Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

AGENT-BASED MODELING AND SIMULATION: ABMS EXAMPLES

Charles M. Macal
Michael J. North

Center for Complex Adaptive Systems Simulation (CAS^2)
Decision & Information Sciences Division
Argonne National Laboratory
Argonne, IL 60439, USA

ABSTRACT

Agent-based modeling and simulation (ABMS) is a new approach to modeling systems comprised of autonomous, interacting agents. ABMS promises to have far-reaching effects on the way that businesses use computers to support decision-making and researchers use electronic laboratories to support their research. Some have gone so far as to contend that ABMS "is a third way of doing science," in addition to traditional deductive and inductive reasoning (Axelrod 1997). Computational advances have made possible a growing number of agent-based models across a variety of application domains. Applications range from modeling agent behavior in the stock market, supply chains, and consumer markets, to predicting the spread of epidemics, the threat of bio-warfare, and the factors responsible for the fall of ancient civilizations. This tutorial describes the theoretical and practical foundations of ABMS, identifies toolkits and methods for developing agent models, and illustrates the development of a simple agent-based model.

1 INTRODUCTION

Agent-based Modeling and Simulation (ABMS) is a new modeling paradigm and is one of the most exciting practical developments in modeling since the invention of relational databases. ABMS promises to have far-reaching effects on the way that businesses use computers to support decision-making and researchers use electronic laboratories to support their research (North and Macal 2007).

The goals of this tutorial are to show how ABMS is:

- Useful: Why ABMS is good and even better than many conventional modeling approaches in many cases,
- Usable: How we are progressively advancing to usable ABMS systems, with better software development environments and more application experiences, and
- Used: How ABMS is being used to solve practical problems.

This tutorial is organized into two parts. The first part is a tutorial on how to *think* about ABMS. The background on ABMS and its motivating principles are described to illustrate its main concepts and to indicate the state-of-the-art. The second part is a tutorial on how to *do* ABMS. Practical applications of ABMS are described. ABMS toolkits are introduced, and the development of a simple agent-based model is illustrated.

2 HOW TO THINK ABOUT ABMS

2.1 What is an Agent

Although there is no universal agreement on the precise definition of the term "agent," definitions tend to agree on more points than they disagree. Some modelers consider any type of independent component (software, model, individual, etc.) to be an agent (Bonabeau 2001). An independent component's behavior can range from simple, reactive if-then decision rules, to general behavioral models, such as the BDI (belief-desire-intent) framework, to complex models based on artificial intelligence (AI). Some authors insist that a component's behavior must be adaptive in order for it to be considered an agent. The agent label is reserved for components that can learn from their environments and change their behaviors in response to their experiences. Casti (1997) argues that agents should contain both base-level rules for behavior as well as a higher-level set of "rules to change the rules." The base-level rules provide responses to the environment while the "rules to change the rules" provide adaptation. Jennings (2000) provides a computer science view of agency emphasizing the essential characteristic of *autonomous* behavior. The fundamental feature of an agent is the capability to make independent decisions. This requires agents to be active responders and planners rather than purely passive components.

From a practical modeling standpoint, we consider agents to have certain characteristics (Figure 1):

- An agent is an identifiable, discrete, or modular, individual with a set of characteristics and rules governing its behaviors and decision-making capability. Agents are self-contained. The discreteness requirement implies that an agent has a boundary and one can easily determine whether something is part of an agent, is not part of an agent, or is a shared characteristic.
- An agent is autonomous and self-directed. An agent can function independently in its environment and in its interactions with other agents for the limited range of situations that are of interest.
- An agent is social, interacting with other agents. Agents have protocols for interaction with other agents, such as for communication. Agents have the ability to recognize and distinguish the traits of other agents.
- An agent is situated, living in an external environment with which the agent interacts in addition to other agents.
- An agent may be goal-directed, having goals to achieve (not necessarily objectives to maximize) with respect to its behaviors. This allows an agent to compare the outcome of its behavior to the goals it is trying to achieve.
- An agent is flexible, having the ability to learn and adapt its behaviors based on experience. This requires some form of memory. An agent may have rules that modify its rules of behavior.

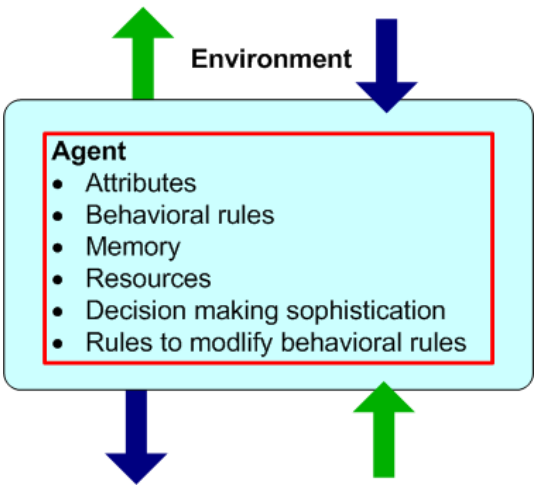

Figure 1: An agent

Often, the agents in an agent-based model will lack one or more of these characteristics. It may not be necessary to model adaptation in a supply chain model if the model's purpose is to evaluate a set of specific inventory management rules, for example. Should such a model be considered an agent-based model? It depends on the structure of such a model. If the structure is such that the model does not preclude the addition of agent characteristics and behaviors with minor modifications to it, then it makes sense to refer to the model as agent-based. We have coined the term *proto-agent* to refer to agents that are missing one or more of the characteristics enumerated above but to which the characteristics can easily be added without modification to the structure of the model.

Unlike particle systems (idealized gas particles for example) which are the subject of the field of particle simulation, agents are diverse, heterogeneous, and dynamic in their attributes and behavioral rules, as shown in Figure 1. Behavioral rules can vary in their sophistication, how much information is considered in the agent decisions (cognitive "load"), the agent's internal models of the external world including the possible reactions or behaviors of other agents, and the extent of memory of past events the agent retains and uses in its decisions. Agents also vary by their attributes and accumulated resources.

Agent-based modeling is known by many names. ABM (agent-based modeling), ABS (agent-based systems or simulation), and IBM (individual-based modeling) are all widely-used acronyms, but "ABMS" will be used throughout this discussion. The term "agent" has connotations in realms other than agent-based modeling as well. ABMS agents are different from the agents typically found in mobile agent systems. "Mobile agents" are lightweight software proxies that roam over the world-wide web and perform various functions for users and to some extent can behave autonomously.

Another point of clarification concerns the term "simulation." Agent-based simulation refers to a model in which the dynamic processes of agent interaction are simulated repeatedly over time, as in systems dynamics, and time-stepped, discrete-event, and other types of conventional simulation. An agent-based model, more generally, is a model in which repeatedly agents interact. For example, when agents collectively optimize behavior through simple exchanges of information as is done in ant colony optimization or a particle swarm optimization models, the purpose is to achieve a desired end-state, i.e., the optimized system, rather than to simulate a dynamic process for its own sake.

ABMS has roots in the fields of multi-agent systems (MAS) and robotics from the field of AI, as well as Artificial Life (ALife). But ABMS is not only tied to understanding and designing "artificial" agents. Its main roots are in modeling human social and organizational behavior and individual decision-making (Bonabeau 2001). With this, comes the need to represent social interaction, collaboration, group behavior and the emergence of higher order social structures.

2.2 The Need for Agent Based Modeling

Why is agent-based modeling becoming so widespread? The answer is because we live in an increasingly complex world. First, the systems that we need to analyze and model are becoming more complex in terms of their interdependencies. Traditional modeling tools are no longer as applicable as they once were. An example application area is the deregulation of the formerly centralized electric power industry in which agents are suddenly free to make pricing and investment choices based on their individual criteria. Second, some systems have always been too complex for us to adequately model. Modeling economic markets has traditionally relied on the notions of perfect markets, homogeneous agents, and long-run equilibrium because these assumptions made the problems analytically and computationally tractable. We are beginning to be able to relax some of these assumptions and take a more realistic view of these economic systems through ABMS. Third, data are being collected and organized into databases at finer levels of granularity. Micro-data can now support individual-based simulations. And fourth, but most importantly, computational power is advancing rapidly. We can now compute large-scale micro-simulation models that would not have been plausible just a couple of years ago.

2.3 Background on ABMS

ABMS has connections to many other fields including complexity science, systems science, systems dynamics, computer science, management science, several branches of the social sciences, and traditional modeling and simulation. ABMS draws on these fields for its theoretical foundations, its conceptual world view and philosophy, and for applicable modeling techniques.

ABMS has its direct historical roots in complex adaptive systems (CAS) and the underlying notion that "systems are built from the ground-up," in contrast to the top-down systems view taken by systems dynamics. CAS concerns itself with the question of how complex behaviors arise in nature among myopic, autonomous agents. In addition, ABMS tends to be descriptive, with the intent of modeling the actual or plausible behavior of individuals, rather than normative such as traditional operations research (OR), which seeks to optimize and identify optimal behaviors.

The field of CAS was originally motivated by investigations into *adaptation* and *emergence* of biological systems. CAS have the ability to self-organize and dynamically reorganize their components in ways better suited to survive and excel in their environments, and this adaptive ability occurs, remarkably, over an enormous range of scales. John Holland, a pioneer in the field, identifies properties and mechanisms common to all CAS (Holland 1995) such as (1) Aggregation: allows groups to form, (2) Nonlinearity: invalidates simple extrapolation, (3) Flows: allow the transfer and transformation of resources and information, and (4) Diversity: allows agents to behave differently from one another and often leads to the system property of robustness. CAS mechanisms are: (1) Tagging: allows agents to be named and recognized, (2) Internal models: allows agents to reason about their worlds, and (3) Building blocks: allows components and whole systems to be composed of many levels of simpler components. These CAS properties and mechanisms provide a useful reference for designing agent-based models.

2.3.1 Simple Rules Result in Emergent Organization and Complex Behaviors

The discussion on the background of ABMS begins with a simple game developed by the mathematician John Conway, the "Game of Life" (Gardner 1970). The GOL, as it is called, is based on cellular automata (CA). Perhaps the simplest way to illustrate the basic ideas of agent-based modeling and simulation is through CA. According to Casti (1997), the original notion of CA was developed by the physicist Stanislaw Ulam in response to a question posed by the famous 20th century mathematician John von Neumann. The question was, "could a machine be programmed to make a copy of itself?" In effect, the question had to do with whether it was possible to develop a logical structure that was complex enough to completely contain all of the instructions for replicating itself. The answer turned out to be yes, and it was eventually found in the abstract mathematical representation of a machine in the form of a cellular automata.

A typical CA is a two-dimensional grid or lattice consisting of cells. Each cell assumes one of a finite number of states at any point in time. A set of simple rules determines the value of each cell based on the cell's previous state. Every cell is updated each period according to the rules. The next value of a cell depends on the cell's current value and the values of its immediate neighbors in the eight surrounding cells. Each cell is identical in terms of its update rules. A CA is deterministic in that the same state for a cell and its neighbors always results in the same updated state. The GOL has three rules that determine the next state (either On or Off) of each cell:

1. The cell will be On in the next generation if exactly three of its eight neighboring cells are currently On.
2. The cell will retain its current state if exactly two of its neighbors are On.
3. The cell will be Off otherwise.

Figure 2 shows a snapshot from a GOL simulation showing an initial random distribution of On cells. After sev-

eral updates of all cells in the grid, distinctive patterns emerge, and in some cases these patterns can sustain themselves indefinitely throughout the simulation (Figure 3). The eight-neighbor per neighborhood assumption built into the GOL determines the scope of agent interaction and the locally available information for each cell to update its state.

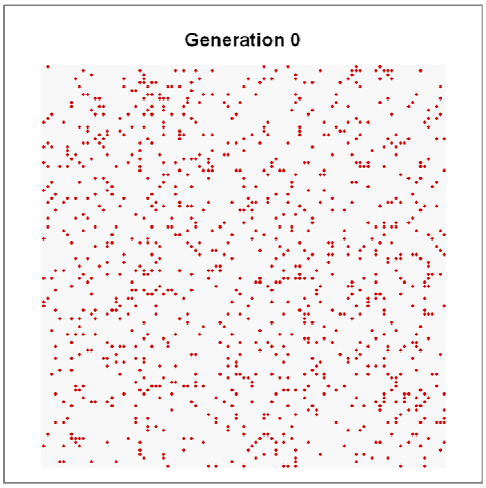

Figure 2: Game of Life simulation, initial random layout of cells in the On state

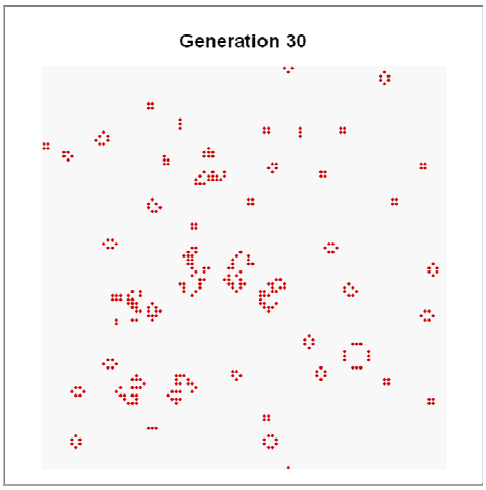

Figure 3: Game of Life simulation, after all cells have been updated 30 times

Two observations are important about the GOL rules: (1) The rules are simple, and (2) the rules use only local information. The state of each cell is based only on the current state of the cell and the cells touching it in its immediate neighborhood. Although these are interesting findings, and observing the patterns created by repeated simulations of the GOL reveals a world of endless creations, other observations have implications for practical ABMS:

- Sustainable patterns can emerge in systems that are completely described by simple rules that are based on only local information, and
- The patterns that may develop can be extremely sensitive to the initial conditions.

The Boids simulation is a good example of how interacting agents, characterized by simple behavioral rules, lead to emergent and seemingly organized behavior (Reynolds 2006). Agent behavior is reminiscent of schooling or flocking behavior in fish or birds. In the Boids model, each agent has three rules governing its movement:

1. Cohesion: each agent steers toward the average position of its nearby "flockmates,"
2. Separation: each agent steers to avoid crowding local flockmates, and
3. Alignment: each agent steers towards the average heading of local flockmates.

Here, nearby or local refers to agents in the immediate neighborhood of an agent as defined by the straight-line distance. A fourth rule is added to the above three rules to ensure that the boids stay close to their initial area. Initially, a set number of boids are randomly assigned positions and orientations (Figure 4). Even with only these simple rules applied at the individual agent level and only to the agents in its "neighborhood", the agents' behaviors begin to appear coordinated, and a leaderless flock emerges (Figure 5).

Two observations are important about the Boids rules: (1) the rules are simple, and (2) the rules use only local information. We can make some observations from the Boids model that have implications for practical ABMS. First, sustainable patterns can emerge in systems that are completely described by simple deterministic rules based on only local information. Second, repeated experiments (not shown here) demonstrate that the patterns that develop can be extremely sensitive to the initial conditions - in this case, the initial random positions and orientations of the boids. .

Based on simple rules of behavior and agent interaction, natural systems seemingly exhibit collective intelligence, or *swarm intelligence*, even without the existence of or the direction provided by a central authority. How is it that an ant colony can organize itself to carry out the complex tasks of food gathering and nest building and at the same time exhibit an enormous degree of resilience if the colony is seriously disrupted? Natural systems are able to not only survive, but also to adapt and become better suited to their environment, effectively optimizing their behavior over time. Swarm intelligence has inspired practical optimization techniques, such as ant colony optimization that have been used to solve practical schedul-

ing and routing problems (Bonabeau, Dorigo and Theraulaz, 1999).

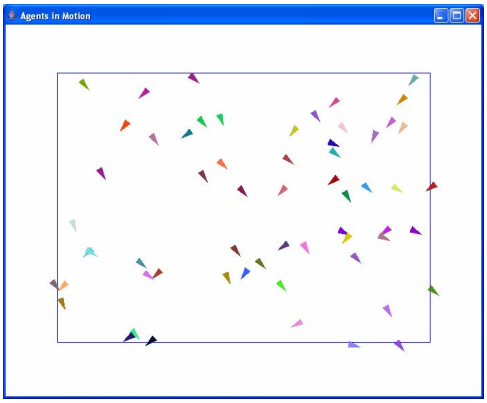

Figure 4: Boids simulation, initial random configuration

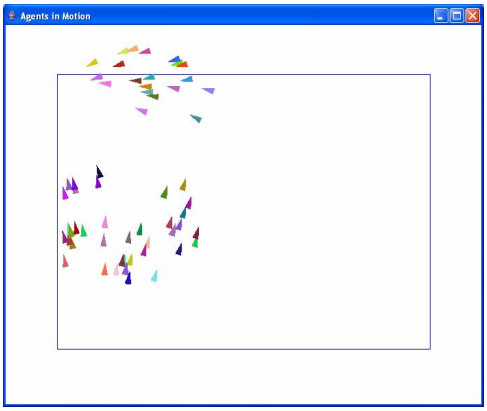

Figure 5: Boids simulation, after 500 updates showing two apparent clusters of agents

2.3.2 Agent-Based Modeling in the Sciences

In applications of ABMS to social processes, agents represent people or groups of people, and agent relationships represent processes of social interaction (Gilbert and Troitzsch 1999). The fundamental assumption is that people and their social interactions can be credibly modeled at some reasonable level of abstraction for at least specific and well-defined purposes, if not in general. This limited scope for representing agent behaviors in ABMS contrasts with the more general goals of AI. From an ABMS perspective, some important questions become immediately apparent: (1) how much do we know about credibly modeling people's behavior? and (2) do we know enough about human social interaction to credibly model it? These two questions have spawned and to some extent reinvigorated basic research programs in the social sciences that have the promise of informing ABMS on theory and methods for agent representation and behavior.

Sakoda (1971) formulated one of the first social agent-based models, the Checkerboard Model, which had some of the key features of a cellular automaton. Schelling applied cellular automata to study housing segregation patterns and posed the question, "is it possible to get highly segregated settlement patterns even if most individuals are, in fact, color-blind?" (Schelling 1978). The Schelling model demonstrated that ghettos can develop spontaneously in the sense that system-level patterns can emerge that are not necessarily implied or even consistent with the objectives of the individual agents.

Extending the notion of modeling people to growing entire artificial societies through agent simulation was taken up by Epstein and Axtell in their groundbreaking Sugarscape model (Epstein and Axtell 1996). In numerous computational experiments, Sugarscape agents emerged with a variety of characteristics and behaviors, highly suggestive of a realistic, although rudimentary and abstract, society. Emergent processes were observed that Epstein and Axtell interpreted as death, disease, trade, wealth, sex and reproduction, culture, conflict and war, and externalities such as pollution.

Economics is adopting agent-based modeling to an extent. Some of the classical assumptions of standard micro-economic theory are: (1) economic agents are rational, which implies that agents have well-defined objectives and are able to optimize their behavior (the basis for the "rational agent" model used in economics and many other social science disciplines), (2) agents are homogeneous, having identical characteristics and rules of behavior, (3) there are decreasing returns to scale from economic processes, decreasing marginal utility, decreasing marginal productivity, etc., and (4) the long-run equilibrium state of the system is the primary information of interest. Each of these assumptions can be relaxed in ABMS applications to economic systems. First, do organizations and individuals really optimize? Herbert Simon, a Nobel Laureate who pioneered the field of AI, developed the notion of "satisficing" to describe what he observed people and organizations actually do in the real world (Simon 2001). Behavioral economics is a relatively new field that incorporates experimental findings on psychology and cognitive aspects of agent decision making to determine people's actual economic and decision making behavior. Second, that diversity among individuals universally occurs in the real-world is a key observation of complexity science. Many natural organizations from ecologies to industries are characterized by populations whose diversity gives rise to its stability and robustness. Third, "positive feedback loops" and "increasing returns" have been identified as underlying dynamic processes of rapid exponential growth in economic systems (Arthur, Durlauf and Lane, 1997). Positive feedback can create self-sustaining processes that quickly take a system away from its starting point to a faraway state. Fourth, long-run equilibrium

states are not the only results of interest. The transient states that are encountered along the way to a long-run state are often of interest. Furthermore, not all systems come to an equilibrium (Axtell 2000). The field of Agent-based Computational Economics (ACE) has grown up around the application of ABMS to economic systems (Tesfatsion 2002, 2005).

Archaeologists and anthropologists are developing large-scale agent-based simulations of ancient civilizations to help explain their growth and decline, based on archaeological data. ABMS has been applied to explain the prosperity of ancient cities in Mesopotamia and understand the social and environmental factors responsible for the disappearance of the Anasazi in the southwestern U.S. (Koehler, Gumerman and Reynolds, 2005).

Sociologists are doing agent-based modeling as well. Macy and Willer (2002) consider agent-based modeling as an approach to modeling the social life of interacting, adaptive social agents. Cognitive science has had its own notion of agency, and social cognitive science is extending these ideas to social settings (Bedau 2003). Synthetic social agents that include models of the influence of emotion and cognition on social behavior are being developed by cognitive scientists and others (Gratch and Marsella 2001). Computational social science is becoming a subfield in the social sciences (Sallach and Macal 2001).

2.3.3 Topologies as a Basis for Social Interaction

As much as modeling agent behaviors, agent modeling also concerns itself 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. Cellular automata represent agent interaction patterns and available local information by using a grid or lattice, and the cells immediately surrounding an agent are its neighborhood. Other agent interaction topologies, such as networks, allow an agent's neighborhood to be defined more generally and may more accurately describe social agents' interaction patterns.

Social Network Analysis (SNA) is a field with a long history that studies the characterization and analysis of social structure and interaction through network representations. Traditionally, SNA has focused on static networks, i.e., networks that do not change their structure over time or as a result of agent behavior. Recently, much progress has been made in understanding the processes of growth and change of real-world networks (Barabási 2002). Dynamic network analysis (DNA) is a new field that incorporates the mechanisms of network growth and change based on agent interaction processes (NRC 2003). Understanding the agent rules that govern how networks are structured and grow, how quickly information is communicated through networks, and the kinds of relationships that networks embody are important aspects of "network ABMS."

2.3.4 Modeling Agent Processes

Identifying the social interaction mechanisms for how cooperative behavior emerges among individuals and groups is an interesting question with practical implications. Evolutionary Game Theory is related to traditional game theory and takes into account the repeated interactions of the players and their effect on strategies. Axelrod has shown that a simple Tit-For-Tat strategy of reciprocal behavior toward individuals is enough to establish sustainable cooperative behavior (Axelrod 1997). The broader need is for a generative type of social science in which the processes from which social structure emerges can be understood as the necessary result of social interactions (Epstein 2007).

3 HOW TO DO ABMS

3.1 Agent-based Modeling Areas

Table 1 lists practical agent-based modeling and simulation applications in many areas. ABS applications range from modeling agent behavior in the stock market (LeBaron 2002) and supply chains (Fang et al. 2002, Macal 2004a), to predicting the spread of epidemics (Huang et al. 2004) and the threat of bio-warfare (Carley 2006), from modeling the growth and decline of ancient civilizations (Kohler, Gumerman and Reynolds, 2005) to modeling the complexities of the human immune system (Folcik and Orosz 2006) and the deregulation of electric power markets (Cirillo 2006), just to name a few.

ABMS applications range across a continuum, from small, elegant, minimalist models to large-scale decision support systems. Minimalist models are based on a set of idealized assumptions, designed to capture only the most salient features of a system. These are exploratory electronic laboratories in which a wide range of assumptions can be varied over a large number of simulations. Decision support models tend to be large-scale applications, designed to answer a broad range of real-world policy questions. These models are distinguished by including real data and having passed some degree of validation testing to establish credibility in their results.

3.2 A Detailed Agent-based Simulation Example

We next describe a hypothetical simulation example provided by Law (2007a). We will implement the example as a simple agent-based model using a fixed-time step approach, noting it has also been programmed as a discrete event simulation (DES) as well (Law, 2007b). The exam-

ple illustrates some of the similarities and differences between agent-based simulation and DES.

Table 1: Agent-based modeling application areas

Business and Organizations • Manufacturing Operations • Supply chains • Consumer markets • Insurance industry Economics • Artificial financial markets • Trade networks Infrastructure • Transportation/traffic • Electric power markets • Hydrogen infrastructure Crowds • Pedestrian movement • Evacuation modeling	Society and Culture • Ancient civilizations • Civil disobedience • Social determinants of terrorism • Organizational networks Military • Command and control • Force-on-force Biology • Population dynamics • Ecological networks • Animal group behavior • Cell behavior and sub cellular processes

Consider an environment consisting of a 12 x 12 grid of cells. Each cell is 100 feet on each side. Initially, each cell is colored green with a probability of 0.8 and red with a probability of 0.2. A cell's state is independent of all other cells. In our ABMS terminology, the grid is the environment. The amount of time that a cell is green before turning red is uniform on the set {70, 71, …, 90} minutes, with mean 80 minutes. The amount of time that a cell is red before turning green is uniform on the set {10, 11,…, 30} minutes, with mean 20 minutes.

Suppose that 100 agents live on the grid. When the agents move, they move at a speed of 60 feet per minute. Initially, each agent is randomly placed on the grid and has a heading that is uniformly distributed on the discrete set of {0, 90, 180, 270} degrees, with zero degrees being north and degrees increasing in the clockwise direction. More than one agent can coexist on a particular cell. An agent can observe the colors of its current cell and its immediately adjacent neighbor cells.

Agents use the following movement rules:

1. An agent will begin moving at its specified heading if, and only if, it starts on a green cell.
2. If an agent starts on a red cell, it will wait until the cell turns green to begin moving.
3. If an agent is moving through a cell and reaches the edge, then it will only enter the next cell if it is green. If the next cell is red, the agent stops and waits until the next cell is green.
4. If an agent is traveling through a cell when it turns red, then it will stop immediately. It will not start moving again until the cell turns green.
5. When an agent reaches one of the edges of the grid, it exits the system, and its total time spent in the simulation is recorded.

The simulation is run until all agents have exited and the statistics for the time spent in the system for all agents is then computed.

An agent's attributes consist of its (x, y) coordinates (in continuous space) and its heading. An agent also "knows" the cell in which it is located. A formal agent attribute specification is as follows:

- *aid*: Unique agent identifier (integer)
- *aloc*, {x, y}: Agent location in two dimensional continuous space.
- *vel*, {vx, vy}: Current velocity (= {0,0} if agent has exited or is located on a red cel
- *vel0*, {vx0, vy0}: Persistent velocity. An agent always assumes this velocity when moving. Note persistent velocity implies an orientation.
- *alive*: Status indicator (-1 indicates the agent is alive. A value greater than 0 indicates the time t at which the agent exited the system).

The other important data structure is the grid cell. A formal cell attribute specification is as follows:

- *cid*: Unique cell identifier.
- *cloc*, {u, v}: Cell location where u and v are integers.
- *switchTime*: The next time at which the cell status will be updated.
- *agentL*: A list of agents that are located in the cell.

The simulation setup is as follows. Initially, 80%of the grid is seeded with green cells and 20% with red cells. The amount of time that a cell is green before turning red is uniformly distributed, U[70, 90]. The amount of time that a cell is red before turning green is uniformly distributed, U[10, 30]. A fixed set of agents is initially distributed with random orientations (four discrete choices) to random locations on the grid.

In summary, the agent aspects of the simulation are as follows:

- Agents are described as discrete and modular entities in the (object-oriented) model implementation (as shown above)
- Agents are autonomous in that they determine whether or not they can move. Individual agents have associated methods that they execute to perform various

functions. For example, *getAgentNeighbors* is a function that an agent uses to find the agents in its current cell, and *getCellStatus* is a function that an agent uses to find out the status of its current cell (red or green).

- An agent's external environment is the grid, from which it extracts information on its cell state.

Some aspects of this example problem do not emphasize the capabilities of ABS because they include the defining characteristics of agents in trivial ways. For example, an agent's goal is simply to keep moving until it has reached the edge of the grid. An agent is social in the sense that it is aware of and recognizes individual agents that are its neighbors in the cell in which it is located, but agent interaction rules are not implemented. Finally, agents are not adaptive. This agent framework does allow the agents to have much more complex behaviors. Modeling each of these aspects of agents could easily be added to the example within the framework described above by adding additional functions that act on the agents.

Figure 6 shows the agent-based simulation logic. The simulation is implemented as a set of two nested loops, one for time, the other for agents. All of the agent behaviors are executed each time period. Agents update their position each time step according to $\{x, y\}_{new} = \{x, y\} + \{vel_x, vel_y\}$. In this example, the order in which the agents update their state does not matter, so the agents are updated in a fixed sequence. If the order of updating the agents did matter, it would be customary practice to either randomize the order of the agent updating or else implement a sequencing procedure that has some rational basis as to why one agent updates itself before the others. This scheduling process could be event-based or even adaptive depending on conditions during the simulation.

The time increment, Δt, is arbitrary and constant throughout the simulation. A value of Δt as small as possible is desirable (constrained by computational resources) to capture the accurate movement of agents as they cross cell and grid boundaries. This example, as implemented, is more like a combined continuous-discrete event simulation than a DES. In effect, with the selection of a small value for Δt, time progresses continuously tracking agent movement over the grid, with discrete events being inserting as cells are updated at specific times.

Snapshots from the simulation are shown in Figures 7 and 8, and recorded exit times are shown in Figure 9.

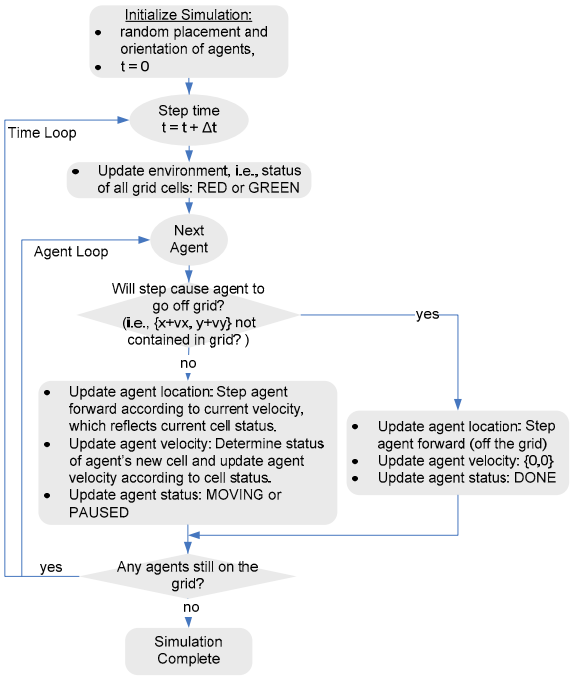

Figure 6: Logic for agent-based simulation example

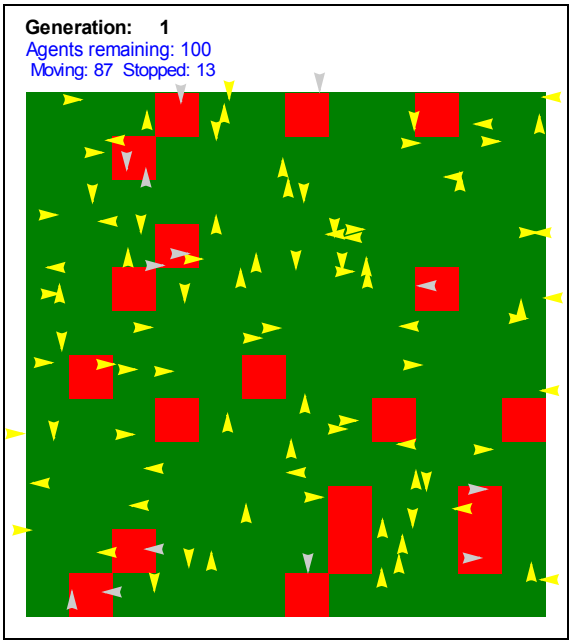

Figure 7: Example agent-based simulation, initial agent states (lightly shaded cells are green, dark cells are red)

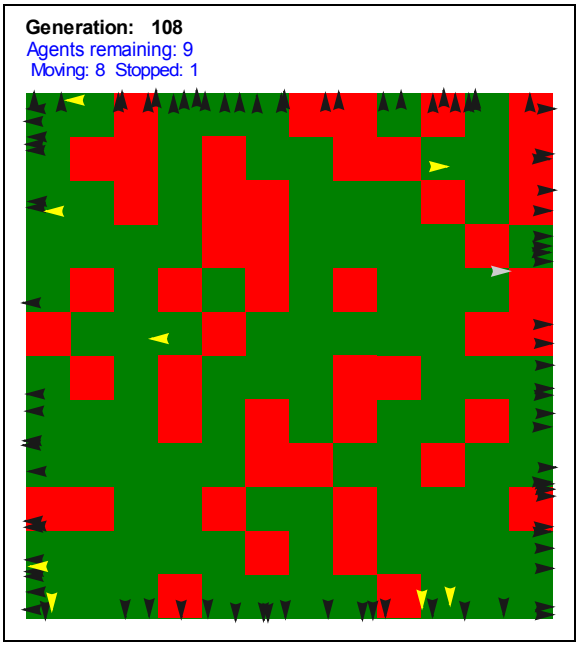

Figure 8: Example agent-based simulation, at time 108 (lightly shaded cells are green, dark cells are red)

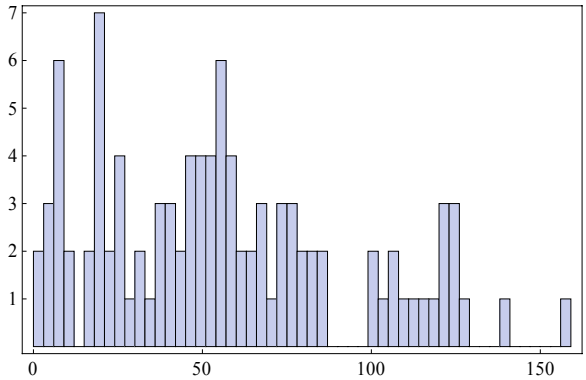

Figure 9: Exit times in agent-based simulation example

3.3 ABMS Software and Toolkits

Agent-based modeling can be done using general, all-purpose software or programming languages, or it can be done using specially designed software and toolkits that address the special requirements of modeling agents. Agent modeling can be done in the small, on the desktop, or in the large, using large-scale computing clusters, or it can be done at any scale in-between these extremes. Projects often begin small, using one of the desktop ABMS tools, and then grow in stages into the larger-scale ABMS toolkits. Often one begins developing their first agent model using the approach that one is most familiar with, or the approach that one finds easiest to learn given their background and experience.

We distinguish several approaches to building ABMS applications in terms of the scale of the software that one can apply according to the following continuum:

Desktop Computing for ABMS Application Development:

- Spreadsheets: Excel using the macro programming language VBA
- Dedicated Agent-based Prototyping Environments: Repast Simphony, NetLogo, StarLogo
- General Computational Mathematics Systems: MATLAB, *Mathematica*

Large-Scale (Scalable) Agent Development Environments:

- Repast
- Swarm
- MASON
- AnyLogic

General Programming Languages:

- Python
- Java
- C++

Desktop ABMS can be used to learn agent modeling, prototype basic agent behaviors, and perform limited analyses. Desktop agent-based models can be simple, designed and developed in a period of a few days by a single computer-literate modeler using tools learned in a few days or weeks. Desktop agent modeling can be used to explore the potential of ABMS with relatively minor time and training investments, especially if one is already familiar with the tool.

Spreadsheets, such as Microsoft Excel, are in many ways the simplest approach to modeling. It is easier to develop models with spreadsheets than with many of the other tools, but the resulting models generally allow limited agent diversity, restrict agent behaviors, and have poor scalability compared to the other approaches. Some useful agent models have been developed using spreadsheet models (Bower and Bunn 2000). In previous WSC papers, we described an spreadsheet implementation of a spatial agent-based shopper model (Macal and North 2007).

Special-purpose agent tools, such as NetLogo, and StarLogo, provide special facilities focused on agent modeling. The most directly visible common trait shared by the various prototyping environments is that they are designed to get first-time users started as quickly as possible. NetLogo is a free ABMS environment (Wilensky 1999) developed at Northwestern University's Center for Connected Learning and Computer-Based Modeling (http://ccl.northwestern.edu/netlogo/). The NetLogo language uses a modified version of the Logo programming

language (Harvey 1997). NetLogo is designed to provide a basic computational laboratory for teaching complex adaptive systems concepts. NetLogo was originally developed to support teaching, but it can be used to develop a wide range of applications. NetLogo provides a graphical environment to create programs that control graphic "turtles" that reside in a world of "patches" that is monitored by an "observer." NetLogo includes an innovative participatory ABMS feature called HubNet (Wilensky and Stroup 1999), which allows groups of people to interactively engage in simulation runs alongside of computational agents.

General-purpose desktop computational mathematics system (CMS) with an integrated development environment, such as MATLAB and *Mathematica*, can be used to develop agent models, although the agent-specific functionality has to be written by the developer from scratch (Macal 2004b). The basic requirements are a full scripting language capability combined with array or list-processing capabilities for efficiency. Computational mathematics systems are structured in two main parts: (1) the user interface that allows dynamic user interaction, and (2) the underlying computational engine, or kernel, that performs the computations according to the user's instructions. The underlying computational engine is written in the C programming language for these systems, but C coding is unseen by the user. The interpreted nature of these systems avoids the compilation and linking steps required in traditional programming languages. Computational mathematics systems have advantages derived from both the mathematical and interactive orientations of these tools. CMS environments have rich mathematical functions, and nearly any mathematical relation or function that can be numerically calculated is available within these tools or their add-on libraries. In some cases, the tools even support symbolic processing and manipulation, which is useful for systems of equations that can be solved analytically (Macal 2004b). If a CMS environment is already familiar, this can be a good place to start agent-based modeling.

Many large-scale ABMS software environments are now freely available. These include Repast (North, Collier and Vos, 2006), Swarm (SDG 2006; Minar et al. 1996), NetLogo (NetLogo 2007) and MASON (GMU 2006) among many others. Proprietary toolkits are also available such as AnyLogic (2006). A recent review and comparison of Java-based agent modeling toolkits is provided by Tobias and Hoffman (2004).

Swarm was the first ABMS software development environment launched in 1994 at the Santa Fe Institute. Swarm was originally written in Objective C and was later fitted with a Java interface. Following the original Swarm innovation, the Repast (REcursive Porous Agent Simulation Toolkit) toolkit was developed as a pure Java implementation (North, Collier and Vos, 2006). Repast has been used extensively in social simulation applications (North and Macal 2005). Repast is a widely used free and open source agent-based modeling and simulation toolkit (ROAD 2007). Repast Simphony (Repast S) is the latest version of Repast, designed to provide visual point-and-click tools for agent model design, agent behavior specification, model execution, and results examination. The Repast S agent model designer is being developed to allow users to visually specify the logical structure of their models, the spatial (e.g., geographic maps and networks) structure of their models, the kinds of agents in their models, and the behaviors of the agents themselves. Once their models are specified, users can use the point-and-click Repast S runtime environment to execute model runs as well as visualize and store results. In addition, the Repast S runtime environment includes automated results analysis connections to a variety of spreadsheet, visualization, data mining, and statistical analysis tools, virtually all of which are free and open source.

3.4 Why and When ABMS

We conclude by offering some ideas on the situations for which agent-based modeling can offer distinct advantages to conventional simulation approaches. When is it beneficial to think in terms of agents? When one or more of the following criteria are satisfied:

- When there is a natural representation as agents
- When there are decisions and behaviors that can be defined discretely (with boundaries)
- When it is important that agents adapt and change their behaviors
- When it is important that agents learn and engage in dynamic strategic behaviors
- When it is important that agents have a dynamic relationship with other agents, and agent relationships form and dissolve
- When it is important to model the processes by which agents form emergent organizations, and adaptation and learning are important at the organization level
- When it is important that agents have a spatial component to their behaviors and interactions
- When the past is no predictor of the future
- When scaling-up to arbitrary levels is important in terms of the number of agents, agent interactions and agent states
- When process structural change needs to be a result of the model, rather than a model input

ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy under contract number DE-AC02-06CH11357. Portions of this tutorial have appeared in previous tutorial papers presented at the Winter Simulation Conference in 2007, 2006 and 2005.

REFERENCES

AnyLogic. 2006. <http://www.xjtek.com/>.

Arthur, W., S. Durlauf and D. Lane. 1997. *The economy as an evolving complex system II*, SFI Studies in the Sciences of Complexity, Addison Wesley: Reading, MA.

Axelrod, R. 1997. *The complexity of cooperation: agent-based models of competition and collaboration*, Princeton, NJ: Princeton University Press.

Axelrod, R. 1997. Advancing the art of simulation in the social sciences, in Conte., R., Hegselmann, R. and Terna, P., eds. *Simulating social phenomena*, Berlin: Springer-Verlag: 21-40.

Axtell, R. 2000. *Why agents? On the varied motivations for agent computing in the social sciences,* Working Paper 17, Center on Social and Economic Dynamics, Brookings Institution, Washington, D.C.

Barabási, A.-L. 2002. *Linked: the new science of networks*. Cambridge, MA: Perseus Pub.

Bedau, M. A. 2003. Artificial Life: Organization, Adaptation and Complexity from the Bottom Up, *TRENDS in Cognitive Sciences* 7(11):505-512.

Bonabeau, E., M. Dorigo, and G. Theraulaz. 1999. *Swarm intelligence: from natural to artificial systems*, Oxford: Oxford University Press.

Bonabeau, E. 2001. Agent-based modeling: methods and techniques for simulating human systems. In *Proceedings of National Academy of Sciences* 99(3): 7280-7287.

Booch, G., J. Rumbaugh, and I. Jacobson. 1998. *The unified modeling language user guide*, Addison-Wesley:New York.

Bower, J., and D. Bunn. 2000. Model-based comparisons of pool and bilateral markets for electricity. *The Energy Journal* 21(3):1–29.

Carley, K. M., D. B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. Chen, B. Kaminsky and D. Nave. 2006. BioWar: scalable agent-based model of bioattacks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 36(2):252-265.

Casti, J. 1997. *Would-be worlds: how simulation is changing the world of science*, New York: Wiley.

Casti, J. 2001. Bizsim: the world of business - in a box, *Complexity International*, 08:6. Available via <http://www.complexity.org.au/ci/vol08/casti01/> [accessed September 23, 2008].

Cirillo, R., P. Thimmapuram, T. Veselka, V. Koritarov, G. Conzelmann, C. Macal, G. Boyd, M. North, T. Overbye, and X. Cheng. 2006. *Evaluating the potential impact of transmission constraints on the operation of a competitive electricity market in Illinois*, Argonne National Laboratory, ANL-06/16 (report to the Illinois Commerce Commission).

Epstein, J. M. 2007. *Generative social science: Studies in agent-based computational modeling*, Princeton University Press:Princeton, NJ.

Epstein, J. M., and R. Axtell. 1996. *Growing artificial societies: social science from the bottom up*, Cambridge, MA: MIT Press.

Fang, C., S. Kimbrough, S. Pace, A. Valluri and Z. Zheng. 2002. On adaptive emergence of trust behavior in the game of stag hunt, *Group Decision and Negotiation* 11(6): 449–467.

Folcik, V., and C. Orosz. 2006. An agent-based model demonstrates that the immune system behaves like a complex system and a scale-free network. *SwarmFest 2006*, University of Notre Dame, South Bend, IN, June.

Gilbert, N., and K. G. Troitzsch. 1999. *Simulation for the social scientist*, Buckingham UK: Open University Press.

Gardner, M. 1970. The fantastic combinations of John Conway's new solitaire game "Life", *Scientific American* 223:120-123.

GMU (George Mason University). 2006. MASON home page. Available via <http://cs.gmu.edu/~eclab/projects/mason/>.

Gratch, J., and S. Marsella. 2001. Tears and fears: modeling emotions and emotional behaviors in synthetic agents, In *Proceedings of 5th International Conference on Autonomous Agents*, 278-285.

Harvey, B. 1997. *Computer Science Logo Style*, MIT Press: Boston, Massachusetts USA

Holland, J. H. 1995. *Hidden order: how adaptation builds complexity*, Addison-Wesley:Reading, Mass.

Huang, C.-Y., C.-T. Sun, J.-L. Hsieh and H. Lin. 2004. Simulating SARS: Small-world epidemiological modeling and public health policy assessments. *JASSS - Journal of Artificial Societies And Social Simulation* 7(4): 100-131.

Jennings, N. R. 2000. On agent-based software engineering, *Artificial Intelligence* 117:277-296.

Kohler, T., G. Gumerman, and R. Reynolds. 2005. Simulating ancient societies, *Scientific American*.

Law, A. M. 2007a. Agent-based simulation: A new approach to systems modeling, presentation, Averill M. Law & Associates, Tucson, AZ

Law, A. M. 2007b. *Simulation modeling and analysis*, 4th ed. New York: McGraw-Hill.

LeBaron, B. 2002. Short-memory traders and their impact on group learning in financial markets. In *Proceedings of National Academy of Sciences* 99(90003): 7201-7206.

Macal, C. 2004a. Emergent structures from trust relationships in supply chains. In *Proceedings of Agent 2004: Conference on Social Dynamics,* eds., C. Macal, D. Sallach and M. North, Chicago, IL, Oct. 7-9, 743-760, Argonne National Laboratory.

Macal, C. M. 2004b. Agent-Based Modeling and Social Simulation with *Mathematica* and MATLAB, In *Proceedings of Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Macal, C., D. Sallach, and M. North, eds., 185-204, Chicago, IL, Oct. 7-9. Available via <http://www.agent2004.anl.gov>.

Macal, C. M., and M. J. North. 2007. Agent-based Modeling and Simulation: Desktop ABMS. *Proceedings of the 2007 Winter Simulation Conference*. Eds. S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 95-106. Washington, DC.

Macy, M., and R. Willer. 2002. From factors to actors: computational sociology and agent-based modeling, *Annual Review of Sociology* 28:143-166.

Minar, N., R. Burkhart, C. Langton, and M. Askenazi. 1996. The Swarm simulation system, a toolkit for building multi-agent simulations. Available via <http://www.santafe.edu/projects/swarm/overview/overview.html>.

NetLogo. 2007. NetLogo home page. Available via <http://http://ccl.northwestern.edu/netlogo>.

North, M. J., and C. M. Macal. 2007. *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*, Oxford University Press: Oxford, U.K.

North, M., N. Collier, and J. Vos. 2006. Experiences in creating three implementations of the repast agent modeling toolkit, *ACM Transactions on Modeling and Computer Simulation*, 16(1):1-25.

North, M. J., and C. M. Macal. 2005. Escaping the accidents of history: an overview of artificial life modeling with Repast. In *Artificial Life Models in Software*, eds. A. Adamatzky and M. Komosinski, Springer-Verlag: Dordrecht, Netherlands.

NRC (National Research Council). 2003. *Dynamic social network modeling and analysis: workshop summary and papers,* R. Brieger, K. Carley, and P. Pattison, Committee on Human Factors, Washington, DC: National Academies Press.

ROAD (Repast Organization for Architecture and Design). 2007. Repast Home Page, Available via <http://repast.sourceforge.net/>.

Reynolds, C. 2006. Boids. Available via <http://www.red3d.com/cwr/boids/>.

Sakoda, J. M. 1971. The checkerboard model of social interaction. *Journal of Mathematical Sociology* 1:119-132.

Sallach, D., and C. Macal. 2001. The simulation of social agents: an introduction, *Social Science Computer Review* 19(3):245–248.

Schelling, T. C. 1978. *Micromotives and macrobehavior,* New York: Norton.

SDG (Swarm Development Group). 2006. Swarm Development Group home page. Available via <http://www.swarm.org>.

Simon, H. 2001. *The sciences of the artificial*, Cambridge, MA: MIT Press.

Simon, J. 2002. *Excel programming*, Wiley Publishing: Hoboken, NJ.

Tesfatsion, L. 2002. Agent-based computational economics: growing economies from the bottom up, *Artificial Life* 8(1):55-82.

Tesfatsion, L. 2005. Agent-based Computational Economics (ACE) home page. Available via <http://<www.econ.iastate.edu/tesfatsi/ace.htm>.

Tobias, R., and C. Hofmann. 2004. Evaluation of free Java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation* 7(1).

Wilensky, U. 1999. *Netlogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University:Evanston, IL USA. Available via <http://ccl.northwestern.edu/netlogo/>.

Wilensky, U., and W. Stroup. 1999. *Hubnet*, Center for Connected Learning and Computer-Based Modeling, Northwestern University: Evanston, IL USA. Available via <http://ccl.northwestern.edu/ps/>.

AUTHOR BIOGRAPHIES

CHARLES M. MACAL, Ph.D., P.E., is the Director, Center for Complex Adaptive Agent Systems Simulation (CAS^2), Argonne National Laboratory. He is a member of the INFORMS-Simulation Society, Society for Computer Simulation International, the Systems Dynamics Society and a founding member of NAACSOS. Charles has a Ph.D. in Industrial Engineering & Management Sciences from Northwestern University. Contact: <macal@anl.gov>.

MICHAEL J. NORTH, M.B.A., Ph.D., is the Deputy Director of CAS^2 at Argonne. Michael has over 15 years of experience developing advanced modeling and simulation applications for the federal government, international agencies, private industry, and academia. Michael has a Ph.D. in Computer Science from the Illinois Institute of Technology. Contact: <north@anl.gov>.