Simulation Monitoring Using Intelligent Mobile Agent Technology

Bob DeAnna, Chief Technology Officer bdeanna@recursionsw.com
Bob Peterson, Senior Architect and Engineer bpeterson@recursionsw.com

Recursion Software, Inc.
2591 North Dallas Parkway
Suite 200
Frisco, TX 75034

Keywords: mobile agent, intelligent agent, Java and .NET interoperability, self- healing, self- monitoring

Abstract

A next generation intelligent, mobile agent platform offers a solution to the current problems faced in simulation environments, one of which is processing large amounts of data across less-than-reliable mobile networks over an increasing number of nodes ranging from enterprise server to handheld and embedded devices. An agent-based approach reduces bandwidth usage, filters and analyzes data at its source, as well as monitors and manages the simulation system. Voyager *Edge*, used in the Department of Defense's Joint After Action Review project, demonstrates the approach discussed in the paper.

INTRODUCTION

The DoD, NASA, FAA, DOE, commercial industry and academia use simulators to conduct studies otherwise infeasible or impossible to perform and measure. As distributed simulations grow in node count, geographic dispersal, heterogeneity, and overall complexity, configuration, startup, monitoring, shutdown, and retrieval of results become increasingly difficult. Some simulations require a person's keystrokes on each system console to start and stop a simulation node. Localizing and automating configuration and management of a distributed simulation controls costs by reducing the required headcount, the time between runs, configuration errors, and the time required to create post-run evaluations.

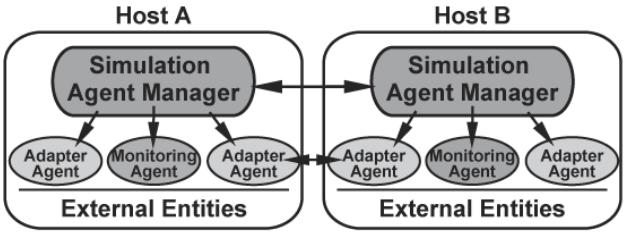
Next generation simulation systems will utilize an agent-based approach to link and manage participating nodes providing pertinent, timely information to participants using PDAs, laptops, and personal workstations. This approach takes advantage of mobile agents to reduce bandwidth usage, filter and analyze data at its source, as well as to monitor and manage the simulation system.

The solution proposed here is equally applicable using conventional simulation techniques, e.g. OneSAF Testbed Baseline, as well as next-generation agent-based simulations, e.g. agent-based components of the DoD's Joint After Action Review (JAAR) discussed later in this paper. [1]

SIMULATION SYSTEM ARCHITECTURE

The agents discussed below are part of an overarching architecture of a widely distributed knowledge network, running on a heterogeneous mix of wired and wireless networks and hardware devices that must be as self-managing and self-healing as possible. The following sections will discuss in more detail aspects of this next-generation architecture and resulting real-time simulation system capabilities.

Using Adapter Agents and Agent Managers, intelligent, mobile agent platforms provide the infrastructure for mobility in a distributed system including remote communications, security, code mobility (Mobile Agents), and proxies.



Hosts: Workstations/ Servers, Sensors, PDAs, and SmartPhones
Key Architectural Components: Agent Mangers, Monitoring Agents, and Adapter Agents
External Entities: Various Data Sources, Data Systems, Application APIs, Simulation Agents, and Simulation Applications

Figure 1. Agent-Based Linking of Distributed Systems Architecture

Simulation Host Manager Agent Capabilities

Simulation Host Managers are responsible for monitoring the simulation processes occurring on the same wide variety of nodes just mentioned.

- A configuration type of Simulation Host Manager Agent acquires from a central repository the configuration data needed by a single node of the simulation. When dispatched, each Simulation Host Manager Agent carries the configuration data to the

appropriate node and updates the node's configuration. When the reconfiguration is complete, the agent sends a "Configuration successful" message and terminates.

- An execution type of Simulation Host Manager Agent for each node involved in the simulation is configured from a central repository and dispatched to the node. When the simulation is to be started on the node, a decision that could be based on time or on arrival of a "go" message, the agent starts execution of the simulation on the node.
- Once the simulation starts, the execution agent monitors startup and execution, and reports status as appropriate. In many cases the only status messages will be "initialization complete" and "simulation node terminated normally." Of course, the execution agent will also report anomalies detected during execution of the simulation. Detectable anomalies include errors written to log files, alert messages broadcast to listening clients, a missing event, or other complex events. [2]
- When directed to do so, the execution agent shuts down the simulation node and starts the process of reporting results.

Adapter Agent Capabilities

- The role of an Adapter Agent is to provide a connection between a local client and the larger system. Each Adapter implements the Adapter Agent interface.
- The Adapter Agent 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 Agent maintains a local table with available services provided by the client and known services that the client uses.
- An Adapter Agent also acts as an Agent Manager client. It relies upon the Manager for suggesting candidate services for fulfilling a client request for service.
- The Adapter Agent evaluates available services and chooses which to use by evaluating service descriptions.

Each existing system has an Adapter that publishes its interface to the Manager, which offers functionality and data to distributed consumers. The Adapter Agent knows how to access, filter and analyze its local simulation data.

Mobile agents are key to scalability for reducing network bandwidth usage. An Adapter or Agent Manager can dispatch a Mobile Agent to a remote platform to interact with the hosted system. This interaction might involve receiving large data volumes from the system, filtering and analyzing the data, and sending the results back to the client from which the mobile agent had been dispatched. This streamlined process results in reduced network traffic and reduced vulnerability to network outages.

Added value is achieved though peer-to-peer interaction among Adapters via Agents, thereby making the architecture more fault-tolerant and independent of a centralized routing system.

Adapter Agent Managers

The Agent Manager brokers links between Adapter Agents. Managers are themselves implemented as scalable agents. Adapters publish their services with the Manager, which maintains a local service database. Adapters make queries for services that match their needs. Once a service has been identified, the Adapter can communicate with it directly, without going through the Manager, regardless of the service's location in the network.

Mobile Agents are created and dispatched to remote platforms by Managers to interact with another Adapter Agent or External Actors. Mobile Agents can have hard-code behavior in Java and will also draw from intelligent systems technology, using an engine and a knowledge base. Additionally, enabled code mobility can facilitate dynamic loading of the latest versions of the mobile agent software. This results in an agent software infrastructure that keeps itself current, regardless of how widely distributed it is, with no additional work from simulation event controllers or administrators.

External Actors

External actors can be discrete event simulations in a Simulation Training and Exercise system, external sensors, or Simulation Training workstations. Each external actor can optionally implement the standard interface to the Adapter Agent. An external actor requests services through its Adapter, or vice-versa, using the standard interface and receives replies with data. The other option is to create specific Adapter Agents that have knowledge for communication with external simulation and training systems such as those based on Test and Training Enabling Architecture (TENA), which uses IIOP via a real-time CORBA implementation, High Level Architecture (HLA) Run Time Infrastructure (RTI), and Distributed Interactive Simulation (DIS) Protocol Data Unit (PDU) which use UDP and TCP/IP.

Monitoring Agents and Applications

One or more management consoles actively manage the agents running the simulation with each console

supervising a subset of the simulation nodes. These administrative console user interfaces must interoperate on a number of different form factors such as tablets, PDAs and smartphones, as well as on conventional workstations. This enables the simulation exercise observers and controllers to access simulation results, as well as the performance and status of the simulation exercise and systems regardless of their location and what device they use.

Interface Agents, which are embedded within the management applications and are compatible with embedded operating systems and virtual machines, enable advanced monitoring by using the protocol that best matches the wireless network capabilities. Of course, this functionality will only be realized if the agent platform itself can run on the same wide range of embedded operating systems such as Windows Mobile, Embedded Linux, Symbian, PalmOS, etc.

In addition to functioning on multiple devices and corresponding form-factors, Administrative Consoles should also have the flexibility to provide authorization-based variations geared for Administrators with varying levels of authority. Some administrators may have authorization to initiate, stop, and restart systems constituting a simulation exercise, while others are strictly observers of the simulation systems, their changing states and performance levels.

Security Agent Managers are ideally suited to perform this authentication, both on the node/device that the management application is running, but also on simulation nodes being monitored, where the Managers will be used to verify any administrative requests from the management nodes/devices.

A simulation controller/administrator operating an administration console can create simulation monitoring Task Agents, move them to the assigned node, run a simulation process, and return them to the console with the results. While at the assigned node, the monitoring Task Agent tracks the progress of the simulation, reporting significant events back to the consoles.

Simulation monitoring Task Agents can also be used for load-balancing purposes to transfer computationally intensive work to underutilized resources. They can direct a simulation agent to move its processing to a node with greater available processing power. This of course must be weighed with any additional network traffic or simulation delays that might result.

Simulation Monitoring Agents & Agent Manager Capabilities

Simulation Monitoring Task Agents are responsible for managing a simulation and training exercise as well as monitoring its status. This is a critical requirement for next-generation, self-healing, and self-managing agent based systems. More specifically, these agents need the characteristics outlined below.

- Monitoring Agents will pre-process, filter, polish, and extract features from raw data on edge devices, ensuring that knowledge is transmitted rather than large amounts of raw data.
- Monitoring Agents will select relevant subsets of data, remove noise and outliers, and decide on a strategy for parsing the data in the most efficient manner to a variety of devices across wireless networks.
- Next-generation intelligent, mobile agents can communicate using a number of different protocols to best adjust to network traffic issues and also to best communicate with the simulation processes running on their local node. The protocols should include standards-based protocols, such as SOAP (Web Services), XML-RPC, IIOP (CORBA), RMI (Java), listed in decreasing network requirements and increasing speed.
- If the agent platform also supports a protocol that exceeds the speed of any of these protocols, this is an added benefit, since the performance of these monitoring agents must be extremely robust, and their network bandwidth requirements held to an absolute minimum.
- Monitoring agents can collect different sets of simulation results, performance, and resource usage information while minimizing network usage, resources usage, and maximizing security. The processing power available to all nodes involved in the simulation is also maximized to any device, whether enterprise or handheld/embedded CPUs, or storage devices.
- Monitoring agents are dispatched from the Administrative tool(s) and/or dynamically loaded and executed on targeted simulation nodes. These Agents are responsible for launching the simulation process at the node where they reside, monitoring the simulation, and returning simulation and performance results back to one or more nodes running Administrative Tools.
- All client-to-agent and agent-to-agent communication will be encrypted using a pluggable protocol framework that can allow for SSL or TLS implementations utilizing public key cryptography. Simulation agents will travel with public keys, but only agent managers will contain private keys, which will be accessible to a simulation agent arriving or resident at a node, only upon verifying the agent with the security agent manager.
- Authentication will be accomplished using X.509 client and site certificates. Site certificates should be resident at each simulation node, and again are

overseen by security agent managers. Simulation agents will travel with client certificates containing its role information, as well as the identity and roles of the Administrator deploying it (see below).

- Authorization of Agents and Administrators is based on their roles, using a Role-Based Access Control Architecture (RBAC)[3-4]. A Super User is responsible for assigning roles to Administrators.
- An Agent Administrator assigns agent roles using the Agent Monitoring tool. The creation of public and private keys and X.509 Certificates, as well as their association with Agents and simulation nodes is also managed by the Agent Administrator using the same tool.
- Administrator tool-based security is performed by a security Agent Manager resident on the management application nodes. The role of the administrator will be used to determine what capabilities he or she can perform, such as creation of keys, certificates, association of such with Agents and nodes, and deployment, management, and monitoring of Agents and the simulation processes for which they are responsible.
- Additionally, agent managers resident on the nodes actually running the simulation will verify and validate arriving agents using the same RBAC architecture. The client certificate, containing identity, key, and role information will be used by the Security Agent Manager to determine if the Agent may arrive at its node and start the simulation process and monitoring desired.

Node Monitoring Agents Capabilities

Node Monitoring Task Agents are responsible for determining the current status of a node. This is a critical requirement for next-generation self-healing and self-managing agent based simulation systems. More specifically, these agents need the characteristics outlined below.

- Agents ideally can run in both Java and .NET virtual machines. In this way, the monitoring agent does not require an additional virtual machine to be loaded on the simulation node. If that simulation node is running a .NET simulation application or process, the agent can run on the same .NET runtime. Similarly for Java, if that simulation node is running a Java simulation application or process, the agent can run on the same Java runtime.
- Agents, and the platform they run on, must be lightweight and not resource intensive. These agents run in a process that is separate from the simulation process they are monitoring. In this way these monitoring agents do not impact the simulation system in any way. If a monitoring agent encounters problems, it will not impact the corresponding simulation system that it is monitoring.
- Agents must be autonomous, mobile and, to a degree, self-healing. More specifically these agents need to be able to operate (i.e. monitor) without a network connection and report the information back to a Administrative Tools and Monitoring Agent Managers once it detects the network connection has been re-established. Agent mobility and self-healing capabilities are in part related. If these monitoring agents are mobile, they have the ability to move to a nearby node, if it is determined that the simulation node is running low on resources.
- Monitoring agents must be manageable via standard mechanisms such as Java Management Extensions (JMX). This enables agent access by multiple standards-based administrative tools in addition to the ones provided by the intelligent mobile agent platform.
- Monitoring Task Agents must be able to measure and react to the changing properties of the following aspects of a node, be it an embedded device or enterprise server, or anything in between.

Node Monitoring Task Agents must have the ability to obtain the status of the hardware and network connections available. Specifically this includes:

- Network Adapter – Used to describe the network adapters currently available in the system, such as IEEE 802.3 (Ethernet) or WirelessLAN.
- Link Protocol - Used to describe the link protocols running on the system, such as IEEE 802.11a, and to monitor the state of network connections.
- Battery – Used to access information about a battery in the system, such as percent charge and life remaining.
- Processor – Used to access information about the system microprocessor, such as the manufacturer, the current processor frequency, and whether streaming extensions are supported.
- Platform – Used to access information about the system platform, such as when the system enters suspend mode or hibernate mode or shutdown.

This data can be processed by a Monitoring Task Agent to extract valuable knowledge, such as its node's current state of:

- Power – Information about system power, such as the system power source (battery or external AC) or the aggregated system battery charge.
- Connectivity – System connectivity information, such as whether the system has at least one valid network interface connected to a network or whether a remote node is reachable.

- Bandwidth – To monitor and control the network bandwidth or transfer rate.

Task Agents can use this information to determine whether to move itself, or simulation agents resident on the same node, to nearby nodes to prevent system degradation or failure.

These agents may use any of several command languages, such as those written in Java or .NET languages or even C++ to describe their intelligence.

They may also leverage RETE-based expert system languages to incorporate reasoning to understand the state of the simulation node.

Finally, monitoring Task Agents can report results from local log files and databases located on the node, as well as the output of the reasoning engine, onto the Management console, or possibly to nearby monitoring agents so that they may act accordingly and preventively.

Yet another viable option for next generation monitoring Task Agents, would be for Monitoring agents to subscribe to certain simulation messaging middleware for key warning or failure events that might necessitate action on its part.

All of these capabilities are available to simulation Adapter and monitoring Task Agents with the basic multi-language, multi-virtual machine, multi-operating system, and multi-protocol capabilities outlined earlier.

INFRASTRUCTURE REQUIREMENTS AND PROTOCOL INTEROPERABILITY

The minimum infrastructure needed for a next-generation, agent-based simulation architecture is intermittent network connectivity and computing platforms that can run the Java Virtual Machine (JVM), or the Microsoft .NET Common Language Runtime (CLR).

PDAs and smartphones will have a J2ME or Compact Framework compliant version of the intelligent, mobile agent platform installed for hosting and/or accessing Mobile Agents, dependent on the device specifications.

The next-generation intelligent mobile agent platform must use the Web Services (SOAP), CORBA (IIOP), Java (RMI), and XML-RPC industry standards to implement the remote communications and mobile agent capability. This will allow for agent communication with diverse systems due to the ubiquity of Java, .NET, and the ever-increasing support of their corresponding virtual machines.

PRIMARY GOALS OF AGENT-BASED SIMULATION SYSTEMS

A next-generation intelligent, mobile agent platform will allow clients to develop real-time distributed knowledge networks, derived from lessons learned from analogous simulation environments, for commercial, civilian, and military distributed computing systems. This will help them to:

- Drastically reduce the amount of raw data sent across wireless networks by enabling intelligent, onboard analysis utilizing mobile agents on edge devices to perform much of the data gathering, filtering, and analysis required by distributed computing platforms.
- Reduce operating and administrative costs, and make more efficient use of simulation environments (and the derivative real-world solutions), the wired and wireless networks over which they pass information, and the enterprise server, desktops, embedded devices, tablets, PDAs, and smartphones on which they query, filter, analyze simulation data and to which they send actionable knowledge.
- Provide a highly secure, easily managed, and self-healing simulation capability that enables real-time feedback and that can potentially gather suggestions for improvements to simulation exercises and distribute this information to a widely distributed audience regardless of the device to which they have access, and the network to which it is attached.

CURRENT AND RELATED WORK

Recursion Software's Voyager *Edge*™ intelligent software agents are currently being used by the Joint National Training Capability (JNTC) in the Joint After Action Review Tool Set (JAAR). Utilizing Voyager *Edge* Intelligent Mobile Agent Framework, the Government streamlined the access, filtering, and analyzing of data stored in heterogeneous systems. This solution was implemented in less than four weeks with the necessary code provided for intelligent agent mobility and communication between systems that were written in different platforms (Java and .NET).

Recursion Software worked closely with the Government's software engineers to design and execute a proof of concept using Voyager *Edge* to build Java-based intelligent mobile agents communicating with a OneSAF server, using auto-generated, fast binary protocol. The intelligent mobile agents accessed, filtered, and analyzed the data at the external system, translating the AAR data into joint AAR knowledge. The Government will use this prototype to build the remainder of the JAAR solution.

Voyager *Edge* is currently used or has been used by the following government contractors: Northrop Grumman, Raytheon, Lockheed Martin, and Boeing Integrated Defense Systems. As well, General Dynamics Advanced Information Systems is currently developing and testing cognitive intelligent software agents, or cogbots, using Voyager *Edge*, in the Cognitive Solutions Laboratory.

CONCLUSION

Simulation Host Managing Agents and Node Monitoring Task Agents enable powerful capabilities, such as self-healing and self-monitoring networks, within highly distributed simulation systems. Intelligent agents are ideally suited for simulation monitoring systems that span multiple wired and wireless networks, need to access varied external systems and involve participants utilizing a wide range of devices.

These agents reduce network traffic and handle network interruptions by both monitoring the host/nodes, filtering, and analyzing the data they collect all locally. They can proactively detect and warn of failing simulation processes or simulation agents, and also move them from hosts or nodes that they may detect to be over-utilized. This is accomplished in part by communicating with Adapter Agents that can utilize their multi- protocol, multi-language and messaging capabilities to access the external systems involved in the simulation.

Finally, it is important to note that this type of architecture can be equally effective in past, present, and future simulation systems. It is not a requirement that the simulation system itself be agent-based.

REFERENCES

[1] "Department of Defense Joint After Action Review: A Voyager *Edge* Case Study," Recursion Software. http://www.recursionsw.com/Solutions/inc/2006-12-JAAR_Case_Study.pdf

[2] A.R. Cassandra; D. Baker; M. Rashid. "CEDMOS Complex Event Detection and Monitoring System." MCC Technical Report CEDMOS-002-99, Microelectronics and Computer Technology Corporation, 1999. http://citeseer.ist.psu.edu/baker99cedmos.html

[3] Dongwan Shin; Gail-Joon Ahn; Sangrae Cho; Seunghun Jin. 2004. "A Role-based Infrastructure Management System: Design and Implementation." Concurrency and Computation: Practice and Experience, Vol. 16, No. 11, John Wiley & Sons, August 2004. http://portal.acm.org/citation.cfm?id=1064505.1064510

[4] Dongwan Shin; Gail-Joon Ahn; Prasad Shenoy. "Ensuring Information Assurance in Federated Identity Management," In Proceedings of the 23rd IEEE International Performance Computing and Communications Conference (IPCCC 04), Phoenix, Arizona, April 14-17, 2004. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1395193

BIOGRAPHIES

Bob DeAnna, Chief Technology Officer, Recursion Software

Bob has 22 years of experience in software architecture, development, and mentoring. Bob's expertise is in distributed application frameworks such as J2EE, CORBA, and ATMI. He has architected and developed applications and middleware in Java, C/C++, COBOL, PLI, and Assembler on operating systems ranging from MVS to Unix, Linux, and Windows. Bob has worked previously in numerous roles ranging from VP of architecture, senior consultant, lead engineer, and trainer/presenter. In these capacities, Bob has worked either as an employee or long-term consultant for such companies as Verizon, ATT, Swiss Re-Insurance, State Farm Insurance, Bloomberg LP, 3M Health Information Systems, Merant Software, and Zeosoft.

Bob received a BS in Mechanical Engineering from Rutgers University, and a continuing education degree in C/C++ and Unix Programming from New York University.

Bob Peterson, Senior Architect and Engineer, Recursion Software

Bob brings 33 years of system development experience to the Voyager *Edge* product team. He co-founded Advanced Coordination Technologies, L.L.C., where he applied his skills to creation of a suite of tools supporting process definition, simulation, configuration, deployment, and agent-based enactment process enactment and evaluation. Bob also worked on projects such as DARPA's Open Object-Oriented Database, DARPA's Trauma Care Information Management System (TCIMS), DARPA's Advanced Logistics Program (ALP), MCC's Infosleuth project, Elagent Corporation's Motera product, Integrated Concepts' Web-based electronic commerce products, and Integrated Concepts' Process Integration Environment.

After graduating from Austin College in 1972, Bob worked for the A. H. Belo Corporation, and then joined Texas Instruments, where he initially worked on semiconductor production line control software. In 1986, following his participation in a project to develop a network DBMS product for TI's minicomputer business, he helped establish TI's object-oriented database research team, housed in the company's corporate research laboratory. His research interests led to his involvement with TCIMS, ALP, and Infosleuth. Mr. Peterson is a co-inventor of two issued and two pending software patents.