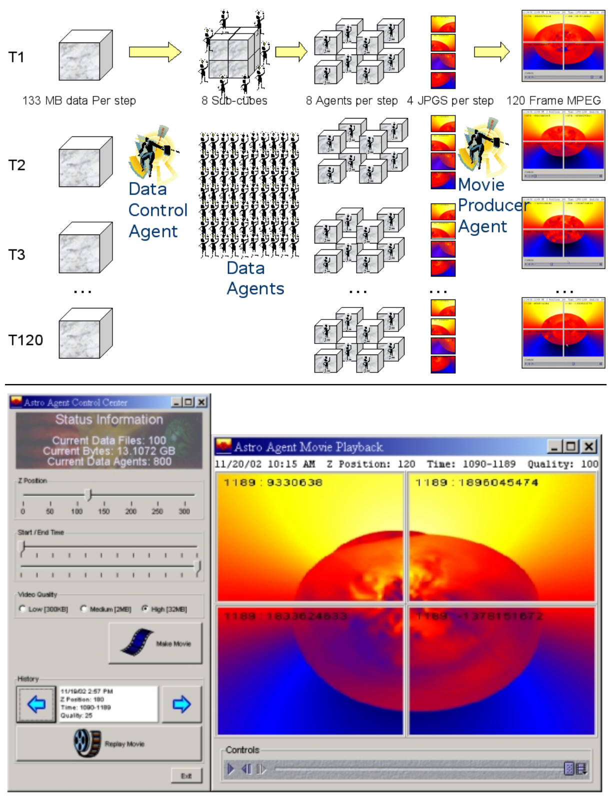
2.3.2 Approach

We proposed a large system of distributed SAs spread over the distributed data as a simple and flexible way to help scientists validate simulation results and refine the simulation model/algorithm. We used 100 time steps of data from a supernova simulation segmented into 800 individual pieces, managed by 800 agents, running on conventional systems. We have developed a system where a single software agent is responsible for each individual segment of data. Upon request, these 800 agents work together to produce a visual representation as shown in Fig. 4. Our results illustrate that a large system of software agents is a simple and flexible solution to the problem of data validation during the development of scientific simulation models. In work with numerous scientists at various laboratories and universities, we have been successfully using this approach to render data from a supernova simulation.

The agent architecture of this experimental system involves multiple software agents, each of which has one of three basic tasks. The first type of agent is called the data controller agent. The data controller agent monitors the simulation output directory for newly created data files. When one is found, this agent then creates and assigns eight new data agents to eight equal sub-cubes of the new file. These data agents are the second type of agent used in the system. The creation of eight agents per new file is arbitrary and easily changed. Each of these new agents is then responsible for fielding requests from other agents. The typical request is for an agent to provide an image from an XY plane of data under its control. In this case, the individual agents will generate an image of a 2D plane from the 3D sub-cube that they are responsible for. If the requested plane falls outside of this cube, the agent ignores the request. The agents also have the ability to vary the quality of the images produced. A blackboard is used to collect images from various responding agents. From this blackboard, the third type of agent, the movie producer agent, assembles the images into a movie that shows an XY plane through the 100 time steps of data. Using different video compressors and decompressors (CODECs) allows the movies to be produced at different detail levels (See Fig. 4 top half).

2.3.3 Results

The dataset was provided by the DOE's Terascale Supernova Initiative (TSI) project. We used a portion of the TSI supernova simulation data to demonstrate that a system made up of a large number of SAs is a viable solution. The original data contains 192 times steps. Each time step containing data from 5 variables, X, Y, and Z velocities, pressure, and density. Data from each variable is represented in a 320 x 320 x 320 matrix of floating point values stored in Hierarchical Data Format (HDF) 4 format in 960 files requiring approximately 128GB of storage. For demonstration purposes, we chose 100 time steps of Y velocity data showing significant activity. This equates to 100 files, each 133 MB stored on two separate PCs.



Current Data Files: 100
Current Bytes: 13.1072 GB
Current Data Agents: 800

Low [300KB] Medium [2MB] [✓ High [32MB]]

11/19/02 2:57 PM
Z Position: 180
Time: 1090-1189
Quality: 25

11/20/02 10:15 AM Z Position: 120 Time: 1090-1189 Quality: 100

1189 : 9330638
1189 : 1896045474
1189 : 1633624633
1189 : -1378151672

Fig. 4. Agent coordination architecture and client GUI.

3 Discussion and conclusions

Lets briefly review our conclusions from the three case studies described here.

3.1 Distributed command and control

A comparison of DCC functional requirements with the capabilities of existing SW technology reveals the limitations of low-level interfaces, synchronous interactions, and requirements for continuous network availability, limited redundancy, and limited productivity improvements. Current technology would require major enhancements (if even feasible) to enable the DCC concept. Moreover, the main strength provided by ABS is derived from the MC model thereby supporting a more flexible and consequently more robust programming model. The intellectual integrity and congruency gained by mapping the DCC requirements onto the ABS model gives a compelling and natural consistency. Furthermore, ABS can support the DCC functional requirements including security, information analysis/summary, and decision support, but the technology does not explicitly provide these capabilities, and these are challenging problems.

3.2 Information fusion

The ISR/VIC problem involves gathering/analyzing more information that can be reasonably accomplished manually (i.e., the common information-glut/overload dilemma that promises to worsen). To address this challenge, the multi-agent VIPAR system was developed using software agents to retrieve, organize, and graphically present Internet-based newspaper information comparable to that accomplished by human intelligence analysts. VIPAR extends the field of agent technology through the use of a flexible RDF ontology for managing information including the capability to dynamically add and cluster new information entering the system.

Agent technology is well suited to this type of problem for three main reasons. First, the communication mechanism allows for broadcast *and* peer-to-peer message passing. Second, using an external ontology allows for an easily maintainable and/or replaceable mechanism for adapting to an open/changing information environment and rules (intelligence needs). Consequently, agents can be redirected without the need to modify code. Finally, agents are mobile, a natural solution to the needs of intelligence gathering. The ability for agents to suspend operations, move to another computer, and resume operations on command provides for various design/implementation options needed for rapid deployment.

3.3 Data management

A multi-agent system (MAS) for analyzing massive scientific data was developed successfully as flexible and economical solution for distributed data management.

Agents monitor the output of a simulation model/run. Anytime the simulation produces new data, the primary monitoring agent logically divides the new data into pieces and creates a new agent for each piece of the new data. Each agent responds to queries about the piece of data that they are responsible for.

In collaboration with scientists from various labs and universities, this approach has been used to render data from a supernova simulation experiment under development. Using 100 time steps of data segmented into 800 individual pieces, managed by 800 agents, running on conventional systems, the agents work together to produce a visual representation of the dataset (Fig. 4). The results indicate that a large system of software agents spread over the candidate dataset can be an adaptable and cost effective method to aid scientists with validating the dataset.

4 References

1. Patel, M., Forward: Advances in the Evolutionary Synthesis of Intelligent Agents, 1st ed. Advances in the Evolutionary Synthesis of Intelligent Agents, ed. P.J. Mukesh, V. Honavar, and K. Balakrishnan. 2001, Cambridge: MIT Press. 480 pages.
2. Subrahmanian, V.S., et al., Heterogenous Agent Systems, 1st ed. 2000, Cambridge: MIT Press pages.
3. Demarco, T. and P.J. Plauger, Structured Analysis and System Specification. 1985, New York: Prentice Hall. 352 pages.
4. Sheldon, F.T., K. Jerath, and H. Chung, "Metrics for Maintainability of Class Inheritance Hierarchies," Jr. of Software Maintenance and Evolution, 2002. 14(3): pp. 147-160.
5. Booch, G., Object-Oriented Design with Applications, 2 ed. 1991, Redwood City: Benjamin/Cummings. 608 pages.
6. Chin, R.S. and S.T. Chanson, "Distributed, Object-Based Programming Systems," ACM Computing Surveys, 1991. 23(1): pp. 91-124.
7. Thorn, T., "Programming Languages for Mobile Code," ACM CS 29, No. 3 (1997)." ACM Computing Surveys, 1997. 29(3): pp. 213-239.
8. Kim, H.Y., Jerath, K. and Sheldon, F.T., Assessment of High Integrity Components for Completeness, Consistency, Fault-Tolerance and Reliability, in Component-Based Software Quality: Methods and Techniques, A. Cechich, M. Piattini, and A. Vallecillo, Editors. 2003, Springer-Verlag: Heidelburg. pp. 259-86.
9. Geihs, K., "Middleware Challenges Ahead," Computer, 2001. 34(6): pp. 24-31.
10. Potok, T.E., N.D. Ivezic, and N.F. Samatova. "Agent-based Architecture for Flexible Lean Cell Design, Analysis and Evaluation," in Working Conf. on Design of Info. Infrastructure Sys., Melbourne Australia: Kluwer, 2000, pp. 181-8.
11. Ivezic, N., T.E. Potok, and L. Pouchard, "Multiagent Framework for Lean Manufacturing," IEEE Internet Computing, 1999. 3(5): pp. 58-9.
12. Potok, T.E., Phillips, L., Pollock, R., Loebl, A. and Sheldon, F.T. "Suitability of Agent-Based Systems for Command and Control in Fault-tolerant, Safety-critical Responsive Decision Networks," in ISCA 16th Int'l Conf. on Parallel and Distributed Computer Systems (PDCS), Reno NV: ISCA, 2003.
13. Kavi, K.M., M. Aborizka, and D. Kung. "A framework for the design of intelligent agent based real-time systems," in Proc. 5th Int'l Conf. on Algorithms and Architectures for Parallel Processing, Beijing: IEEE CS, 2002, pp. 196-200.
14. Kavi, K.M. and H.B. D.C. Kung, G. Pandcholi, M. Kanikarla and R. Shah. "Extending UML to modeling and design of multi agent systems," in Proc. of 2nd Intl Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS collocated with ICSE03), Portland: Springer, 2003.
15. Rao, A. and M. George. "BDI agents: From theory to practice," in Proc. First Int'l Conf. on Multi-Agent Systems (ICMAS-95), San Francisco: AAAI Press, 1995, pp. 312-319.
16. Papavassiliou, S., et al. "Integration of Mobile Agents and Genetic Algorithms for Efficient Dynamic Network Resource Allocation," in Sixth IEEE Symp. on Computers and Communications (ISCC'01), Hammamet, Tunisia, 2001, pp. 456-63.
17. Jennings, N.R., K. Sycara, and M. Wooldridge, "A Roadmap of Agent Research and Development," Jr. of Autonomous Agents and Multi-Agent Systems, 1998. 1(1): pp. 7-38.
18. Sycara, K., et al., "Distributed Intelligent Agents," IEEE Expert, 1996. 11(6): pp. 36-46.
19. Griss, M.L. and G. Pour, "Accelerating Development with Agent Components," Computer, 2001. 34(5): pp. 37-43.
20. Cabri, G., L. Leonardi, and F. Zambonelli, "Mobile-agent Coordination Models for Internet Applications," Computer, 2000. 33(2): pp. 82-9.
21. Gelernter, D. and N. Carriero, "Coordination Languages and Their Significance," 1992. 35(2): pp. 96-107.
22. Labrou, Y., T. Finin, and Y. Peng, "Agent Communication Languages: The Current Landscape," IEEE Intelligent Systems, 1999. 14(2): pp. 45-52.
23. Vogler, H., T. Kunkelmann, and M. Moschgath. "An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transactions," in Int'l Conf. on Parallel and Distributed Systems, Seoul: IEEE, 1997, pp. 268-74.
24. Abadi, M., "Secrecy by Typing in Security Protocols," Jr. of the ACM, 1999. 46(5): pp. 749-786.
25. Liu, Z., et al. "Pluggable Active Security for Active Networks," in IASTED Proc. Int'l Conf. PDCS, Nov. 2000," in Int'l Conf. PDCS, Las Vegas: IASTED, 2000.
26. K-Praxis, "SonicBoomerang: Semantic Prime-Time for Intelligent Information Agent Technologies." 2003, K-Praxis, http://www.k-praxis.com/archives/000036.html.
27. CCNMatthews, "Intelligence Gathering Service Wins Information Highways Magazine's 2002 E-Content Innovation Award." 2003, CCNMatthews: Toronto, http://www.ccnmatthews.com/scripts/headlines1.pl.
28. Potok, T., Elmore, M., Reed, J. and Sheldon, F.T. "VIPAR: Advanced Information Agents Discovering Knowledge in an Open and Changing Environment," in SCI 2003 Proc. 7th World Multiconference on Systemics, Cybernetics and Informatics (Special Session on Agent-Based Computing), Orlando: IIIS, 2003.
29. Mladenic, D., "Text-learning and Related Intelligent Agents: A Survey," IEEE Intelligent Systems, 1999. 14(4): pp. 44-54.
30. Klusch, M., "Information Agent Technology for the Internet: A Survey," Data & Knowledge Engineering, 2001. 36.
31. Samatova, N.F., T.E. Potok, and M.R. Leuze, "Vector Space Model for the Generalized Parts Grouping Problem,"

Robotics and Computer-Integrated Manufacturing, 2001. 17(1-2): pp. 73-80.
32. Anderberg, M.R. "Cluster Analysis for Applications," in Probability and Mathematical Statistics, 19, New York: Academic Press, 1973.