December 2003

eAgents: An Approach for Modeling and Simulating Multi-Agent Systems

N. Narendra
Hewlett-Packard India Software Operations Ltd

EAGENTS: AN APPROACH FOR MODELING AND SIMULATING MULTI-AGENT SYSTEMS

N. C. Narendra
Hewlett-Packard India Software Operations Ltd.
ncnaren@india.hp.com

Abstract

The shift from traditional store-and-catalogue business transactions to internet-enabled e-commerce promises to change the way businesses interact with each other and with their customers in several fundamental ways. It has also been claimed that e-commerce has the potential to create an environment where companies will be at their most agile and marketplaces will approach perfect efficiency. Hence it is worthwhile to investigate how useful e-commerce would truly be. In other words, we would need to ask (and obtain answers for) questions such as: how to prevent decision-makers from being overwhelmed with too much information, how to efficiently build and manage certain kinds of e-commerce environments, etc. Many researchers are turning to architectures of collaborating agents to provide answers to these and other questions, since agents and multi-agent systems seem to be highly promising technologies for e-commerce applications. However, before we build and deploy multi-agent systems on a large scale, we need to be sure that they deliver as promised, and we need to determine this at minimal cost. Hence there is a need for modeling and simulating such systems.

In this paper, we first present a lifecycle-based approach to simulating multi-agent systems borrowing from ideas developed in Software Process Modeling research. We then present a reference architecture for multi-agent systems called eAgents. Using the eAgent architecture as a building block, we show how large multi-agent systems can be simulated, and their behavior observed and measured for different e-commerce models and situations.

Introduction and Motivation

The advent of e-commerce has revolutionized typically store-and-catalogue business transactions to such an extent, that it has been claimed that e-commerce will maximize the agility of businesses. Hence there is a need to investigate how true this claim would be. In other words, we need to investigate and answer questions such as: how to prevent decision-makers from being overwhelmed with too much information, how to efficiently build and manage certain kinds of e-commerce environments, etc. Many researchers are turning to architectures of collaborating agents to provide answers to these and other questions, since agents and multi-agent systems seem to be highly promising technologies for e-commerce applications. However, before we build and deploy multi-agent systems on a large scale, we need to be sure that they deliver as promised, and we need to determine this at minimal cost. Hence there is a need for modeling and simulating such systems.

To that end, in this paper, we first present a lifecycle-based approach to simulating multi-agent systems borrowing from ideas developed in Software Process Modeling research. We then present a reference architecture for multi-agent systems called eAgents. Using the eAgent architecture as a building block, we show how large multi-agent systems can be simulated, and their behavior observed and measured for different e-commerce conditions and situations.

The rest of this paper is structured in two parts. The first part of the paper discusses our approach to lifecycle-based multi-agent system simulation. In the third section, we introduce the reader to Software Process modeling, and our simulation approach. The fourth section discusses two well-known approaches to multi-agent system simulation. The fifth and sixth section s describe our simulation approach in greater detail.

The second part of the paper describes our eAgent reference architecture in the seventh section, which we propose as a means to model and represent agents for multi-agent simulation. In the eighth section, we show how our simulation approach is different from the other work in this area.

The ninth section concludes with directions for further research in this area.

Introduction to Agents and Workflow

Currently, there is not much consensus on what an "agent" is, and many definitions abound. For our purposes, we will adopt the following one-line definition [MASIF]: " An agent is a computer program that acts autonomously on behalf of a person or organization". To this we add another definition, which is "an autonomous software component that interacts with its environment and with other agents" (Griss, 2000). Hence we define an agent as "a software component that acts autonomously on behalf of a person or organization, and is also able to interact with its environment and with other agents".

We also require our agent to possess the following characteristics: *autonomous* (able to function by itself without direct user intervention), *pro-active* (capable of adjusting its behaviour on its own in response to changing stimuli), *reactive* (capable of reacting to external stimuli), and *social* (capable of negotiating with other agents)

Agents can be characterized as "weak" or "strong" agents (Wooldridge and Jennings, 1995). "Weak" agents possess the characteristics described above, whereas "strong" agents possess the following additional characteristics: *mentalistic notions* (they have Beliefs, Desires & Intentions [BDI]), *rationality* (capable of reasoning about their actions and perform actions in line with their BDI), and *learning* (capable of learning from their actions and the external environment). In this paper, we will restrict our attention primarily to weak agents.

Another (7-axis) characterization of agents can be found in (Griss 2000): *adaptability* (the degree to which an agent's behavior can be changed), *autonomy* (the degree to which agents are able to pursue their goals independently), *collaboration* (the degree to which agents work cooperatively), *intelligence* (the degree to which agents display knowledge and understanding), *mobility* (the ability for an agent to move from place to place), *persistence* (the degree to which agents retain knowledge and state over periods of time), *personality/sociability* (the degree to which agents exhibit a pleasant style – this also relates to customization, perhaps by learning) If we map this characterization to our definition of ("weak") agents, we see that that the following axes are relevant for us: autonomy, collaboration, persistence, and, to a certain extent, adaptability.

The Foundation for Intelligent and Physical Agents (FIPA) is one of the standards bodies working towards developing standards for agent modeling and interaction (see **http://www.fipa.org/**). FIPA has developed a reference model, consisting of the following components:

- *Agent Management System (AMS)* - controls creation, deletion, suspension, resumption, authentication and migration of agents. Also provides local "white pages" service (roaming and locating) of agents
- *Agent Communication Channel (ACC)* - routes messages between local and remote FIPA agents, realizing messages using FIPA's Agent Communication Language (ACL).
- *Directory Facilitator (DF)* - provides "Yellow Pages" service for FIPA agents; they register some agent capabilities so an appropriate task-specific agent to handle the task can be found.
- *Agent Platform (AP*) - provides communication and naming infrastructure, using the Internal Platform Message Transport.

Interesting linkages between workflow and agents have been described in (Griss, 2000):

- *Agents can collaborate to perform a workflow,* e.g., telecom provisioning, service provisioning, scheduling, …
- *Agents can be used to make workflow more intelligent,* e.g., by adding negotiation, reasoning or decision points
- *Workflow can be used to choreograph a set of agents,* e.g., application management, …

- *Workflow can be used to coordinate interaction between people and agents, having agents delegate to people or other agents*, e.g., telecom management system alerting a human operator, or assigning a repair or provisioning engineer

We recognize that the first two linkages represent *agent-enhanced workflow* (i.e., using agents to enhance workflow systems - also known as "macro workflow") and the last two represent *agent conversations* (i.e., using "choreographing workflow processes" to model agent interactions and conversations in multi-agent systems - also known as "micro workflow"). Micro workflows can be modeled as single tasks within macro workflows. We have achieved some preliminary results in expanding on this idea in (Narendra 2002, Narendra 2003).

Part I: Lifecycle-Based Multi-Agent Simulation

Introduction to Software Process Modeling

From the section above, we see that multi-agent systems can be modeled as agent-oriented workflow systems, using macro- and micro-workflow concepts. This naturally raises the question of whether workflow and process modeling techniques can be used for modeling and simulating their behavior. To that end, we propose using Software Process Modeling (SPM) techniques. There are several reasons for this: firstly, both software process systems on the one hand, and multi-agent systems on the other, use similar process-based ideas; secondly, both the techniques are lifecycle-based, i.e., a definite start-execute-end cycle can be noticed; thirdly, SPM is a well-researched area with significant technical results which can be leveraged for modeling and simulating multi-agent systems.

Interest in the SPM area has grown ever since the publication of Leon Osterweil's landmark paper titled "Software Processes are Software Too" (Osterweil, 1987). Osterweil's basic idea was that software processes are also software artifacts, and can be modeled like any other artifact, in computer-representable form. This idea has enabled SPM to become a fertile area for research. However, SPM has not really had much success within the software industry itself, largely due to the dynamic and semi-chaotic nature of software development. There was also a perception among most software project managers that the value addition from SPM would not be sufficient to justify investments in SPM research or tools.

From the point of view of multi-agent system simulation, however, we can leverage off the extensive research work in SPM for modeling and simulating agent interactions as processes and workflows, especially since managing multi-agent systems should be a better value proposition for e-commerce system managers.

The ATRIUM Approach to Software Process Modeling and Enactment

One of the most significant research contributions to SPM to date, has been the ATRIUM project (**http://www.usc.edu/ept/ATRIUM/**). The key idea behind ATRIUM, is that SPM has to follow a lifecycle-based approach (Scacchi and Mi, 1997). In other words, SPM should be done as per the steps in the lifecycle listed in the Table below. The Table also shows how each of the process steps can be modeled and simulated in a multi-agent simulation. Hence it describes how SPM can be effectively used for modeling and simulating multi-agent systems. It also shows that a life-cycle based approach is needed for effectively modeling and simulating multi-agent systems.

Current Approaches to Multi-Agent System Simulation

Although the field of multi-agent system simulation is vast and growing (see, for example, **http://jasss.soc.surrey.ac.uk/JASSS.html**), we feel that there are two significant research contributions that are most relevant for our purposes.

Table 1. ATRIUM Lifecycle

Process Step in Lifecycle	*Description*	*Mapping to Multi-Agent Simulation*
Metamodeling	Define process models and process instances from the models	As in (Griss, 2000), defining ontologies for agent interactions and also workflow and multi-agent conversation models
Modeling	Capturing and representing process instances formally and representing them in computer-understandable form	Capturing and representing the choreographing workflow models/instances and agent conversations, in computer-understandable form
Analysis	Evaluating static and dynamic properties of a process model, such as consistency, completeness, internal correctness, traceability as well as other semantic checks	Defining and modeling static and dynamic (including semantic) properties of agent interactions/conversations
Simulation	Symbolically executing process models in order to observe their behavior	Symbolically executing the workflow models and agent conversations in order to observe their behavior
Redesign	Redesigning process models and process instances based on results of simulation	Redesigning workflows and agent conversations based on results of simulation
Visualization	Graphical visualizations of process models and instances	Graphical visualizations of workflow execution and agent conversations
Prototyping, walkthrough and performance support	Incremental enactment of process instances in order to perform a thorough evaluation	Incremental enactment of workflows and agent conversations for evaluation purposes
Administration	User monitoring of the simulation system	As per (Griss and Letsinger, 2000), user monitoring of the simulation system
Integration	Simulating the integration of external systems and repositories	Simulation of external tools/services into the system, and how they would integrate with each other
Monitoring, recording and auditing	Collecting and measuring process enactment data needed to improve subsequent process enactment iterations	Collecting and measuring execution data
History capture and replay	Recording the enactment history and graphically simulating the re-enactment of a process, in order to analyze the enacted process in greater detail	Recording the enactment history of the transactions, in order to analyze them either wholly or in part
Articulation	Diagnosing, repairing and rescheduling actual or simulated process enactments that have unexpectedly broken down	Diagnosing, repairing and rescheduling transactions that have broken down
Evolution	Incrementally adapting or reengineering process models to more effectively meet emerging user requirements	Using performance data to incrementally and iteratively enhance and restructure workflow definitions and agent conversation models
Process asset management	Organizing and managing the collection of meta-models, models and instances of processes, products, tools, documents and organizational structures/roles for engineering, redesign and reuse activities	Organizing and managing the collection of workflow definitions, agent conversations, and instances of executions.

TAEMS from University of Massachusetts, Amherst

In this approach (Horling, et. al., 2000), the behavior of each agent is controlled by a simple simulation script. Each script is composed of three parts:

- The script - this checks the Assertions and realizes the Reactions when the Assertions hold
- One or more Assertions - these are basically conditions that must hold if the Reactions have to be executed
- One or more Reactions - these are executed provided the Assertions hold

An example script is:

```
Script
        AndScript
Assertions
        Time > 5
        ExecuteEventActiveMethod-4
Reactions
        QuitSim
```

This is an AND script, which means that all the Assertions must be ANDed - in other words, all Assertions must be true for the Reactions to be executed. Hence in the above example, if time is greater than 5 and if the currently active method is method-4, then the QuitSim reaction will be executed. In other words, the simulation will end.

Similarly, OR scripts and EXPRESSION scripts can be defined. In this manner, a multi-agent system can be simulated by defining such agent behaviors in a modular and extensible way. When the agents are put together, it is possible to synthesize highly interesting behaviors for the multi-agent system from the simple behaviors of an agent, where each behavior is modeled as a simulation script like the one described above.

SWARM from Santa Fe Institute

The SWARM (Minar, et. al., 2002) approach provides a different way of simulating multi-agent systems. Here, the basic unit of simulation is the swarm, a collection of agents executing a schedule of actions. SWARM supports hierarchical modeling approaches whereby agents can be composed of swarms of other agents in nested structures. However, SWARM does not include the notion of lifecycle-based simulation of the workflow processes executed by agents.

Our Lifecycle-Based Simulation Approach

Briefly, our proposed approach for modeling and simulating multi-agent systems is as follows:

- *Hierarchical Modeling*: Workflows can be represented hierarchically - see the subsection below - as instantiations of a particular workflow definition.
- *Simulation Scripts*: The simulation script idea from TAEMS can be used to define rules for the execution of each activity in the workflows and agent conversations.
- *Goals and Performance Measures*: We can define goals for the multi-agent system that we are simulating. Based on these goals, we can derive performance metrics that tell us the extent to which the goals are being met.
- *Life-Cycle Based Modeling*: The life-cycle approach from ATRIUM can be used to model and create the workflows and agent conversations. This can then be viewed at a lower level of abstraction using the Modeling and Visualization features described in earlier.

Hierarchical Modeling

We can model workflows hierarchically via the specialization and decomposition approach as specified in the Process Handbook project (Malone, et. al. 1999). In this approach, all processes in any organization are represented in a hierarchical manner, with processes lower down in the hierarchy being specializations of the root process. Each process is, in turn, decomposed into sub-processes recursively until the lowest level activity is represented. An example of workflows for buying on the Internet (it could be either B2B or B2C), depicting specialization and decomposition, is given in Figure 1 below.

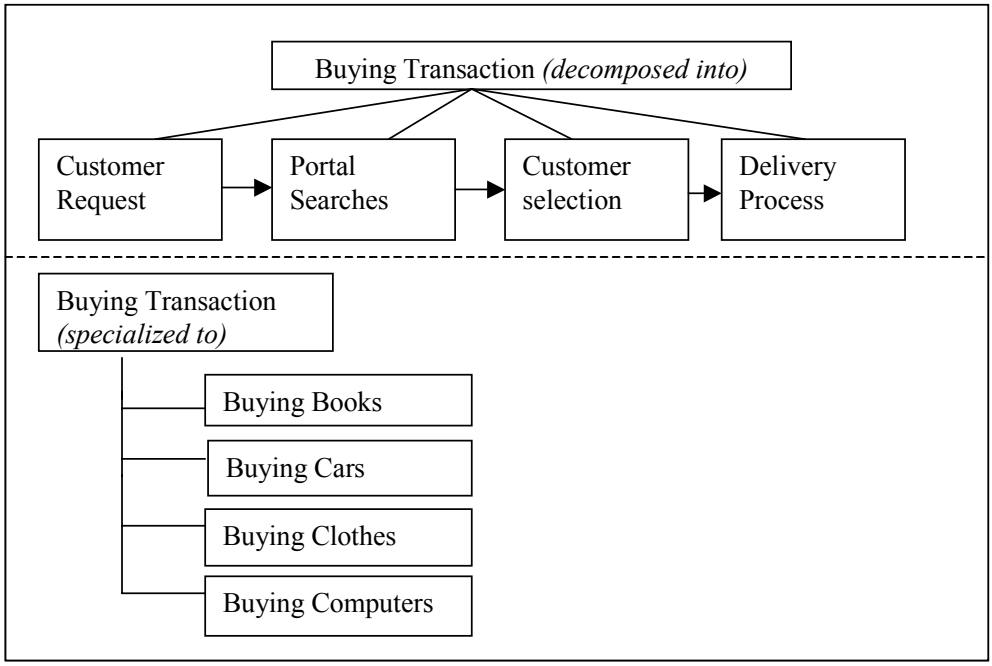

Figure 1. Hierarchical Workflow Modeling

Simulation Scripts

We can use simulation scripts, as described in earlier, to model the preconditions and postconditions of the different tasks of the workflows.

Apart from the fact that we can model and simulate various types of behaviors of any process model, the main advantage of the scripting mechanism, is that it is modular and extensible, thus allowing users to develop new scripts if the existing ones do not meet their needs. The scripts are also readable, hence making them easy to understand and debug.

Goals and Performance Measures

In (Machiraju, et. al., 2000), some requirements for managing e-service environments are listed, along with performance measures for them. We can use the results from there for setting goals for our multi-agent system, and determining the appropriate metrics to be collected for measuring the performance of the system against the goals.

Some common metrics that can be collected from a multi-agent system, are: performance, reliability, transaction cost/speed, metrics related to contracts among the agents. Indeed, it is interesting to note that in the software process community, the notion of deriving metrics from business goals is a well-established craft. The methodology followed, called Goal-Question-Metric (also known as GQM; see Park, et. al. 1996), typically involves deriving the appropriate metrics from business goals by posing questions related to the goals. For example, if one of the business goals is "Availability of >99%", then one appropriate question

to ask, is "How long does the system remain available to clients?" The derived metrics from that question, would be "Percentage of availability of the service over a period of time" and "Frequency of service unavailability".

Life-Cycle Based Modeling and Simulation

One of the most important benefits of SPM, is its ability to model and simulate software processes across the life-cycle. It is for this reason that we have chosen the ATRIUM approach, since it is one of the few SPM approaches in the literature that enables us to model software processes across the life-cycle. Following this approach, we can make use of the Table 1 with the following possibilities:

- At the level of simulation scripts - we can simulate the behavior of individual eAgents (or a small collection of eAgents) at different hierarchical levels
- At the level of individual sub-systems of eAgents - we can simulate global behavior at arbitrarily higher abstraction levels
- We can also simulate the system - at any level of abstraction - for different loads and observe the performance of the system as per the metrics defined and collected as in the previous subsection.

The "Butterfly Effect"

Small perturbations in any part of a dynamical system are capable of causing major changes in the behavior of the system as a whole, i.e., the "Butterfly Effect" (see **http://www.cmp.caltech.edu/~mcc/chaos_new/Lorenz.html**). Since this phenomenon is quite common in real-world systems, its behavior should be investigated in multi-agent systems by adding a bit of "randomness" to some of the simulation scripts, e.g., by ensuring that the behavior of certain simulation scripts is controlled by appropriate random number generators, and measuring then uncertainty generated thereof.

Issues to be Addressed and Detailed Simulation Approach

The following issues arise when trying to simulate multi-agent systems:

1. *What types of simulation to run*
   - What are the different scenarios that need to be simulated?
   - Can these scenarios be arranged in any particular order, from the simplest to the most complex, so that we can determine some structured way of calibrating multi-agent systems?
   - *Quality Attributes:* we need to determine the quality attributes applicable for multi-agent systems, as per [Park, et. al. 1996; Machiraju, et. a., 2000]. We can also leverage the work done in (Woods and Barbacci, 1999) on multi-agent system architecture evaluation, which has identified the following quality attributes as relevant: performance, security, adaptability to environment changes and fault tolerance.

2. *How to incorporate "random" or "unexpected" behavior:=*
   - Determining the different ways that exceptions and random behavior can be introduced into our simulation
   - Programming this into the simulation scripts
   - Observing the behavior of the system and adjusting the parameters accordingly

Hence our simulation can proceed along the following lines:

- Develop the multi-agent architecture that needs to be simulated
- Develop usage scenarios based on how the architecture will be used in the "real world"
- Based on the usage scenarios, determine the workflow processes that need to be executed to conform to the scenarios. Use the ideas from (Minar, et. al., 2002) for hierarchical modeling.
- Specify the quality attributes of the architecture that need to be observed and analyzed during the simulation. Use the (Park, et. al. 1996) and (Woods and Barbacci, 1999) approaches to determine the appropriate performance measures that need to be collected for evaluating the simulation

- Based on the usage scenarios, model the expected behaviors of the components using the simulation scripts from (Horling, et. al., 2000).
- Take care to model the exceptions and random behavior into the simulation scripts. Also, be sure to watch out for manifestations of the "Butterfly Effect", in case of random behavior
- Based on the analysis of the measurements collected, appropriate conclusions about the multi-agent system architecture can be drawn, and the appropriate decisions taken

Part II: The eAgents Reference Architecture

The eAgents Approach and Reference Architecture

The eAgent

In order to effectively simulate multi-agent systems, we need to have a good representation of what the internals of an agent's architecture would be. Hence there is a need to develop such a generic reference architecture.

Our proposed reference architecture, the eAgents Architecture, is given in Figure 2 below. It consists of a collection of what we call eAgents, i.e., agents augmented with certain prespecified features and properties. Each eAgent has the following components:

- A *Knowledge Database* - it stores information relating to the workflow processes executed by the eAgent, the outcomes of the workflow execution, and the goals that led to the creation and execution of the workflow processes in the first place. It also stores information on the products and services supplied to clients and other eAgents as part of executing the workflow processes. This will perform the Process Asset Management function described in Table 1.

- A facility for *Tools/Services Integration* - with this facility, the eAgent can integrate with other external tools and services that are not part of the Architecture. Some examples could be logging, authentication, registry, etc. This will perform the Integration function described in the Table 1.

- A *Workflow Modeling and Automation* component - this is the heart of the eAgent, and basically defines how the eAgent will interact with the others in the multi-agent system, using the macro- and micro-workflows as described in the second section.

- A *Client Interface* for interacting with external agencies, which can be either users (human or automated) or other eAgents.

Hence each eAgent is a self-contained entity with its own Knowledge Base, feature for integrating with external tools/services, and Workflow component.

Such a representation makes it possible to model multi-agent systems in a modular and decentralized fashion.

The Agent Coordinator

Sometimes a central Agent Coordinator is needed for managing a collection of interacting eAgents. As shown in Figure 2, the Agent Coordinator consists of three components:

- A *Centralized Database* - this stores information about all the eAgents in the system, along with high-level state information on the overall state of the system

- A *Mediator* - this component assists the Agent Coordinator and the requesting eAgent in the task of dynamic brokering

- An *Interface Component* - this component is responsible for communicating with, and coordinating, all the Workflow Modeling and Automation components in the eAgents. This component will also need to create and deploy the "choreographing workflow processes" that control the behavior of the agents.

In case one needs to model and simulate self-organizing multi-agent systems (AgentCities Task Force, 2003), the Agent Coordinator would not be needed. Instead, each eAgent would communicate directly with other eAgents via its Client Interface.

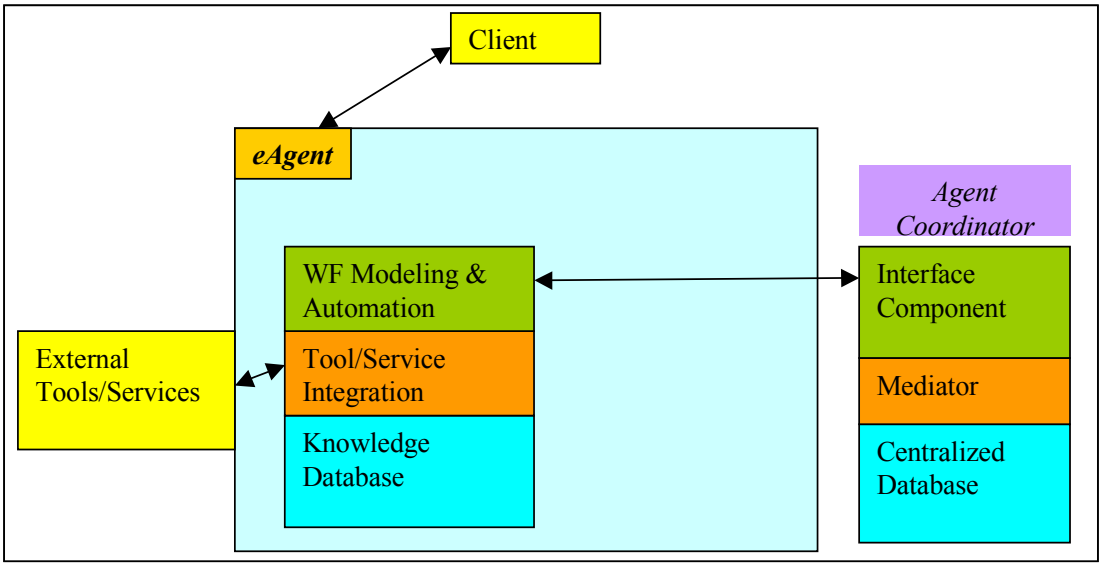

Figure 2. eAgent Architecture

Comparison with FIPA Reference Model

Our eAgent reference architecture is quite compatible with the FIPA reference model described in the second section. Since the FIPA reference model does not specify how the agent will be constituted, we will need to focus on the inter-agent functionality. With respect to our description of the central Agent Coordinator in Figure 2, we see the following mapping to the FIPA reference model:

- The Agent Management System functionality can be performed by the Centralized Database of Figure 2
- The Directory Facilitator functionality can be performed by the Mediator of Figure 2
- The Agent Communication Channel functionality can be performed by the Interface Component of Figure 2
- The Internal Platform Message Transport could be the link by which the eAgent will communicate with its external environment

Related Work

The reason for first modeling a multi-agent system's architecture before simulating the system itself, is that it clarifies our understanding of what we are simulating. This also ensures that we understand the different components, interactions and usage scenarios before we begin to simulate them. To that end, it is always useful to develop the agent architecture before simulation. The eAgent reference architecture proposed in this paper is a reference architecture. That is, it is generic enough to represent the different applications where collections of agents can be deployed. Hence we have proposed the eAgent reference architecture as a template that can be used to model and simulate different types of multi-agent systems. We have also shown that it is compatible with the FIPA reference model.

There have typically been two approaches to agent-based simulation:

- *Analytical* - this involves developing mathematical models/equations to represent agent behavior [Sandholm 1998; Mullen and Wellman, 1996], followed by experimental evaluation upon computer implementation of the equations. Some example applications are, auctions, negotiations and social phenomema (see **http://jasss.soc.surrey.ac.uk/JASSS.html** for extensive literature survey, too numerous to describe here in detail).

- *Experimental* - this approach is used when mathematical modeling are not feasible, e.g., for modeling derivative trading in financial markets (Streltchenko, et. al., 2001). In this case, software programs are written for simulating agent behavior, and the experimental results are observed.

In contrast to the above two cases, our approach is lifecycle-based, since multi-agent systems are typically workflow based. Hence our simulation approach can leverage off a well-researched SPM methodology such as ATRIUM (Scacchi and Mi, 1997). To the best of our knowledge, this is the first time that SPM methodologies have been proposed for multi-agent simulation.

Conclusions and Future Work

In this paper, we have discussed the need for modeling and simulating multi-agent systems before they can be built and deployed, and we have proposed a lifecycle-based simulation approach for multi-agent systems based on Software Process Modeling research. In this connection, we have also presented a generic reference agent architecture called eAgents, which can be used as a building block for simulation. We have also shown how ideas from Software Process Modeling (SPM) can be used for life-cycle based simulation of multi-agent systems. We have also suggested that randomness and the "Butterfly Effect" are bound to exist in real-world multi-agent systems in the future, hence we should also incorporate randomness into the simulation.

We have only defined what multi-agent system simulation should look like, and the experimental work still remains to be done. We need to implement our simulation system and test it out on several real-life examples. We should then use the results from the simulations to investigate quality attributes of multi-agent systems, as described in the sixth section. /For this, we could also leverage Petri-net based techniques such as those described in (Merseguer, et.al. 2000).

As for the eAgent reference architecture itself, we would first need to develop detailed models of interaction between the components of the eAgent, and also between the eAgent and the Agent Coordinator. We will also need to determine the distinction between "macro-workflow" and "micro-workflow", and how they impinge on each other. Some work in this direction is available from (Narendra, 2002; Narendra, 2003), where we have extended the Agent Coordinator idea to that of an Agent Society, where agents can meet, form collaborations, and execute workflows in order to meet common business goals.

Acknowledgments

The author wishes to thank his manager, and the SES Director, for supporting his work. Thanks are also due to Martin Griss, Reed Letsinger, Indradeb P. Pal, Arindam Banerji and Chris Clangdon for their useful feedback. Special thanks are also due to the anonymous referees, whose feedback improved the quality of the paper.

References

AgentCities Task Force, "Self-Organizing Applications: A Survey," available from **http://cui.unige.ch/~dimarzo/eoawg/survey.pdf**, February 2003.

Griss M L, "My Agent will Call Your Agent - But Will It Respond?," Software Development Magazine, Feb 2000; also available from **http://www.hpl.hp.com/techreports/1999/HPL-1999-159.pdf**.

Griss, M.L. and R. Letsinger, "Games at Work: Agent-Mediated E-Commerce Simulation," HP Labs Technical Report HPL-2000-52, available from **http://www.hpl.hp.com/techreports/2000/HPL-2000-52.pdf**.

Horling B, Victor L, Vincent R, "Multi-Agent System Simulation Framework," In 16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation, EPFL, Lausanne, Switzerland, August, 2000; also available from **http://mas.cs.umass.edu/~vincent/papers/IMACS-2000/imacs-2000.ps**.

Machiraju V, Dekhil M, Griss M, and Holland J, "E-Services Management Requirements," HP Labs Internal Technical Report HPL-2000-60, available from **http://www.hpl.hp.com/techreports/2000/HPL-2000-60.pdf**.

Malone T W, Crowston K, Lee J, Pentland B, Dellarocas C, Wyner G, Quimby J, Osborn C S, Bernstein A, "Tools for inventing organizations: Toward a handbook of organizational processes," Management Science, Vol. 45, No. 3, March 1999; also available from **http://ccs.mit.edu/21c/mgtsci/**.

Merseguer J, Campos J, and Mena E, "Performance Evaluation for the Design of Agent-Based Systems: A Petri Net Approach," Proceedings of the Workshop on Software Engineering and Petri Nets, within the 21st International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, June, 2000 (to appear); also available from **http://www.cps.unizar.es/deps/DIIS/CRPetri/papers/jcampos/00_MCM_SEPN.pdf**.

Minar N, Burkhardt R, Langton C, and Askenazi M, "The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations," available from **http://www.santafe.edu/projects/swarm/**.

Mullen T, and Wellman M P, "Some issues in the design of market-oriented agents," In M. Wooldridge, J. Mueller, and M. Tambe (eds.), *Intelligent Agents II: Agent Theories, Architectures, and Languages,* Springer-Verlag, 1996; also available from **http://ai.eecs.umich.edu/people/mullen/atal-revised.ps.Z**.

Narendra N C, "AdaptAgent: Integrating Adaptive Workflows and Multi-Agent Conversations for E-Commerce," submitted to South African Computer Journal, 2002; preliminary version available from **http://www.ipipan.waw.pl/mas/sdc-wg/Docs/N.C.Narendra-adapt-agentB2B-ijec-modified.pdf**.

Narendra N C, "Design considerations for incorporating Flexible Workflow and Multi-Agent Interactions in Agent Societies," Journal of Association for Information Systems, 2003, available from **http://jais.isworld.org/articles/default.asp?vol=3&art=4**.

Osterweil L J, "Software Processes are Software Too," Proceedings of the Ninth International Conference on Software Engineering, pp. 2-13, Mar. 30- Apr.2, 1987, Monterey, CA.

Park R E, Goethart W B, and Florac W A, "Goal-Driven Software Measurement - A Guidebook," SEI Technical Report; available from **http://www.sei.cmu.edu/publications/documents/96.reports/96.hb.002.html**.

Sandholm T, "Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation," Autonomous Agents and Multi-Agent Systems, 3(1), 73-96, Special Issue on Best of ICMAS-98, invited submission, 1998.

Scacchi W, and Mi P, "Process Life Cycle Engineering: A Knowledge-Based Approach and Environment," Intelligent Systems in Accounting, Finance and Management, Vol 6: 83-107 (1997); also available from **http://cwis.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html**.

Streltchenko O, Narendra N C, and Yesha Y, "A Reference Architecture for Multi-Agent Simulation of Derivatives Markets," Proceedings of CIMA 2001, Bangor, Wales, June 2001; available from **http://www.cs.umbc.edu/~streltch/download-files/ACFM2001.ps**.

Woods S G, and Barbacci M R, "Architectural Evaluation of Collaborative Agent-Based Systems," SEI Technical Report CMU/SEI-99-TR-025; available from **http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr025.pdf**.

Wooldridge M, and Jennings N R, "Intelligent Agents: Theory and Practice," In Knowledge Engineering Review 10(2), 1995; also available from **http://www.csc.liv.ac.uk/~mjw/pubs/ker95.ps.gz**.