	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
UNIVERSITY OF CALGARY	Session: Fall, 2003	Department: Electrical and Computer Engineering
		Document Type: Tutorial Report

SENG 609.22 Agent Based Software Engineering

Course Instructor: Dr. Behrouz H. Far

Tutorial Report

Simulation for Agent-Based Software Engineering

Zhizhong Li

zhizli@ucalgary.ca

Simulation for Agent-Based Software Engineering

Zhizhong Li

Abstract

Agent-based systems are one of the most vibrant and important areas of research and development to have emerged in information technology in the 1990s. Simulation is an important category of applications of multi-agent systems in natural and artificial societies. In this tutorial, first I introduce the characters of agent-based simulation models; then discussed the areas of agent-based simulation applications; after that talk about the challenges existed in agent-based simulation and models as well as solutions; at the end of this tutorial apply the above knowledge into reality and study a concrete agent-based simulation model in detail.

1. Introduction of Agent-based simulation

Agents can be interpreted as software systems that are aimed at working autonomously in dynamic and uncertain environments [6]. The interrelations between agents and simulation are manifold [2]. Software agents are used to develop "state of the art" simulation systems on the one hand. On the other hand simulation provides a means for systematically analyzing the behavior of agents in dynamic virtual environments.

Multi-agent systems offer strong models for representing real-world environments with an appropriate degree of complexity and dynamism. These simulation models are characterized by the intersection of three scientific fields, namely agent-based computing, the social sciences, and computer simulation. The social sciences study interaction among social entities and include social psychology, management, policy and some areas of biology. Computer simulation concerns techniques for simulating phenomena on a computer, such as discrete event, object-oriented, and equation-based simulation. Interesting and relevant work occurs in related intersection areas, including:

- Social sciences and agent based computing (social aspects of agent systems);
- Computer simulation and agent-based computing (multi agent based simulation);
- Social sciences and computer simulation (social simulation).

As is evident, simulation covers a range of phenomena from the most applied (e.g., manufacturing processes, traffic systems, information and control systems) to the most abstract (e.g., social dimensions to belief, trust, duty and right).

2. Agent-based simulation Application Areas

The applications of agent-based simulation are of both natural and artificial societies. These applications include education and training systems, scenario exploration and policy systems, and entertainment systems.

2.1 Education and Training

Multi-agent systems provide a natural basis for training of decision-makers in complex decision-making domains. For example, defence simulations using multi-agent systems, can enable military planners, strategic defence staff and even operational staff to gain experience of complex military operations through simulations and war games. These simulated experiences are obtained instead of, or in addition to, experiences gained in actual military operations. Similarly, decision-makers in other complex and dynamic environments can gain valuable experience through exercises which simulate their real-world domain using multi-agent systems. Applications include marketplaces subject to rapid change, such as telecommunications markets undergoing deregulation, and markets for fast-moving consumer goods, such as breakfast cereals, where consumer tastes and competitor activities can lead to market turbulence. In these applications, as for those in defence, multi-agent systems may simulate over a few hours the dynamics of an actual market which could occur over several years, and so give trainee decision-makers rapid exposure to many diverse experiences. In addition, as the military example reveals, the decision-maker is allowed to learn through his or her mistakes, without these having real-world consequences.

2.2 Scenario Exploration

Social simulation is somewhat unusual in that it does not require many of the challenges listed earlier to be addressed in order for it to succeed in the timescales considered in this report. Since simulations are by their nature closed (even though they may model open systems) they are almost immediately enabled. However, there are many open issues to be resolved before agent-based simulation models can be applied more widely to public policy domains. For example, there is as yet no general understanding of what constitutes good performance by a multi-agent system, except perhaps in some domains. There is no guarantee, for example, that an agent society in which different species of agents co-evolve in the course of their interactions with one another will progress in any sense; later generations of a species may be less fit than earlier generations of that same species when pitted against earlier generations of their competitor species.

In addition, many applications of agent systems to public or social policy domains involve the development of alternative scenarios, the outcomes of which are used to guide human decision-makers. But at present there is no formal theory of scenarios and scenario analysis, which would tell us how to construct scenarios, how many scenarios to construct and how to reason between and across their outcomes. Developing formal theories of scenarios and rigorous methods of performance assessment for multi-agent systems will require collaboration between computer scientists, philosophers and decision theorists, as well as the domain experts to which these systems are applied.

2.3 Entertainment

Multi-agent systems as social simulations are also of increasing importance in entertainment applications. These applications range from single-(human)-player computer games to multi-player games, where the other players may be both humans and agents. The popularity of social simulation games, such as Maxis' SimCity, for example, where human players construct artificial societies, show the potential for multi-agent simulation applications. Potential applications also exist in other interactive media, such as interactive movies, television and even books, where viewers and readers may have their own avatar participate in the story and may interact with fictional characters directly.

3. Challenges and Route of Establishing Agent-based Modeling and Simulation

The challenges of establishing modeling and simulation within the software development cycle are twofold. Technical challenges rise naturally on the path. Overcoming those is necessary but not sufficient to reach the final goal. Interdisciplinary work is not a purely technical endeavor, but closely related to the perception and acceptance of one discipline by the other.

3.1 Flexibility and “Easy to Use”

Test environments for agents shall support a testing of agents in the small and in the large [10]. Testing in the small, e.g. the TILEWORD scenario (Figure 1), is used to reveal basic limits, problems, and strengths of the proposed agent architecture, which again might re-surface in real world settings of a specific type. A number of test scenarios for testing in the small have been developed. Early test-beds supported only a specific type of test scenarios. Many of them have been inspired by the perception of agents as robots living in a two dimensional space through which agents move continuously or which, even more often, has been discretized into a grid world. In order to support test scenarios of a different type, more general simulation tools are required that support different application domains likewise. Providing different component libraries for different domains is one step towards the solution. A support of continuous as well as discrete simulation and an easy extensibility at the simulation and experimental level ask for a layered and component-based approach, e.g. [4].

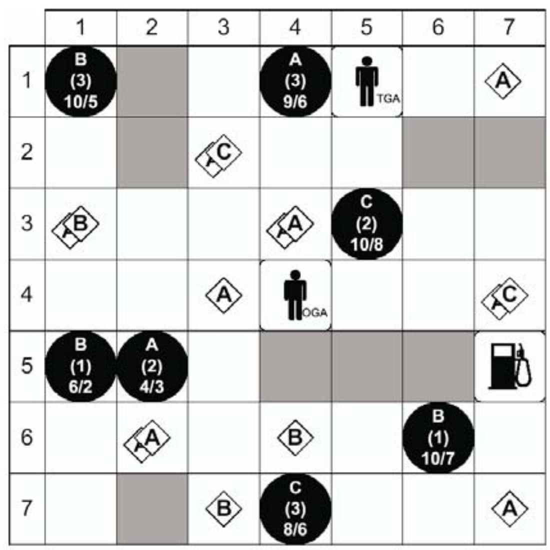


Figure 1. The TILEWORLD scenario in JAMES. Diamonds tiles, black circles denote holes which are attributed with a type, a depth, and with a score to be gained by filling it with the correct and incorrect type of tiles (6/2). The grey grids are obstacles

We envision modeling and simulation methods and tools which support a flexible and comfortable composition of test environments for multi-agent systems by re-using models of different languages and by an interoperation with other simulation systems that takes into consideration the semantics of the exchanged information.

3.2 Dynamic Structure

Agents are characterized by interaction pattern or spheres of influence that vary over time. The ability to adapt their own interaction, composition and behavior pattern challenges the expressiveness of formalisms, and efficiency of tools [13] likewise. To “move the modeling and simulation process towards science” [18, p.27] a theoretical foundation is required. However, whereas the number of simulation systems that allow the creation and deletion of model components and interactions has been steadily increasing, work on corresponding formalisms has not been keeping pace. Among the formal approaches to discrete event simulation the DEVS formalism has been analyzed more thoroughly with respect to possibilities and implications of integrating and expressing variable structures than others.

3.3 Efficient Execution

Many test beds for multi-agent systems do not execute their models concurrently. They simply maintain the illusion of simultaneity on a single machine. If only a single deliberative agent is tested in a dynamic environment or the coordination strategies of a moderate number of reactive agents are tested, there is no need for a distributed, parallel execution of agents. However, to efficiently test multiple deliberative agents, each of which consumes significant storage and computation resources, a concurrent, distributed simulation is desirable.

Often a pure discrete simulation does not suffice, and continuous and discrete simulation strategies are combined, e.g. to simulate robots. As agents move continuously through space so do their spheres of interest [19] and thus the regions they can and want to access. Strategies for a dynamic load balancing and for a dynamic re-distribution of logical processes have to be developed to adapt the execution to the current needs.

Thus, methods have to be developed that support the efficient execution of combined, continuous, distributed models that exhibit dynamical interaction and composition structures. As no single strategy will yield the optimal solution independently of the concrete test scenario, flexible simulation tools are required that allow to replace and refine execution methods on demand.

3.4 Models and the “Real Thing”

The implementation and application of dynamic test scenarios for multi-agent systems require considerable modeling efforts. In most cases, the virtual environment, the interaction between agent and virtual environment, and the agent have to be modeled. However, different strategies exist to lighten the load of modeling in experimenting with agents.

Typically, the agent is not modeled in its entirety. Already the first simulation systems for agents allowed to plug code fragments, or single modules into the modeled agent skeleton [21]. A further step towards reducing the modeling effort treats agents as external source and drain of events [3]. The simulation is interpreted as a black box with a clearly specified interface. The agent software has to be changed to transform and to redirect requests and messages that are normally sent to its real environment to the simulation system. If the agent is synchronized with the simulation system in simulation time, messages are labeled with a time-stamp.

3.5 Schedule

Work to date in agent development has largely ignored recent developments in simulation methodology, and has instead tended to employ various ad-hoc approaches to simulation. Jennings and Wooldridge conclude, "systematic testing is the least developed area in developing multi-agent systems" [16]. Simulation seems often to be perceived as an area of straightforward techniques and handcrafted solutions rather than of multifaceted scientific research. For this to change a close interaction with agent developers and computer scientists working on software engineering methods for agents has to be installed.

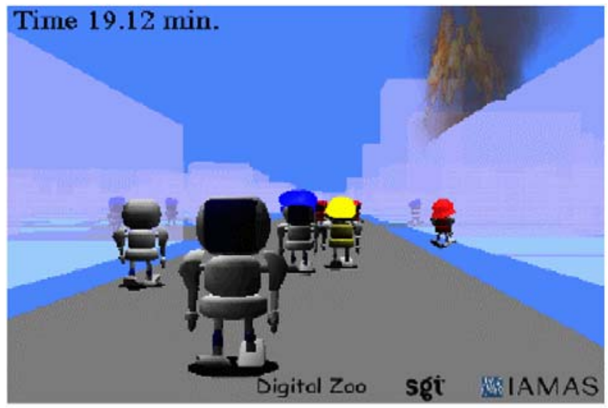


Figure 2. 3D view of the RoboCup rescue simulator

In cooperation with agent developers test scenarios shall be selected. There should be no shortness of possible candidates. One group of agent developers, which is concerned with the definition of suitable test scenarios, is the international research initiative RoboCup. The soccer game is one scenario used to promote research on cooperation between autonomous agents and robots in dynamic multi-agent environments. A simulation league complements the robot league in the many national and international contests each year. Maybe inspired by the virtual island PACIFICA [11] in which the generation and execution of plans for non-combatant evacuation operations have been tested, the RoboCup initiative has defined a new challenge: the RoboCup Rescue challenge. Its simulator of a large scale urban disaster currently embraces four modules, i.e. for simulating collapsing buildings, blockages in roads, the spreading of fire, and the flow of traffic (Figure 2).

4. An Example of Agent-based simulation (simulation of a trading multi-agent system)

In a trading scenario agents interact with each other, selling and buying resources. In order to control the behavior of the trading scenario, the interactions must be coordinated. We present a brief discussion of communication types and coordination models applicable in multi-agent systems. We discuss the advantages of a trading agent model that deals with the trading strategy, concentrating on what to buy or sell. This relieves the agent from the task of coordinating the negotiations and their revoking or acceptances.

4.1 Communication in Multi-agent System

Agents need some way to exchange information. This is done through communication. Here we are only interested in the communication type, which may affect coordination. Communication can be roughly divided in point-to-point and broadcast [5]. There are also communication types with features from these two broad classes, such as local broadcast (as used by social insects) and group communication. Next we present a brief summary of each type of communication.

4.1.1 Point to point

In point to point communication [5], an agent needs the address (such as mail address, IP address, identification number) of its partners in order to engage in communication with them. An agent may change location, an agent's address may have a time limit (the agent advertises its current address periodically), and an obstacle may preclude direct communication (an intermediate network node is down).

Point-to-point communication requires that an agent knows, before hand, what are the capabilities of the receiver. Either the capabilities are fixed and known at starting time, or there is some repository or there is some announcement protocol. This communication type is useful if the entities of the problem being tackled are organized in a hierarchy.

4.1.2 Broadcast

In broadcast communication the agent's address is managed by the communication type. Therefore, agents can enter and leave the multi-agent system without some of the constraints point-to-point has. This in turn gives more flexibility to the agents and removes the burden of address management from the agents.

4.1.3 Local broadcast

Social insects present one of the best examples of multi-agent systems. Pheromenes degrade with time and their concentration throughout space is used as localization aid.

4.1.4 Group Communication

A multi-agent system can be organized in groups or federations. Inside each group or federation, the communication system provides a global view, as updated as possible, of its participants. This type of organization has some advantages over previous communication models. Inside a group, agents communicate using broadcast. If however an agent cannot fulfill its needs, it queries the federation manager that has to find another federation where the agent can meet its needs.

4.2 Coordination Models For Multi-agent System

4.2.1 The Distributed Computing and Parallel Processing View

The coordination problem traces back to parallel program solving. Historically the coordination programming languages or extensions to current languages can be classified as control-driven or data-driven.

4.2.2 Distributed Artificial Intelligence View

In summary, the important approaches to coordination in the Distributed Artificial Intelligence View are:

- Contracting
- Organizational structuring
- Decision-theoretic & Meta-level information exchange
- Multi-agent planning
- Reactive tuple spaces

4.2.3 Coordination Summary

In coordination there are two main groups:

- Low level computing-based, where the algorithm or control flow is the important aspect.
- Information-based or logical-based in formulation. The goals are expressed in some logical form and the agents are equipped with plan solvers and logic reasoning capabilities.

4.3 Trading Framework

This framework allows selling and buying of commodities simulating a simple market. Using a programmable tuple space we take the complexity of managing multiple negotiations from the trading agent, allowing the agent to concentrate on the sell and buy decision process. Changes in the market rules are straight forward. We only need to change or add reactions to the tuple space.

We may have several programmable tuple spaces that represent different markets. They may have different commodities or different rules of interaction. However, we will need to add a code mobility layer, so that trading agents can move between the several markets.

4.3.1 Parameters

We now describe the trading framework parameters. It consists of a set of trading agents *A* that are able to buy and/or sell certain resources *R*. Trading agent *i* trades resource *j* at price ${P^s}_{ij}$ for selling and ${P^b}_{ij}$ for buying. In the beginning, each trading agent has a certain amount of resources, x_{ij}, which is the quantity of resource *j* trading agent *i* has.

Also, each trading agent starts with a certain amount of money (or wealth), w_i, which means wealth in possession of trading agent *i*.

P^b_{ij}	buy price		
P^s_{ij}	sell price		
x_{ij}	starting resources	n	number agents
w_i	starting wealth	m	number resources
g_{ij}	resources goal	i	$1 \ldots n$
t_{ij}	time to achieve	j	$1 \ldots m$

Table 1: Simulation Parameters

The goal of a trading agent is to achieve a certain amount of resources. Trading agent *i* has to achieve g_{ij} units of resource *j*. To have a strictly buying trading agent we need to set up $g_{ij} > x_{ij}\ ,\ \forall j$. In addition, the trading agent has a time constraint to reach its goals. Trading agent *i* has t_{ij} time units to reach resource *j* goal. See table I for the complete simulation parameters.

4.3.2 Trading Agent Model

A trading agent in order to achieve its individual goal must interact with other trading agents. We will, therefore, start to describe the trading agent model by the trading agent's protocol, figure 3. In this protocol a trading agent that needs to trade some resources, issues several ResourceTradePrice messages, one for each resource goal. Each of these messages has several data fields, including resource type and whether the trading agent wants to sell it or buy it. Table 2(a) has the full description of the message. After a while, it receives some TradeProposal messages. This TradeProposal message has other data fields such as the proponent trading agent, the resource it wishes to trade, how much resources it needs, and the resource price, see table 2(b). After deciding what resources it wishes to trade, the trading agent sends an AcknowledgeTrade. This message has the final resource price and the final resource amount. Table 2(b) has the complete data fields.

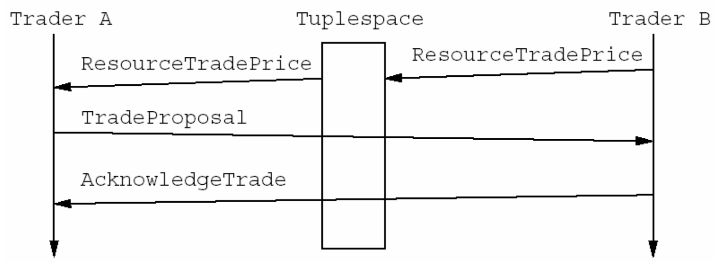


Fig. 3: Trading agents protocol (through arrows means addressed tuples)

`traderNumber`	trader
`resourceNumber`	resource to announce
`trade`	trade intention (buy/sell)
`price`	resource price

Table 2(a) ResourceTradePrice

`fromTrader`	sender agent
`toTrader`	receiver agent
`resourceNumber`	resource to trade
`trade`	trade type (buy/sell)
`price`	resource price
`resourceQuantity`	resource amount

Table 2(b) TradeProposal and AcknowledgeTrade

The current protocol and trading agent model only copes with two negotiations at the same time. Each trading agent has two threads of execution, each one for the two roles of the trading agent protocol. In step CheckAcknowledgeTrade the trading agent has a timeout $timeout_{proponent}$. If this time elapses and there is acknowledge, the trading agent resigns the trade, otherwise it updates its resources.

In the current implementation, trading agents sell for the higher price and buy for the lower price. When there are several ResourceTradePrice (or TradeProposal in the passive thread), the trading agent chooses the most profitable (cheapest when it wishes to buy and expensive when it needs to sell). This kind of decision resembles a reactive approach, since this trading agent model does not account for other possible trading agents' proposals. Therefore, this strategy is greedy and does not adapt to market dynamics.

4.4 Results

For each combination of number of traders and number of resources we performed 10 simulations. All 10 simulations used the same set of initial parameters.

Table 3 shows the percentage of trading agents that achieved their goals in the alloted time. In the simulations with the highest number of resources and with higher number of trading agents (256 and 64 trading agents with 16 resources), many of them did not finish their goals in time, even when the time to achieve the goal was proportional to the number of trading agent times the number of resources. The trading agents with high resource goal will need more time to reach goal since they only perform 2 negotiations at the same time. Also, since the trading strategy is greedy, all trading agents will try to trade with the same trading agent.

traders	resources			
	2	4	8	16
4	100.0%	100.0%	95.0%	75.0%
16	98.7%	98.7%	100.0%	100.0%
64	100.0%	100.0%	60.1%	64.8%
256	99.9%	16.1%	5.7%	0.7%

Table 3

Table 4 has the average time the trading agents took to achieve their goals. These results may indicate some problems with scalability. However, since the trading agent strategy is greedy and does not adapt the resource price, all trading agents will try to trade with the trading agent that has the most profitable offer. Therefore, as the number of traders rises, trading agents with high P^{s}_{ij} or low P^{B}_{ij} will finish their goals later.

traders	resources			
	2	4	8	16
4	0:02	0:02	0:03	0:03
16	0:03	0:04	0:06	0:11
64	0:06	0:12	0:49	10:47
256	1:29	2:48	2:11	3:29

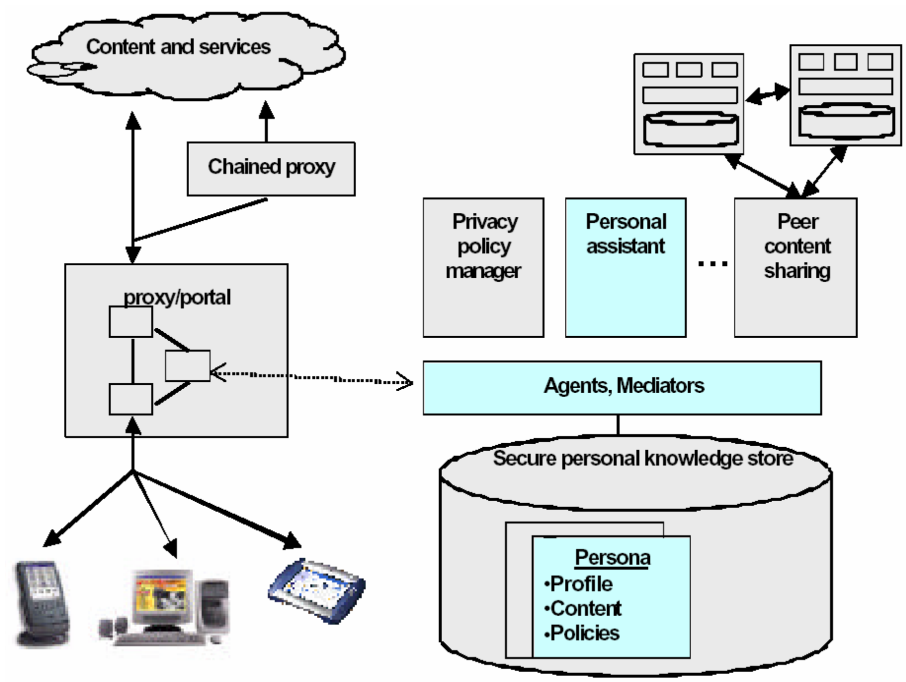
Table 4

5. Conclusion

Agent-based methodology is one of the most important contributions to the field of Software Engineering. It has several benefits compared to existing development approaches. Agent-based simulation and modeling have a great potential for improving the representation of the human dimension in both integrated assessment models and integrated assessment process.

The more software systems are needed that work autonomously in open dynamic environments, the more important a systematic experimenting will become. This tutorial has sought to give a topical overview of recent research and industrial applications of agent-based simulation methodologies. It summarized briefly a number of conceptual issues that should be addressed to improve the application of agent based simulations, as well as it also give a concrete example model and analyze the simulation process.

The goal is to develop modeling and simulation methods based on theoretical foundations that allow a systematic, comfortable, efficient and effective testing of agents and to firmly root simulation within agent-oriented software engineering. In the future to come simulation methods should play an important and accepted role in designing agents.



The user accesses web-services through any existing appliance: PDA, phone, laptop, voice mail, or pager and communicates either with proxies and portals (current static manner), or through agents or mediators as part of a middleware transforming the web-services into an active environment.

Reference:

[1] http://www.agentlink.org/roadmap/contents.html

[2] A.M. Uhrmacher, P.A. Fishwick, and B.P. Zeigler, editors. *Special Issue: Agents and Simulation: Exploiting the Metaphor*, volume 89 of *Proceedings of the IEEE*, 2001.

[3] S.D. Anderson. Simulation of Multiple Time-Pressured Agents. In *Proc. of the Wintersimulation Conference, WSC'97*, Atlanta, 1997.

[4] H. Praehofer, J. Sametinger, and A. Stritzinger. Discrete Event Simulation Using the JavaBeans Component Model. In A.G. Bruzzone, A. Uhrmacher, and E.H. Page, editors, *1999 International Conference onWeb-Based Modeling and Simulation*, volume 31, pages 107--112, 1999.

[5] Andrew S. Tanenbaum. *Distributed Operating Systems*. Prentice Hall International Editions, 1995.

[6] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Resarch and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1):275--306, 1998.

[7] http://www.aaai.org/AITopics/html/agents.html

[8] Ingargiola, Giorgio P. ``Search." http://yoda.cis.temple.edu:8080/UGAIWWW/lectures97/search. 1 Dec. 1998.

[9] www.agentbuilder.com/AgentTools

[10] S. Hanks, M.E. Pollack, and P.R. Cohen. Benchmarks, Test Beds, *Controlled Experimentation and the Design of Agent Architectures*. AAAI, (Winter):17--42, 1993.

[11] Y. Gil, M Hoffman, and A. Tate. Domain Specific Criteria to Direct and Evaluate Planning Systems. Technical Report ISI-93-365, Information Sciences Institute, University of Southern California, Marina delRey, CA, 1994.

[12] http://www.enel.ucalgary.ca/People/far/Lectures/SENG609-22/PDF/tutorials/2001/Agent_Based_Simulation.pdf

[13] A.M. Uhrmacher. Dynamic Structures in Modeling and Simulation - a Reflective Approach. *ACM Transactions on Modeling and Simulation*, to appear.

[14] http://www.ecs.soton.ac.uk/~nrj/download-files/roadmap.pdf

[15] World Wide Web Consortium Website (www.w3.org),

[16] N.R. Jennings and M. Wooldridge. Applications of Intelligent Agents. In N.R Jennings and M. Wooldridge, editors, *Agent Technology : Foundations, Applications, and Markets*. Springer, 1998.

[17] http://www.thesimguy.com/GC/papers/WMC02/G067_UHRMACHER.pdf

[18] E.H. Page. *Simulation Modeling Methodology: Principles and Etiology of Decision Support*. PhD thesis, Virigina Polytechnic Institute and State University, 1994.

[19] B. Logan and G. Theodoropolous. The Distributed Simulation of Multi-Agent Systems. *Proceedings of theIEEE*, 89(2):174--185, 2001.

[20] Agentcities web site http://www.agentcities.org/

[21] T. Montgomery and E. Durfee. Using MICE to Study Intelligent Dynamic Coordination. In *Conference on Tools for Artificial Intelligence*, pages 438--444, Washington, DC, 1990. IEEE.

[22] Berners-Lee, T., R. Fielding, and H. Frystyk. ``Hypertext Transfer Protocol -- HTTP/1.0." *RFC:1945*. http://info.internet.isi.edu:80/in-notes/rfc/files/rfc1945.txt. May 1996.

[23] Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* McGraw-Hill, New York, N.Y., 1997.

[24] Crossley, John. ``IP Network Index." http://ipindex.dragonstar.net. 22 Dec. 1997.

[25] *Agentcities: a Worldwide Open Agent Network*. Steven Willmott, Jonathan Dale, Bernard Burg, Patricia Charlton, Paul O'Brien. AgentLink review, in review.

[26] *System for document telenegotiation (negotiator agents).*Rodrigez J.M and J. Sallantin. COOP'98: 3rd International Conference on the Design of Cooperative Systems, Cannes, France, May 26-29, 1998, pp. 61-66.