ABMS: Agent-based Modeling and Simulation in

Web Service Selection

Jebrin AL-SHARAWNEH
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
jebrin@it.uts.edu.au

Mary-Anne WILLIAMS
Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, Australia
Mary-Anne.Williams@uts.edu.au

Abstract: **Agent-based modeling and simulation (ABMS) is a new approach to modeling systems comprised of autonomous interacting agents. It promises to have an important role in research and education. Some researchers have contended that ABMS "is a third way of doing science". ABMS has been applied to a wide range of research in a varied number of complex domain problems. Social simulation is playing an increasingly important role in today's interconnected society. In this paper we apply agent based modeling and simulation to investigate the impact of Goldbaum's innovative "Follow the Leader" in social networks in web services selection using a recommender system that guides a user to select the best service that matches his requirements and preferences. We test and evaluate several customers' behaviors scenarios using our simulation tool "SSSS: Service Selection Simulation Studio".**

Keywords: Agent based modeling and simulation, ABMS, NetLogo, Service Selection, Social Networks, Follow the Leader.

I. INTRODUCTION

Agent-based Modeling and Simulation (ABMS) is a new approach to modeling systems comprised of autonomous, interacting agents. Computer simulations are based on agent-based modeling of a real (or imagined) system in order to solve a concrete problem. Some researchers presume that ABMS "is a third way of doing science" [1] in addition to traditional deductive and inductive reasoning. ABMS promises to have comprehensive effects on the way businesses use computers to support decision-making and researchers use electronic laboratories to support their research.

ABMS draws its importance from the fact that we live in an increasingly complex world. The systems that we need to analyze and model are becoming more complex in terms of their interdependencies [1]. Now we can compute large-scale micro simulation models that would not have been possible a couple of years ago. The research area of Agent-based Modeling and Simulation continues to produce techniques, tools, and methods. Wide varieties of ABMS applications [2] and simulation tools have been developed in recent years, their primary goal is to assist model building. Tools differ with their design, availability of functions for scientific domain modeling, visualization of results and the support of modeling process.

Computational advances have made possible a growing number of agent-based models across a variety of application domains. Application of ABMS is in various domains such as biology, business, chemistry, commerce, economy, engineering, environment, management, individuals, groups and organization behavior. ABMS draws on these fields for its theoretical foundations, its conceptual worldview and philosophy, and for applicable modeling techniques.

Agent based social systems simulation is a new research strategy that is developing very rapidly. Using ABMS to study a social phenomena is now commonplace. Such phenomena are often studied via simulation modeling. In such simulations, the model consists of a representation of the structure and behavior of a particular real world entity [3]. A simulation consists of running the model under a specific set of circumstances defined by a set of parameters and then analyzing the outcome. The aim of the simulation is to construct a model where behavior matches the real entity in at least a significant aspect. By constructing such a model, social scientists aim to develop conclusions that provide insight into the behavior of real world entities or phenomena; such modeling is often exploratory. Alternatively the simulation may be used to confirm how reliable a predicted behavior is under certain key conditions, which may or may not be under direct control. Simulation results obtained are repeatable; this allows models to be shared among the scientific community allowing further analysis and reuse.

Further more, ABMS can serve as a viable tool to help researchers from many disciplines to recreate and predict the actions of complex phenomena, especially for domains which require a long time to evolve or require exposing real people to dangers, such as disease transmission, corporation management, and military operations. Parunak et al. recently concluded that "agent-based modeling is most appropriate for domains characterized by a high degree of localization and distribution and dominated by discrete decision" [4]. For each problem under investigation researchers should consider a modeling approach, the implementation of the simulator, and how the results are assessed.

Agent-based modeling and Simulation, ABMS, is known by other names such as Agent Based Simulation (ABS) [2], ABM (agent-based modeling), Agent directed simulation ADS [4], Agent - based social systems simulation [5], ABS (agent-based systems), and IBM (individual-based modeling). For specific domains, any combination from the above mentioned

acronyms and a domain, generate a new name, for example: Agent-based social simulation -ABSS [6], Figure-1 shows the intersections of agent-based computing, computer Simulation and the research area domains.

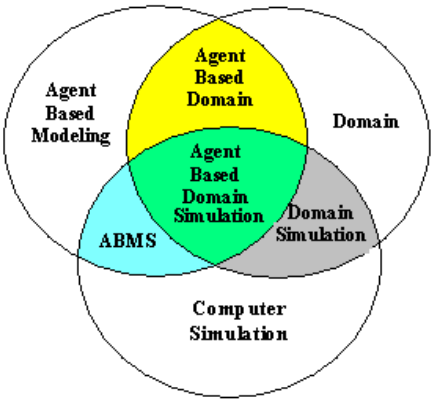

Figure-1: Intersections of agent-based computing, Simulation and the research Domain.

In the following sections, we provide an overview of Agent Based Modeling and Simulation, and corresponding platforms. Additionally NetLogo capabilities are explored as an example of such platforms. Finally, we present our simulation tool: Service Selection Simulation Studio (SSSS).

II. AGENT BASED MODELING

Agent based modeling is the corner stone in the ABMS process. Agents represent entities in a model and agent relationships represent processes of entities interaction. Agents are autonomous computer programs, capable of independent actions in environments that are typically dynamic and unpredictable [4]. Agents are used to generate model behavior in simulation.

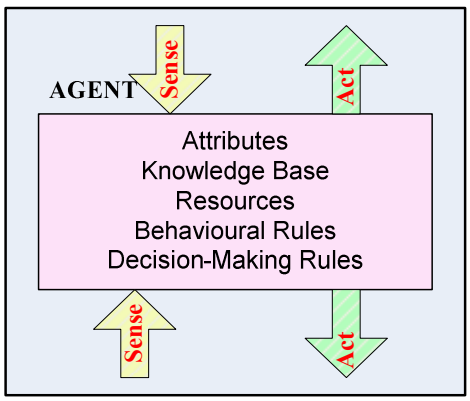

Figure-2: An agent model

A. *What is an agent?*

An agent as depicted in Figure-2 is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives. Agents are: (1) problem solving entities with well-defined boundaries and interfaces; (2) situated (embedded) in a particular environment - they receive inputs related to the state of their environment through sensors and they act on the environment through effectors; (3) designed to fulfill a specific purpose - they have particular objectives (goals) to achieve; (4) autonomous - they have control both over their internal state and over their own behavior; (5) capable of exhibiting flexible problem solving behavior in pursuit of their design objectives - they need to be both reactive (able to respond in a timely fashion to changes that occur in their environment) and proactive (able to opportunistically adopt new goals) [7]. Agents have the ability to learn and adapt their behavior based on experience. This requires some form of memory and knowledge base. An agent may have rules that modify its rules of behavior.

B. *Multi-Agent Systems (MAS): Overview*

The study of Multi-Agent Systems (MAS) focuses on systems where many intelligent agents interact with each other. Agents' interactions can be either cooperative or self-interested. That is, the agents can share a common goal or they can pursue their own goals. In most cases, agents act to achieve objectives either on behalf of individuals or as part of some wider problem solving initiative. Thus, when agents interact, there is typically some organizational context, this context defines the nature of the relationship between agents and influences an agent's behavior.

Agents are often embedded in an environment. Traditionally the actions of agents are executed within some environment to allow for the observation of their outcomes [4]. The individual agents themselves become tied to the environment that surrounds them including other nearby agents. This environment can either be spatial in nature or logic relations between agents. MAS used in modeling human social and organizational behavior and individual decision-making, which reflects the need to represent social interaction, collaboration, group behavior, and the emergence of higher order social structures.

C. *MAS Architectures*

MAS is used to denote a particular arrangement of data structures, algorithms, and control flows, which an agent uses in order to decide what to do. Agent architectures can be characterized by the nature of their decision-making. Example types of agent architectures [8] include:

1. Logical based architectures – in which decision-making is achieved via logical deductions.
2. Reactive architectures – in which decision-making is achieved via simple mapping from perception to action.
3. Belief-Desire-Intention architectures – in which decision-making is viewed as practical reasoning of the type that we perform every day in furtherance of our goals. Beliefs are the information an agent has about its own environment, which may be false; desires are those things that the agent would like to see achieved, and intentions are those things the agent either committed to doing (intending to) or committed to bring about (intended that).

4. Layered architectures – in which decision-making is realized via the interaction of a number of task accomplishing layers.

D. *Agent Execution Cycle*

In a vast majority of cases, agent architectures differ only by the data structures and algorithms they choose to utilize. Figure-3 illustrates agent execution cycle, with details about each of the five major steps listed below [4]:

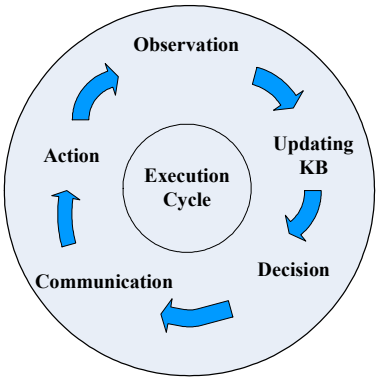

Figure-3: Agent Execution Cycle [4]

Observation. This step collects information on current environmental conditions and maps those conditions to precepts. This step is domain dependent and limited in its scope by its implementation.

Updating KB (Knowledge Base). An agent's knowledge base will be updated under three cases: (1) when the agent observes the environment, it will update the knowledge base by the new percepts; (2) when the agent performs an action, it will update the knowledge base by the effects of the action; and (3) when the agent receives a communication message, it will update the knowledge base by that message. For all cases, the function update must check the entire knowledge base for inconsistencies.

Decision. Here agents make two separate decisions: (1) what action to perform and (2) what message to communicate and to whom.

Communication. In MAS environment, agents cannot force other agent's to perform a specific action or directly alter their knowledge base. However, they can exert influence over other agents through communicative actions, through sending and receiving messages.

Action. The functional nature of an agent's action step is rather intuitive and simple, its purpose is to ensure a successful, coherent and fault proof execution of the optimal action that was recommended by the agent's decision making mechanism.

Hence, Agent modeling concerns with modeling agent interactions. The primary issues of modeling agent interaction are (1) who is connected to who and, (2) the mechanisms governing the nature of the interactions. Agent interaction topologies [1], such as networks, allow an agent's neighborhood to be defined more generally and more accurately describe social agents' interaction patterns

III. AGENT BASED SIMULATION

Simulation is the imitation of some real thing, a state of affairs, or process. The act of simulating something generally entails representing certain key characteristics or behaviors of a selected physical or abstract system. Thomas Schelling is generally credited with developing the first social agent-based simulation in which agents represent people and agent interactions represent a socially relevant process.

The simulation problem is always grounded in empirical data and algorithms, and are utilized to model real systems, where by simulation must also be modeled in a functionally equivalent way to the real world. Each problem description includes [2] the domain studied, the intended end-user, and the purpose of the ABMS application. When modeling a MAS [9], one should specify a number of agents, their types and abilities, the environment in which they operate, the stimuli that can be perceived by the agents, the agents' communication network and the rules that govern the structural mutation of the overall system.

A. *Simulation Environment*

ABMS uses agent technology to develop simulation techniques. As Davidsson et al. point out this makes the modeling approach of ABMS unique and it should be captured by the following aspects [4]:

Simulated Entities: The simulated entities are the foundation components of the studied system. Three different categories of entities are identified: (1) Agents. They include software agents, robot agents or human agents. (2) Environment objects. They include any natural objects in the environment. (3) Organizations. They include any group format of agents.

Agent Types: Four basic agent types can be considered in modeling: (1) Simple reflex agents. These agents select actions on the basis of the current percept, ignoring the rest of the percept history. (2) Model based agents. These agents have their mental states so they can keep track of the past world. (3) Goal based agents. These agents not only know the current state of the environment but also goal information that describes situations that are desirable. (4) Utility based agents. These agents have utility function, which can help them compare different world states and generate high quality behavior.

Interaction: The interaction between agents in MAS can be direct through communication and indirect by observation.

Simulated Environment: Agents are dependent on their local environment; this environment can be either a spatial space or some relative locations between agents. Thus separating agents and their local environment would increase the network communication load. Therefore, a well-balanced and distributed ABMS needs to take the dependence of agents on their environment into account at modeling stage.

Mobility: Mobility is the ability of an agent to migrate between machines in a network of machines. Mobility impacts the computation and communication load on the agent.

Adaptivity: Adaptivity means that agents can learn from their experience and adapt to new environments. Therefore,

adaptivity is an important feature that can lead to agents' autonomous behavior.

B. *Simulation Services*

Since scientists from various domains increasingly need agent based simulation for understanding real world phenomena, the need for simulation services such as visualization and data analysis [4] that facilitates rapid development of multi agent systems is growing.

- **Visualization.** An ABMS platform should provide means of visualizing the simulated scenario. This includes a variety of textual and graphical (2D or 3D) browsers that allow the modeler to detect trends and relationships in the simulation scenario. Agent's state is dynamic and to be visually reflected within an acceptable time limit. The simulation produces data concerning actions, events, and communication between agents in addition to agents' states.
- **Data Analysis.** Data analysis is the process of looking at and summarizing data with the intent to extract useful information and develop conclusions.

C. *Simulation Platforms*

There are a vast number of platforms used in simulation, for example [4], [2], [1] and [10] present surveys about platforms used, and provid a comprehensive selection criteria for selecting simulation platform, below are several of these tools:

NetLogo http://ccl.northwestern.edu/netlogo/
RePast http://repast.sourceforge.net
Swarm http://www.swarm.org
Quicksilver http://sourceforge.net/projects/javu4u
VSEit http://www.vseit.de

IV. NETLOGO

NetLogo [11] is a multi-agent programming and modeling environment based on JAVA platform. NetLogo conceived for simulating complex phenomena, namely large-scale patterns that arise out of the complex interactions of numerous independent micro-agents at a lower scale [12]. Many of these emerging patterns can be found in nature and social sciences. Despite its simplicity, NetLogo is a powerful tool in many fields of research and education. It is especially well suited for modeling large collections of interacting or independent agents developing over time, so it is a promising solution for simulating and analyzing complex problems.

Beginners find it an easy-to-learn, intuitive, and well-documented programming language with an elegant graphical interface as shown in figure-4. Researchers can take advantage of NetLogo's advanced features, such as BehaviorSpace that runs automated experiments, 3D visualization, user extensibility, a System Dynamics Modeler that enables mixing agent-based and aggregate representations, and NetLogoLab, which connects to external physical devices.

NetLogo is free and works on all major computing platforms. It is one of the most widely used Multi-Agent modeling tools today [13], with a community of over twenty thousands of users worldwide. NetLogo is convenient platform for the analysis of any complex system developing over time, as the programmer can give instructions to thousands of agents all operating concurrently.

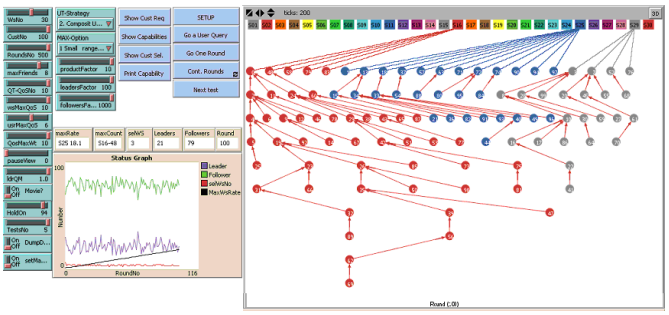

Figure-4: Example of NetLogo Graphic User Interface (GUI) of our Service Selection Simulation Studio

NetLogo comes with extensive documentation, including a library with over 150 sample models in a range of domains, tutorials, a primitive's dictionary, and code examples. An exhaustive user manual is included in the package [14]. Moreover, the embedded Models Library can be used and modified. With this "extensible modeling" approach [12], even beginners can easily get familiar with the language and acquire skills in order to build their own models.

Although a complete description of the environment is beyond the scope of this paper, here is a list [12] of the most interesting features (for further details, please refer to the user NetLogo User Manual [14]): (1) Simple language structure (2) Unlimited number of agents and variables (3) Large vocabulary of built-in language primitives (4) Extensive Models Library with Code Examples (5) Friendly and multi-purpose Model's Interface (6) Monitors for inspecting and controlling agents (7) Info tab for annotating variables and debugging (8) Exchange (export\Import) data with other applications.

Finally, the significance advantage of NetLogo can be summarized as follows: it is a simple and easy to use, but it is advanced enough to serve as powerful tool for analysis and simulation for complex problems. In this way, the researcher can quickly develop his models without being excessively concerned on the programming and debugging practice.

V. SERVICE SELECTION SIMULATION STUDIO (SSSS)

Web Service Selection is a complex process where a service that best satisfies user preferences is selected from a set of candidate services based on user requirements [15]. As per the selection criteria, various nonfunctional (QoS) properties such as service level agreements (SLA), duration and cost, can be used and expressed as user preferences.

We propose a recommender system that utilizes Social Network Analysis (SNA) in making recommendations to customers to select the best service based on their query and their behavior. We used simulation to investigate the following issues:

1. Is a recommender system a feasible approach to make recommendation? And to what extend?
2. Benchmark our proposed selection algorithm with other selection algorithms.

Follow the leader[1] in dynamic social networks [16], is a model of opinion formation with dynamic confidence in agent-mediated social networks where the profiling of agents as leaders or followers is possible. An opinion leader is specified as a highly self-confident agent with strong opinions. An opinion follower is attracted to those agents in which it has more confidence.

Each leader has specific preferences within a service selection context, while each follower imitates one or more leaders in their selection. Some customers don't have such preferences because simply they do not know such preferences prior to the Web Service selection process. Such innate (personal) preferences play a constraint on selecting their leader and consequently on selecting the Web Service. So, followers who have some preferences finally chose their leader on the basis of matching their preferences with the perceived preferences exhibited by the leaders. Social network analysis (SNA) is adopted to find the appropriate leader for a follower.

A. Theoretical Background:

The relationship between two agents defined as a ***confidence function*** [17]**,** such that: an agent (i) increases its confidence in another agent (j) based on how well (j's) opinion meets the criteria specified in i's mind-set. A mind-set here represents the agent preferences.

User Query (Request):

$\boldsymbol{U_R}$ = Set of user attributes captured from user query (request)

Let (m) be the number of nonfunctional (QoS) attributes that considered as constraints over the selection process, such that:

$$U_R = q_r^m : (q_r^1, q_r^2, ... q_r^j .. q_r^m) \quad (1 \le j \le m), \qquad (1)$$

$$q_r^j = \{q_N^j, q_O^j, q_V^j, q_W^j\} \qquad (2)$$

Where:

N: *is QoS attribute name,* ***O:*** *is the operator constraint* $(\le, \ge)$ to maximize or minimize selected value. ***V:*** *is the value constraint,* ***W*** *is the weight constraint.*

For example user might search for: (home loan interest rate ≤ 5.5%, with weight 10).

Functional Match Services:

The set of published services common in their functionality considered for QoS selection process, defined as:

$$S_M = S_M^n : (S_M^1, S_M^2, ..., S_M^n) \qquad (3)$$

Each atomic service has a set of QoS attributes mapped to the user nonfunctional (QoS) attributes, such that:

$$S_i = q_i^m : (q_i^1, q_i^2, ..., q_i^j, .. q_i^m), \quad (1 \le j \le m) \qquad (4)$$

B. Selection Algorithm:

1. For each non-available QoS attribute (q_i^j) value in service S_i *where* $(1 \le j \le m)$, use the worst offered value from other services.

2. Normalize weights (priorities) given by user, such that: $\sum_{j=1}^{m} w_j = 1$

3. Compute agent utility function from each atomic service.

4. Select the atomic service with the highest utility function as the best candidate.

VI. Simulation Results and Analysis

We have developed a Service Selection Simulation tool utilizing the NetLogo platform [11]. The user interface as shown in figure-4, and we use it to analyze and evaluate the validity of our approach, in the following section we outline the testing environment and associated outcomes.

A. Simulation model:

Our model as shown in figure-4 composed of a fixed number of atomic services (30) with the same functional properties and vary in their (QoS) attributes, each atomic service maintains the following information: (service code, a list of QoS attribute codes and corresponding values). Each service' information is hidden from other services and static during any simulation session.

Each simulation session is composed of a fixed set of rounds (500). In each round a number of customers (100) enter their queries to the system. Each customer (agent) has the following information: (ID, a list of attribute QoS codes and corresponding values and weights –priorities- and a set of variable number of friends).

The system evaluates customers based on their query and their friends information, if the customer has no friends and has a number of QoS attributes greater than a predefined threshold value, then the customer qualified as a "leader" otherwise it qualified as a "follower". So, a follower has at least one friend. As well, the system maintains a service rating algorithm based on their number of QoS attributes provided by each service and customers' feedback. All services have the same rate in the beginning of each session.

Each simulation session starts by setting the services and customers with their corresponding information. Each round starts with a new set of customers each with new query preferences and friends information. No communication between customers is assumed even if they are friends. In each round, each customer passes its query and information to the recommender system, if the customer is a qualified leader, then the system performs the service selection process, selecting the best service from available services according to the selection process mentioned previously. If the customer is a follower, then the system finds the best friend from customer friends to follow. However in real life scenario, friends can be discovered using social network analysis tools. Friend relationships can be captured directly on the basis of Confidence Relation. Each customer provides feedback to the system about his satisfaction, and this feedback can be used to influence the evolution of the underlying network connections. By the end of each transaction, the system updates its registry for each service with its new rate based on customers' feedback.

[1] www.business.uts.edu.au/finance/research/wpapers/wp155.pdf

B. Simulation results:

The right window of Figure-4 shows the virtual word of our simulation studio, it contains services in the upper-most rows, followed by leaders with lines connecting leaders to services, then followers with direct links to their leaders.

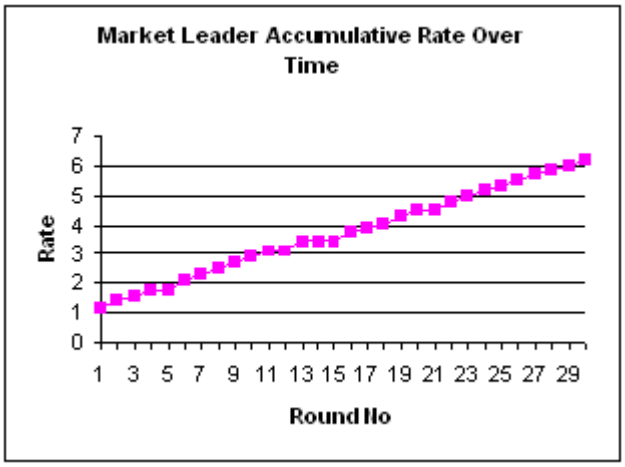

Figure-5: Market Leader Accumulative Rate Over Time

1) Market Leader Analysis:

Market leader (best service) is one of the services with high capability. As shown in figure-5, it shows high attraction ratio in the first couple of rounds and continues attracting leader customers and their followers. It is not necessary to be the leader in all rounds. From various test results market leader shows attraction ratio of about (40%) of all transactions, while other competitors share the rest based on their capability measure.

2) Customer Behavior Analysis:

From the simulation settings, we note that the number of customers who are qualified as "Leaders" varies in each round. On average (21%) of the customers are qualified as leaders each round. Which concludes the model promotes adequate number of leaders' opinions to establish the recommender system knowledge base.

Based on the previous conclusion, our recommender system can support on average (79%) of users who act as followers, providing them with high quality recommendations. This conclusion highly supports the usability of recommender system.

In order to create an efficient user personalized service, our framework recommends the best service based on user preferences in a specific Domain\context. This improves user query performance to discover and select best and frequent Web service, and provides the highest accuracy in its recommendation for the current user. So, we can conclude that our framework provides an effective approach to capture user preferences within a specific domain and context and provides an efficient user personalized service.

VII. CONCLUSION

In this paper we have seen that ABMS has been applied to a wide range of MAS research and designed problems, from models of complex individual agents employing sophisticated internal mechanisms to models of large-scale societies of relatively simple agents that focus more on the interactions between agents. Social simulation is playing an increasingly important role in today's interconnected society. Agent-based social simulation has already been widely applied to many promising areas, such as social and psychological research.

From our experience with modeling Social Networks in Service Selection, utilizing Service Selection Simulation Studio (SSSS), we conclude that (SSSS) can be considered as adequate tool for modeling and simulation "Follow the Leader" in a recommender system for service selection problem. So, we recommend NetLogo as an efficient platform for modeling and simulating similar problems.

REFERENCES

[1] C. M. Macal and M. J. North, "Agent-based modeling and simulation: desktop ABMS," 2007, pp. 95-106.

[2] P. Davidsson, J. Holmgren, H. Kyhlback, D. Mengistu, and M. Persson, "Applications of agent based simulation," *Lecture Notes in Computer Science,* vol. 4442, p. 15, 2007.

[3] P. Edwards, A. Preece, E. Pignotti, G. Polhill, and N. Gotts, "Lessons Learnt from Deployment of a Social Simulation Tool to the Semantic Grid," 2005, pp. 22-24.

[4] Y. U. Zhang, M. Lewis, and M. Sierhuis, "12 Programming Languages, Environments, and Tools for Agent-Directed Simulation," 2008.

[5] D. Pozdnyakov, "An overview of the agent-based social system simulation tools," 2006.

[6] X. Li, W. Mao, D. Zeng, and F. Y. Wang, "Agent-Based Social Simulation and Modeling in Social Computing," *Lecture Notes in Computer Science,* vol. 5075, pp. 401-412, 2008.

[7] M. Wooldridge and P. Ciancarini, "Agent-Oriented Software Engineering: The State of the Art," *Agent-Oriented Software Engineering,* vol. 1957, pp. 1-28, 2001.

[8] E. By and N. Protogeros, "Agent and Web Service Technologies in Virtual Enterprises," *CHOICE,* vol. 45, 2008.

[9] I. Stamatopoulou, I. Sakellariou, P. Kefalas, and G. Eleftherakis, "OPERAS for Social Insects: Formal Modelling and Prototype Simulation," *SCIENCE AND TECHNOLOGY,* vol. 11, pp. 267-280, 2008.

[10] R. Tobias and C. Hofmann, "Evaluation of free Java-libraries for social-scientific agent based simulation," *Journal of Artificial Societies and Social Simulation,* vol. 7, 2004.

[11] NetLogo, "NetLogo Home Page. On line at: http://ccl.northwestern.edu/netlogo/," 2009.

[12] F. Albiero, F. H. P. Fitzek, and M. Katz, "Introduction to NetLogo," *Cognitive Wireless Networks: Concepts, Methodologies and Visions Inspiring the Age of Enlightenment of Wireless Communications,* p. 579, 2007.

[13] P. Blikstein, D. Abrahamson, and U. Wilensky, "NetLogo: Where we are, where we're going," 2005.

[14] NetLogo, "NetLogo User Manual version 4.0.4, http://ccl.northwestern.edu/netlogo/docs/," 2009.

[15] T. Vitvar, A. Mocan, M. Kerrigan, M. Zaremba, M. Moran, E. Cimpian, T. Haselwanter, and D. Fensel, "X- Semantically-enabled service oriented architecture: concepts, technology and application," *Service Oriented Computing and Applications,* vol. 1, pp. 129-154, 2007.

[16] D. Goldbaum, "Follow the Leader: Simulations on a Dynamic Social Network," vol. UTS Finance and Economics Working Paper No 155, http://www.business.uts.edu.au/finance/research/wpapers/wp155.pdf, 2008.