RECURSION SOFTWARE, Inc.

Software for Software Developers™

VOYAGER®

Voyager and Software Agents Applied to Simulation-Based Training and Exercise Management

By James C. Bender, M.S.

Recursion Software, Inc.

December 9, 2004

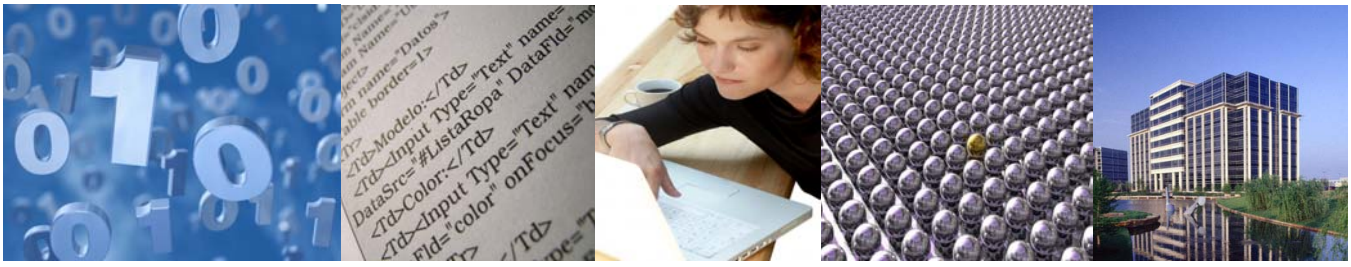


TABLE OF CONTENTS

VOYAGER®

Voyager and Software Agents Applied to Simulation-Based Training and Exercise Management

By James C. Bender, M.S.

1 Introduction

An Incident Management System, real or simulated, is an exercise within a distributed environment with many diverse clients (first responders, agencies, sensors, databases, etc.) networked over distances by radio or computer. Ideally, a wireless system that is seamlessly scaleable and provides high-level security without absorbing huge amounts of bandwidth can be adapted to the needs of the National Incident Management System (NIMS) immediately, and in the near future enhanced with additional components to broaden the scope of such a powerful tool [3].

Voyager®, a distributed development platform for Java™ applications, provides the necessary data communications infrastructure for the NIMS simulation-based training. Currently Voyager is deployed in secure, critical applications within the Defense and Finance industries and is an integral part of several critical infrastructure control systems. Voyager is an established, proven technology with inherent capabilities that continue to be developed; among these are Peer-to-Peer technology and Mobile Agent Technology. Other potential development areas are Voice Over Internet Protocol (VOIP) and Grid Computing.

Diverse clients involved in a training scenario can be linked by using wireless communications and Mobile Agent based problem solving capabilities. At the lowest level, the clients have PDA's or laptops available to them and can communicate in a distributed fashion with a publish/subscribe feature allowing a hierarchical communication from the Incident Commander.

2 System Description

2.1 The Underlying Science

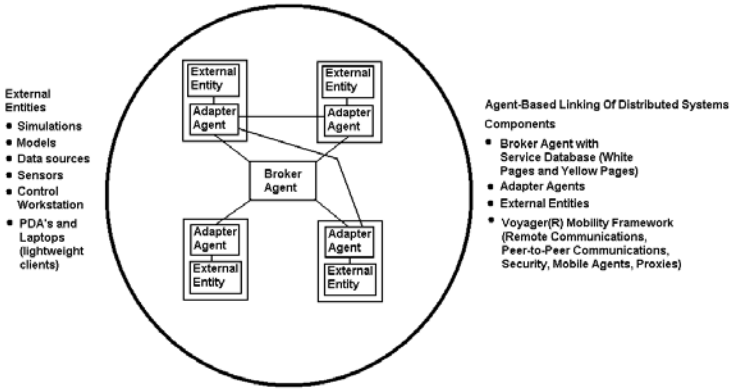
An agent-based approach can be used to link and manage a heterogeneous system composed of datasets, sensors, other simulations (such as weather simulations), and Incident Command System (ICS) exercise communications to provide a realistic experience to participants using PDA's, laptops, and personal workstations. This approach includes mobile agents for reducing bandwidth usage. The agent-based approach for linking distributed systems has been an ongoing research topic that has been maturing. In parallel with the research into using agent-based architectures for distributed systems, has been the research into using mobile agents in ad-hoc wireless networks. This is relevant to the ICS (now incorporated into the NIMS), as that field has pioneered use of wireless computing devices by first responders and by Incident Commanders.

2.2 Technology Functional Capabilities

The architecture provides the framework for interconnecting a heterogeneous collection into a distributed training and exercise management system, using Voyager. Voyager provides the mobility infrastructure for code mobility and remote communications. The architectural components are external actors (user client workstations,

simulations, datasets, and sensors), Adapter Agents (Adapters), the Broker Agent (Broker), and Mobile Agents [3] [5] (see Figure 1). These components provide the environment for linking the various pieces into an Incident Management Training and Exercise system [4].

Figure 1. Agent-Based Linking of Distributed Systems Architecture



2.2.1 Adapter Agent

The Adapter provides a connection between a local client and the larger system (similar in concept to the Dartmouth Generic Local Agent [1]). Each Adapter implements the Adapter interface. The messages and semantics are described in declarative notation (candidates being predicate calculus, the ontology language OWL, KQML, or KIF). The Adapter communicates with the local client and provides a consistent interface to the other system components. The local client can be as diverse as a simulation, an external sensor, or user client workstation. The Adapter maintains a local table with available services provided by the client and known services that the client uses [5]. A service description entry has name, location, semantic description, method name, parameters, and returned values.

2.2.1.1 Standard Client Interface

The Adapter Agent accepts requests and provides responses, and implements the client standard interface commands:

- Service query (find a service)
- Service announcement that includes service name, semantic description, parameters, and outputs
- Service dispatch (invoke a service)
- Command table query (retrieve an entry from the local table for a service)
- Disconnect a client from the system
- Service dispatch and command table query are available to Adapter Agent peers.

2.2.1.2 Adapter Agent-to-Broker Agent Interaction

The Adapter is a Broker client, as the Adapter relies upon the Broker for suggesting candidate services for fulfilling a client request for service. The Adapter evaluates available services and chooses which to use by evaluating service descriptions [1][4]. The Adapter creates a Mobile Agent as a helper and dispatches the agent to a remote platform to perform specific tasks. The goals are to reduce network traffic and to reduce vulnerability to network outages. Interactions between Adapter and Brokers include [12]:

- Publishing service announcements to Broker when requested by the client
- Requesting a service from the Broker Agent by category
- Placing a “standing request” for a service that is not presently available with the Broker Agent so that the Broker Agent can notify the Adapter Agent when one becomes available
- Notifying Broker Agent when a service becomes degraded or unavailable
- Accessing the Broker’s service database to retrieve service descriptions

2.2.1.3 Peer-to-Peer Interaction Between Adapter Agents

The peer-to-peer relationship between Adapters makes the architecture more fault-tolerant and not dependent on a centralized routing system. Interactions between Adapters include:

- Receive requests for service from other Adapter Agents and provide responses
- Request service directly from a known peer Adapter Agent
- Provide data quality metrics upon request
- Provide informational messages to service requestors regarding pending requests

2.2.2 Broker Agent

The Broker brokers links between Adapters. Peer-to-peer linking between Adapters in an unstructured network topology will involve too much bandwidth, so brokering is used instead. The Broker can be implemented as a scalable service by using J2EE technology that minimizes overloading associated with a single computational entity [10]. Adapters publish their services with the Broker that lists the services in the service database, accessible either as a “White Pages” (by name) or as a “Yellow Pages” (by category). Adapters make queries for services that broadly match their needs. The Adapters evaluate candidate services through examining the semantic descriptions and input and output parameters [8]. Once a service has been identified, the Adapter can communicate with it directly, without going through the Broker [8].

2.2.2.1 Service Registration

The Broker accepts new service announcements from Adapters and issues a unique service identifier [5]. An entry is added to the service. An entry includes service name, semantic description, parameters, and outputs. The parameters and outputs describe data types and semantics, as needed by Adapters choosing services. New service announcements are locked during the registration process to preserve data integrity.

2.2.2.2 Service Notifications

The Broker will accept informational messages from Adapters for cases where a service has been found to be degraded or unavailable. The Broker will alter the service status to denote the condition. These annotations regarding service quality can be queried by Adapters to aid in selecting services.

2.2.2.3 Queries

The Broker provides candidate services to requesting Adapters, including the description for inclusion in the agent’s local command table [12]. As previously noted, a suitable notation is needed to describe service semantics so that Adapters have sufficient knowledge for selecting services. Queries can be by category (from the “yellow pages” view) or by name (from the “white pages” view). The Broker will supply a proxy for a service to the Adapter that provides the methods needed to interact with the service. Once the linking occurs, the Adapters interact independently from the Broker.

2.2.2.4 Service Database

The Broker maintains a service database that is externally accessible by Adapters [12]. This can be implemented with UDDI from Web Services. The database needs to support access locking during updates. The white pages and yellow pages views into the database need to be available to Adapters.

2.2.3 External Actors

External actors can be discrete event simulations in the Incident Management Training and Exercise system, external sensors, or Incident Management workstations. Each external actor must implement the standard interface to Adapter. An external actor requests services through their Adapter using the standard interface and receives replies, possibly with data. They also request their Adapter to announce new services, including their description. When requested, external actors are expected to respond to requests for their advertised services. They may publish events and subscribe to other events. Selected external actors perform system control and administrative functions, since systems need to be started and shut down in an orderly fashion. Systems also need to control access, as authentication must be used and enforced. Authentication and access control will be done using the Voyager security library in addition to facilities provided by J2EE. External actors will use Web Services and WSDL to interact with their Adapters.

2.2.4 Mobility Infrastructure with Voyager

Voyager provides infrastructure for mobility in a distributed system. The components are remote communications, security, code mobility (Mobile Agent), and proxies.

2.2.4.1 Remote Communications

Remote communications are used by all other system components. Voyager capabilities include [11]:

- Sending and receiving event notifications use multicasting and “publish-subscribe” facilities
- Network transport (sockets)
- Messaging (one-way, synchronous, or “future” messages, where the result will be returned at some later time)

2.2.4.2 Peer-to-Peer Communications

Peer-to-peer communications between first responders is provided with Voyager as the platform [2]:

- Join or leave a group
- Join in virtual meetings
- Create a communications path
- Chat or instant messaging
- Find a resource
- Send a request
- Query peer status
- Propagate messages
- Route messages
- Deliver messages when connected [7]

2.2.4.3 Security

In a distributed, agent-based system, security is a major issue [6]. Needs include exchanging appropriate data while protecting sensitive data, preventing sensitive data from propagating, and segregating systems to protect data [9]. We use Voyager security library and API for the security infrastructure over and above the basic security facilities offered by J2EE. We answer the question whether code being executed is trusted and if the user who initiated the code execution is also trusted using the Voyager security framework [11].

2.2.4.4 Code Mobility

Mobile Agents implement the Mobile Agent interface. Mobile Agents are created and dispatched to remote platforms by Adapters to interact with another Adapter without network traffic overhead [13]. Parent Adapters and their child Mobile Agents form a peer-to-peer group for security purposes. Mobile agents usually have hard-coded

behavior in Java, but can draw from intelligent systems technology, using an engine and a knowledge base. Mobile Agents will evaluate services for quality and reliability and report that status back to their parent Adapters. Mobile Agents can also be used for load-balancing purposes to transfer computationally intensive work to underutilized platforms within the system confines. This implies that platform utilization information is available to Adapters.

2.2.4.5 Proxies

Voyager provides dynamic proxy generation that eliminates the need to manually generate stubs, skeletons, and helper classes for remotely enabled classes. Where needed for RMI, stubs and skeletons are automatically generated at runtime [11].

2.3 Capability for Accessing Community-Specific Real-World Data

Each existing system has an Adapter that publishes its interface to the Broker, which offers functionality and data to distributed consumers. The Adapter requests a service (data or an operation to be performed). If the service is available, the request is forwarded to the Adapter for that service.

2.4 Infrastructure Required

The infrastructure needed is at least intermittent network connectivity, computing platforms that can run the Java Virtual Machine (JVM), and Voyager installed. PDA's can have a lightweight Voyager installed to add capabilities beyond a simple web-browser.

2.5 Current Technology Users

2.5.1 Voyager

Voyager is a mature commercial product that has been used in Defense, Finance, Telecommunications, and Control Systems for various industries. Voyager provides the infrastructure for mobility in a distributed system [11]. The components are remote communications, security, code mobility, and proxies.

2.5.2 Agent-Based Distributed System Linking

Using agents as the basis for linking distributed simulations systems has been used in ongoing research conducted by Linda F. Wilson, Ph.D. and her students at Dartmouth College [1] [5] [8] [10]. General agent-based and agent-directed simulation has appeared in the literature since at least 1997.

2.5.3 Mobile Agents in Wireless Networks

There has been work on applying mobile agents in wireless networks for at least a decade. Most recently, there has been ongoing research at the College of Engineering at Drexel University exploring the issues involved with using mobile agents with PDA's and laptops over a secure ad hoc wireless network test bed (SWAT) [9].

2.6 Integration with other Models or Simulation Tools

The capability will need to be integrated with other models and simulation tools by installing a JVM, Voyager, and implementing the adapter agents for the different systems.

2.7 Standards and Interfaces

The Voyager system uses the CORBA and J2EE industry standards to implement the remote communications and mobile agent capability. This readily integrates on diverse platform configurations due to the ubiquity of Java and the JVM.

2.8 Fidelity Level

The architecture is designed to select the best available data sources so to achieve the best performance. The brokering and knowledge about data source properties by the adapter agents allow for intelligent selection.

2.9 Scalability

A major lesson learned from building systems using agent-based linking was that simulation and data source ranking and selection needs to be distributed. Another key piece is the mobile agent, which is needed to reduce network bandwidth usage. A mobile agent can be dispatched to a remote platform, interact with the system hosted there, which might involve receiving large data volumes from the system, and then send the results back to the client from which the mobile agent had been dispatched.

2.10 Open Architecture

Voyager and the agent-based architecture for linking distributed systems uses industry standard technology (CORBA and J2EE) for the infrastructure for integrating distributed simulation systems. Voyager is a proprietary technology, but one built from open standards (Java, J2EE, and CORBA, among others).

2.11 Restrictions on Use in an Open Environment

The only restriction is that for platforms that require remote communications and have a need to allow mobile agents, Voyager must be installed.

3 Conclusion

A system can be implemented that will leverage the mobility infrastructure already available from Voyager (remote communications, security, code mobility, and proxies). That is sufficient to provide the basic architecture that uses agent-based linking for distributed systems. The components are the broker agent, service database with white pages and yellow pages, and adapter agents to connect the various external entities (which include simulations, exercise control workstations, data sources, sensors, etc.). When peer-to-peer communications are implemented with Voyager as the platform, the adapter agents will use that for interacting. The architecture is an excellent way to interconnect a simulation-based training and exercise management system that includes heterogeneous components.

4 Bibliography

1. G. Ayorkor Mills-Tettey, Greg Johnston, Linda F. Wilson, Joseph M. Kimpel, "The ABELS System: Designing an Adaptable Interface For Linking Simulations," Proceedings of the 2002 Winter Simulation Conference.

2. Daniel Brookshier, "Overview of JXTA," at Developer.com, SAMS Publishing, 2002.

3. Nicholas R. Jennings, "On Agent-Based Software Engineering," *Artificial Intelligence 17* (2000), pp.277-296.

4. Anush Kumar, Linda F. Wilson, Thomas B. Stephens, and Jeanne Sucharitaves, "The ABELS Brokering System," 35th Annual Simulation Symposium, ASTC, April 2002.

5. John P. Murphy, Ayorkor Mills-Tettey, Linda F. Wilson, Greg Johnston, Bin Xie, "Demonstrating the ABELS System Using Real-World Scenarios," 2003 Symposium on Applications and the Internet.

6. T. I. Ören, S. K. Numrich, A. M. Uhrmacher, L. F. Wilson, Joseph Kimpel, "Agent-Directed Simulation—Challenges to Meet Defense and Civilian Requirements," in J. A. Joines, R. R. Barton, K. Kang, P. A. Fishwick, (Hrsg.), Proceedings of the 2000 Winter Simulation Conference, December 10-13, 2000.

7. Venkata N. Padmanabhan, Kunwadee Sripanidkulchai, "The Case for Cooperative Networking," First International Workshop on Peer-to-Peer Systems, 2002.

8. Thomas B. Stephens, "Improving a Brokering System for Linking Distributed Simulations," Dartmouth College CS TR2001-39, May 30, 2001.

9. Evan Sultanik, Donovan Artz, Gustave Anderson, Moshe Kam, William Regli, Max Peysakhov, Jonathan Sevy, Nadya Belov, Nicholas Morizio, Andrew Mroczkowski, "Secure Mobile Agents on Ad Hoc Wireless Networks," in *Proceedings of the 15th Innovative Applications of Artificial Intelligence Conference (IAAI-03)*, pages 129-136, 2003.

10. Boleslaw Szymanski, Gilbert Chen, and Linda F. Wilson, "Component-Based Simulation and Agent Brokering: Towards Ad Hoc Simulations in Crisis and Emergency Management," *Proc. Computer Networks and Distributed Systems Modeling and Simulation, CNDS'03*, January 2003

11. *Voyager® 4.7 Documentation*, Recursion Software, Inc., 2003.

12. Linda F. Wilson, Bin Xie, Joseph M. Kimpel, G. Ayorkor Mills-Tettey, Greg Johnston, "The Design of the Distributed ABELS Brokering System," Presented at the 6th IEEE International Workshop on Distributed Simulation and Real-Time Applications, October 2002.

13. Yanyan Yang, Omer F. Rana, David W. Walker, Christ Georgousopoulos, Roy Williams, "Mobile Agent on the SARA Digital Library," CACR Technical Report CACR-186, March 2000.

About the author

Jim Bender is a senior software engineer at Recursion Software, Inc. He may be contacted by email at engineer@recursionsw.com.

VOYAGER®

Recursion Software, Inc.
2591 North Dallas Parkway
Suite 200
Frisco, Texas 75034
1.800.727.8674 or 972.731.8800
www.recursionsw.com

RECURSION SOFTWARE, Inc.