Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

TUTORIAL ON AGENT-BASED MODELING AND SIMULATION

Charles M. Macal
Michael J. North

Center for Complex Adaptive Agent Systems Simulation (CAS^2)
Decision & Information Sciences Division
Argonne National Laboratory
Argonne, IL 60439, U.S.A.

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 besides deductive and inductive reasoning. Computational advances have made possible a growing number of agent-based applications in a variety of fields. Applications range from modeling agent behavior in the stock market and supply chains, to predicting the spread of epidemics and the threat of bio-warfare, from modeling consumer behavior to understanding the fall of ancient civilizations, to name a few. This tutorial describes the theoretical and practical foundations of ABMS, identifies toolkits and methods for developing ABMS models, and provides some thoughts on the relationship between ABMS and traditional modeling techniques.

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 (North and Macal, in press). 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. The goals of this tutorial are to show how ABMS is:

- Useful: Why ABMS is a good and even better modeling approach 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. Some practical applications of ABMS are described, ABMS toolkits are presented, and ABMS development approaches are discussed. Several ABMS examples are demonstrated throughout the tutorial.

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 primitive reactive decision rules to complex adaptive intelligence. Others insist that a component's behavior must be adaptive in order for it to be considered an agent (Mellouli et al. 2003); the agent label is reserved for components that can in some sense learn from their environments and change their behaviors in response. 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 of the component to make independent decisions. This requires agents to be active rather than purely passive.

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

- An agent is identifiable, a discrete 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 situated, living in an environment with which it interacts with other agents. Agents have protocols for interaction with other agents, such as communication protocols, and the capability to respond to the environment. Agents have the ability to recognize and distinguish the traits of other agents.
- An agent is goal-directed, having goals to achieve (not necessarily objectives to maximize) with respect to its behaviors.
- An agent is autonomous and self-directed. An agent can function independently in its environment and in its dealings with other agents, at least over a limited range of situations.
- An agent is flexible, and has the ability to learn and adapt its behaviors over time based on experience. This requires some form of memory. An agent may have rules that modify its rules of behavior.

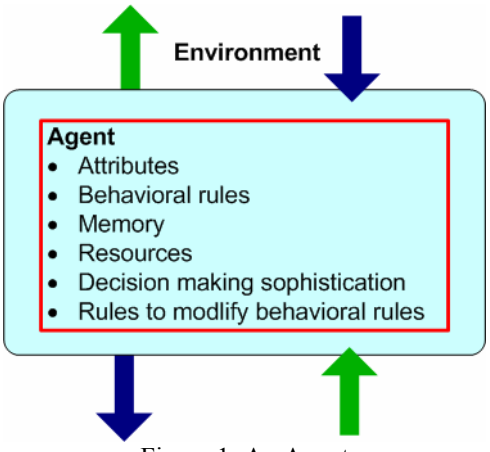


Figure 1: An Agent

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. Behavioral rules 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 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. The diverse nature of agents makes agent modeling particularly interesting!

Agent-based modeling is known by many names. ABM (agent-based modeling), ABS (agent-based systems), and IBM (individual-based modeling) are all widely-used acronyms, but "ABMS" will be used throughout this discussion. The term "agent" has connotations other than ABMS as well. ABMS agents are different from the typical agents found in mobile agent systems. Mobile agents are light-weight software proxies that perform various functions for users and to some extent can behave autonomously. ABMS is not the same as object-oriented simulation, but the object-oriented paradigm is a useful basis for agent modeling, as an agent can be considered a self-directed object with the additional capability of action choice. For this reason, large-scale agent-based modeling toolkits are almost universally object-oriented.

ABMS has strong roots in the fields of multi-agent systems (MAS) and robotics from the field of artificial intelligence (AI). But ABMS is not only tied to designing and understanding "artificial" agents. Its main roots are in modeling human social 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 structure.

2.2 The Need for Agent Based Modeling

Agent-based modeling is becoming widespread. Why? The answer is because we live in an increasingly complex world. First of all, the systems that we need to analyze and model are becoming more complex in terms of their interdependencies. This means that the traditional modeling tools are not as applicable as they once were. An example application area is the deregulation of the electric power industry. Interdependencies among infrastructures (electric power, natural gas, transportation, petroleum, water, telecommunications, etc.) are becoming the focus public attention as these systems approach their design limits and suffer regular breakdowns. Second, some systems have always been too complex for us to adequately model. For example, 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 take a more realistic view of these systems through ABMS. Third, data are becoming organized into databases at finer levels of granularity. Micro-data can now support micro-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. These observations lead us to conclude that our traditional modeling tools are not adequate,

and we need to search for new approaches that are more applicable to today's world.

2.3 Background on ABMS

ABMS has connections to many other fields including complexity science, systems science, Systems Dynamics, computer science, management science, the social sciences in general, 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.

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). CAS properties are: (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 framework for designing agent-based models. It should also be noted that Holland also developed Genetic Algorithms (GA) in his research on CAS. GAs are generic search procedures based on the mechanics of genetics and natural selection and are one of the bases for swarm optimization algorithms.

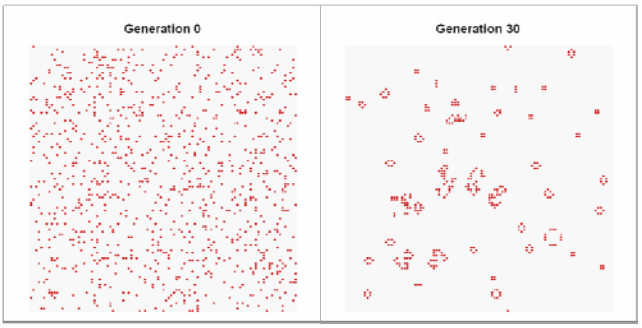
2.3.1 Simple Rules Result in Emergent Organization and Complex Behaviors

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 (1994), 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 eventually 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 snapshots from a GOL simulation. Initially, On cells are distributed randomly. After several updates of all cells in the grid, distinctive patterns emerge, and in some cases these patterns can sustain themselves indefinitely throughout the simulation. 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.



(a) Initial random layout (b) after 30 updates

Figure 2: Game of Life Simulation

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 immedi-

ate 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:

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

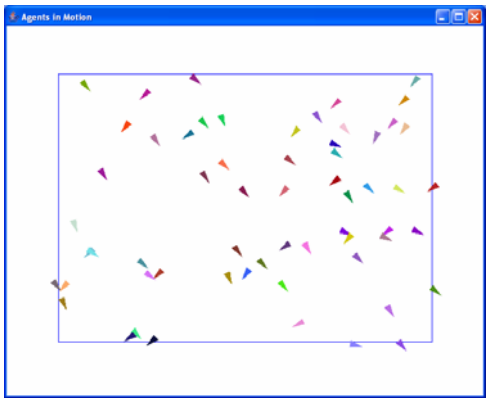
Wolfram has demonstrated that varying the rules in cellular automata produces a surprising range of emergent patterns, and these patterns correspond directly to a wide range of algorithms and logic systems (Wolfram 2002). Wolfram contends that simple rules can be used to understand much of the complexity observed in the real world.

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. Natural systems are able to not only survive, but also to adapt and become better suited to their environment, in effect optimizing their behavior over time. 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? Swarm intelligence has inspired practical optimization techniques, such as ant colony optimization that have been used to solve practical scheduling and routing problems (Bonabeau et al. 1999).

The Boids simulation is a good example of how simple rules lead to emergent and seemingly organized system behavior reminiscent of schooling or flocking behavior in fish or birds (Reynolds 2005). 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 some distance measure. Even with only these three simple rules applied at the individual agent level and only to the agents in its neighborhood, the agents' behavior begins to appear coordinated, and a leaderless flock emerges (Figure 3).



(a) Initial random configuration

(b) After 500 updates

Figure 3: Boids Simulation

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:

- How much do we know about credibly modeling people's behavior?
- How much do we know about modeling human social interaction?

These 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.

Thomas Schelling is credited with developing the first social agent-based simulation in which agents represent people and agent interactions represent a socially relevant process (Schelling 1971, 1978). Schelling applied notions of cellular automata to study housing segregation patterns. He posed the question, "is it possible to get highly segregated settlement patterns even if most individuals are, in fact, color-blind?" The Schelling model demonstrated that ghettos can develop spontaneously. Interpreted more generally, Schelling showed that patterns can emerge that are not necessarily implied or even consistent with the objectives of the individual agents. This was an important observation that spurred interest and gave direction to the field of ABMS. (Note, Schelling's initial models were not even done with the aid of a computer; agents were represented as coins moving on a checkerboard). 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 society. Emergent processes were observed including death, disease, trade, wealth, sex and reproduction, culture, conflict and war, and externalities such as pollution.

Agent-based modeling is also used in economics. Some of the classical assumptions of standard microeconomic 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. Economic agents are homogeneous, that is, agents have identical characteristics and rules of behavior,
3. There are decreasing returns to scale from economic processes, decreasing the marginal utility, decreasing the marginal productivity, etc., and
4. The long-run equilibrium state of the system is the primary information of interest.

Each of these assumptions is relaxed in ABMS applications to economic systems. First, do organizations and individuals really optimize? Herbert Simon, a Nobel Laureate who pioneered the field of artificial intelligence, developed the notion of "satisficing" to describe what he observed people and organizations doing 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 in determining people's actual economic behavior (Smith 1989). Second, that agent diversity 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, Arthur has identified "positive feedback loops" and "increasing returns" as the underlying dynamic processes of rapid exponential growth in economics (Arthur et al. 1997). Positive feedback loops 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).

Anthropologists are also 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 help understand the social and cultural factors responsible for the disappearance of the Anasazi in the southwestern U.S. (Koehler et al. 2005) and the fall of the ancient Mesopotamian civilization (Christiansen and Altaweel 2004).

Agent-based modeling is also being used in sociology. Macy and Willer (2002) review the agent-modeling approach as the basis for modeling social life as interactions among adaptive agents who influence one another in response to the influences they receive. A recent issue of the American Journal of Sociology is devoted largely to agent-based modeling in sociology (Gilbert and Abbot 2005) and social science computing is becoming a subfield (Sallach and Macal 2001). Agent-based modeling is being used in political science as well. For example, Cederman (2002) is using agent-based modeling to understand the basic processes involved in national identity and state formation.

Cognitive science has had its own notion of agency, and social cognitive science is extending these ideas to social settings (Bandura 2001). Cognitive scientists are developing agent-based models of emotion, cognition, and social behavior based on the notion that a person's emotional state impacts their behavior as well as their social interactions (Gratch and Marsella 2001). The goal is to create synthetic agents who embody the nuanced interplay between emotion, cognition and social behavior.

In addition to the social sciences, ABMS is also being used in the physical and biological sciences as an adjunct to laboratory and theoretical research. In the physical sciences, ABMS is being used to model the possible emergent structures resulting from molecular self-assembly (Troisi et al. 2005). In biology, agent-based modeling is being used to model bacterial behavior and interaction at multiple-scales (Emonet et al. 2005) and the self-organization of bacterial colonies (Krawczyk et al. 2003).

2.3.3 Topologies as a Basis for Social Interaction

Cellular automata represent agent interaction patterns and available local information by using a grid or lattice and the cells immediately surrounding an agent as the neighborhood. Other agent interaction topologies, such as networks, allow an agent's neighborhood to be defined more generally and flexibly. Networks 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 (Wasserman and Faust 1994). 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 growth and change of real-world networks (Barabási 2002). In particular "small world" or scale-free networks have been discovered in a wide range of settings, such as the World Wide Web, membership in corporate executive boards, and ecological habitats.

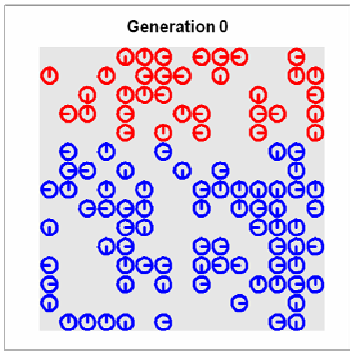
Dynamic network analysis is a new field that incorporates the mechanisms of network growth and change based on agent interaction processes (NRC 2003). Understanding the agent rules that govern how networks are structured and grow, how quickly information is communicated through networks, and the kinds of relationships that networks embody are important aspects of modeling agents and of "network ABMS."

Some of the important modeling techniques of ABMS have their origins in models of physical systems, and network modeling is an example. The Ising system, for example, is a model of imitative behavior in which individuals modify their behaviors to conform to the behavior of other individuals in their vicinity (Callen and Shapero 1974). Originally developed by Ernest Ising in 1925, the Ising model was used to study phase transitions in solids, for example how individual atoms become aligned in a material causing it to become magnetized. In the 1990's, social scientists adapted the Ising model to study social processes. Opinion formation can be modeled by an Ising approach if the lattice assumption is relaxed so that individuals are free to interact with neighbors defined by a social network or by spatial proximity. The key behaviors of such a model are the sudden phase transitions that can occur without warning, signifying rapid opinion change. The Ising model has been used to model diffusion of new ideas, fads, friendship formation, social communication, and the spread of disease and epidemics.

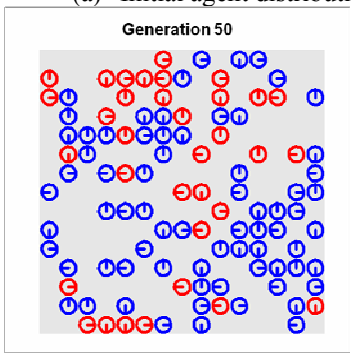
2.3.4 Generative Social Science

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). Young has investigated how larger social structures, social norms and institutions, arise from micro-level interactions between individuals (Young 1998). 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 2005, Sallach 2003). Progress in generative social sciences will be a fertile basis for developing ABMS in the future.

We close this section with a simple ABMS example of social influence. In this simulation there are two types of agents, red and blue agents. Initially, there are twice as many blue agents as red agents. Each side tries to convince the other side of its position. An agent surrounded by more agents of one color than another, adopts the position of the dominant agent. There is one stipulation: each red agent is twice as convincing as each blue agent. An agent need only be surrounded by half as many red agents as blue agents to be just as likely to adopt red's position. Thus, the total convincing power of the red and blue agents is equal. The simulation suggests some interesting questions: will one type of agent have its opinion dominate or will there be a stable mix of positions in the long run? The simulation is shown in Figure 4.



(a) Initial agent distribution



(b) After 50 generations

Figure 4: Social Influence Simulation

3 ABMS APPLICATIONS

Practical agent-based modeling and simulation is actively being applied in many areas (Table 1). ABMS applications range across a continuum, from small, elegant, minimalist academic 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 ABMS 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 and are designed to answer real-world policy questions. These models include real data and have passed some validation tests to establish credibility. We will explore a couple of agent simulations in more detail to illustrate the approach of agent modeling.

Table 1: Agent-based Modeling Applications

Business and Organizations • Manufacturing • Consumer markets • Supply chains • Insurance Economics • Artificial financial markets • Trade networks Infrastructure • Electric power markets • Hydrogen economy • Transportation Crowds • Human movement • Evacuation modeling	Society and Culture • Ancient civilizations • Civil disobedience Terrorism • Social determinants • Organizational networks Military • Command & control • Force-on-force Biology • Ecology • Animal group behavior • Cell behavior • Sub cellular molecular behavior

3.1 An Electric Power Market ABMS Application

EMCAS (Electricity Market Complex Adaptive System) is an agent-based simulation model of the electric power market designed to investigate market restructuring and deregulation and to understand implications of a competitive market on electricity prices, availability, and reliability. The EMCAS model is described elsewhere from various perspectives including the benefits of agent-based modeling for deregulated electric power markets (North et al. 2002, Koratarov 2004). The agents in EMCAS represent the participants in the restructured electricity market (Figure 5). Different types of agents capture the heterogeneity of restructured markets, including generation companies, demand companies, transmission companies, distribution companies, independent system operators, consumers, and regulators. The agents perform diverse tasks using specialized decision rules. Agents learn about the market response to their price-quantity bids, infer the strategies of their competitors, and adapt their actions accordingly. Agents continually explore new strategies in response to dynamic supply and demand forces and identify strategies that perform better. Generating company agents engage in price discovery, learning how they can influence the market through their actions to increase their utility, defined as a combination of profits and market share. As a prerequisite for a realistic model of the electric power market, EMCAS agents interact with and are constrained by the physics of the electric power grid which is represented at the individual bus level for a regional transmission network.

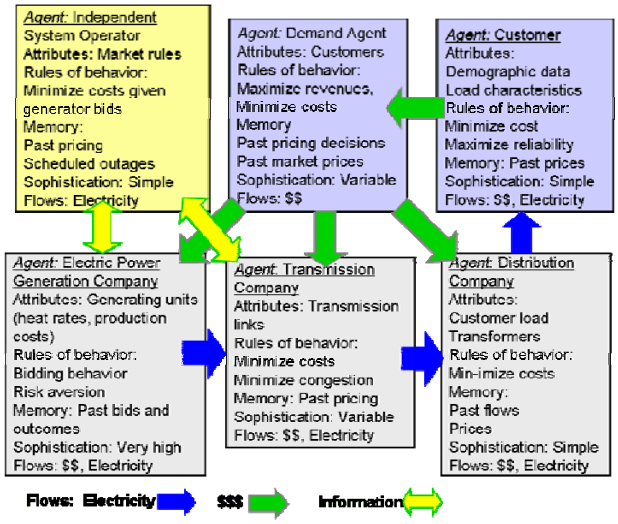


Figure 5: Agents in EMCAS Electric Power Market Model

3.2 An Supply Chain ABMS Application

An agent-based model of a supply chain illustrates the ABMS approach and shows how a simple model can reveal important insights. We recast Sterman's "Beer Game" simulation from its original Systems Dynamics implementation (Sterman 1989) to an agent-based simulation. The supply chain consists of four stages: factories, distributors, wholesalers, and retailers who respond to customers' demand. Various simplifying assumptions are made such as: there is only one commodity, no transformation of goods is made and no assembly of materials into products is required. The flows of goods and information in the form of orders between stages (agents) as well as physical shipments are included in the model, but the flows of payments and the additional complexities of pricing, negotiation, and financial accounting that this would entail are not included. However, these aspects of supply chain agent behavior could easily be incorporated in the agent-based version of the supply chain model.

Supply chain agents consist of the customer, retailer, wholesaler, distributor, and manufacturer (Figure 6). Each period, supply chain agents execute behaviors:

1. The customer places an order with the retailer.
2. The retailer fills the order immediately from its respective inventory if it has enough inventory in stock (if the retailer runs out of stock, the customer's order is placed on backorder and filled when stock is replenished).
3. The retailer receives a shipment from the wholesaler in response to previous orders. The retailer then decides how much to order from the wholesaler based on an "ordering rule." The ordering decision is based in part on how much the retailer expects customer demand will be in the future. The retailer estimates future customer demand using a "demand forecasting" rule. The retailer then orders items from the wholesaler to cover expected demand and any shortages relative to explicit inventory or pipeline goals.
4. Similarly, each wholesaler receives a shipment from the distributor, forecasts future demand by the retailer, and places an order with the distributor. This process continues up the chain to the factory who decides on how much to put into new production.

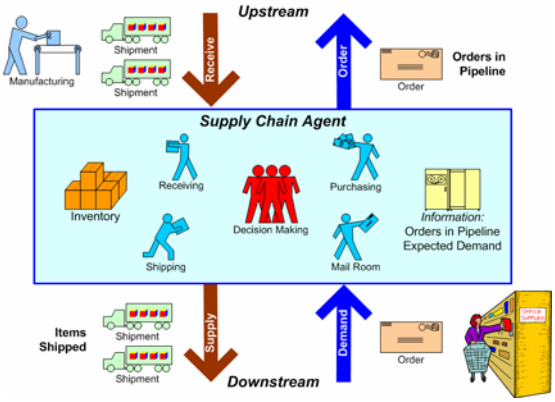


Figure 6: The World of the Supply Chain Agent

The goal of the agents in the model is to manage their inventory in such a way as to minimize their costs through judicious decisions based on how much to order each period. When inventories are too low and there is a danger of running out of stock, agents order more; when inventories are too large and agents incur high inventory holding costs, agents order less. Each agent incurs a cost when holding stock in inventory, the inventory holding charge. Agents also incur a backorder charge when they receive an order and cannot immediately meet that order because they have no stock. Each agent strikes a delicate balance between having too much inventory, which runs up inventory holding costs, and too little inventory, which puts the agent at a greater risk of running out of stock and incurring excessive backorder charges.

In this example, supply chain agents only have access to local information. No agent has a global view of the supply chain or is in a position to optimize the system as a whole. Agents adopt decision rules that only this local information in making their decisions.

The results of the agent-based supply chain model exactly duplicate the results from Sterman's original Beer Game Simulation. The "Bull-Whip" effect, observed in real supply chains, is observed even in this highly simplified supply chain model. This simple agent-based model is a very useful foundation for more realistic models of supply chains, such as models based on supply network topologies and alternative agent decision rules. Several agent models of supply chains have been developed with various enhancements such as non-local information (Macal 2003).

4 HOW TO DO ABMS

One goes about building an agent model in much the same way that one builds any type of model or simulation. First, identify the purpose of the model, the questions the model is intended to answer and the potential users. Next, systematically analyze the system under study, identifying components and component interactions, relevant data sources, and so on. The usual steps of model building apply to agent-based modeling as well. See Law and Kelton (2000) for an excellent description of good simulation model building practice.

Agent-based modeling brings with it a few unique twists owing to the fact that ABMS takes the agent perspective, first and foremost, in contrast to the process-based perspective that is the traditional hallmark of simulation modeling. In addition to the standard model building tasks, practical ABMS requires one to:

- Identify the agents and get a theory of agent behavior,
- Identify the agent relationships and get a theory of agent interaction,
- Get an ABMS platform(s) and an ABMS model development strategy,
- Get the requisite agent-related data,
- Validate the agent behavior models (in addition to the model as a whole), and
- Run the model and analyze the output from the standpoint of linking the micro-scale behaviors of the agents to the macro-scale behaviors of the system.

We discuss a few of these aspects of ABMS in this section.

4.1 Discovering Agents

Identifying agents, accurately specifying their behaviors, and appropriately representing agent interactions are the keys to developing useful agent models. Agents are generally the decision-makers in a system. These include traditional decision-makers such as managers as well as non-traditional decision-makers such as complex computer systems that have their own behaviors. Even groups can be considered agents for some modeling purposes.

Once the agents are defined, discovering the key agent behaviors is the next challenge. How can agent behaviors be discovered? First, get a theory of agent behavior. For example, one may begin with a normative model in which agents are attempting to optimize and use this model as a starting point for developing a simpler and more descriptive heuristic model of behavior. One may also begin with a behavioral model if some applicable behavioral theory is available and seems appropriate. For example, numerous theories abound for modeling consumer shopping behavior based on empirical studies. Alternatively, a number of formal logic frameworks have been developed in order to reason about rational agents and these can serve as the basis for agent models. Frameworks such as BDI (Belief-Desire-Intent) (Rao and Georgeff 1991) and BOD (Behavior-Oriented Design) (Bryson 2002) used combined modal and temporal logics as the basis for reactive plans and action selection.

When the behaviors of individuals are to be the basis for agent models for existing or hypothesized systems, knowledge engineering and participatory simulation are useful techniques to employ. Knowledge engineering consists of a collection of techniques for eliciting and organizing the knowledge of experts while accounting for reporting errors and situational biases. Knowledge engineering uses structured interviews to elicit information on agent behaviors.

Participatory ABMS combines the agent modeling paradigm with ideas from organization theory to specify goal-driven simulations that consist entirely of human participants playing roles, akin to gaming, but with much more structure. Participatory simulations are very useful to set up as a prelude to developing an agent model. With proper structure, instruction, and discipline, people in participatory ABMS can reveal much information about agent behaviors, such as: How much information are people able to process in the given amount of time for making decisions?, What key factors and indicators do people consider in making their decisions?, How do people's past experiences enter into their decision-making process?, and Which strategies do people formulate that are most effective? Participatory agent modeling can be used to develop insights into and validate plausible agent behavior models, demonstrate agent modeling concepts to stakeholders, and test ideas on agent behavior in contrived situations.

4.2 ABMS Development Tools

Agent modeling can be done in the small, on the desktop,
or in the large, using large-scale cluster computers, or at any scale in-between.

4.2.1 Desktop ABMS

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 ABMS can be used to learn how to do agent modeling, test agent modeling design concepts and perform many types of serious modeling and analysis. Desktop tools include general spreadsheets and computational mathematics systems. One of the key features of desktop ABMS systems is that they are interpreted environments, requiring no compilation or linking steps that are required by general programming languages or agent-based modeling toolkits.

Spreadsheets are natural tools for modelers already familiar with spreadsheets and macro-programming. They are an excellent starting point for developing desktop agent models. Computational mathematics systems (CMS) such as Mathematica (Wolfram Inc. 2005) and MATLAB (MathWorks 2005) can also be used to build agent models, although no dedicated agent facilities are currently provided by these systems. CMS tend to offer better mathematical modeling libraries, visualization, statistical analysis, and database routines than conventional spreadsheets. However, these environments are harder to learn for those not already familiar with them. Desktop ABMS systems are generally limited to handle agents that number in the range of dozens to hundreds.

4.2.2 Large-scale ABMS

Large-scale ABMS extends agent modeling beyond simple desktop environments and allows thousands to millions of agents to engage in sophisticated interchanges. Large-scale agent modeling is usually done with computer-based agent simulation environments. These environments support several features specific to agent modeling including the availability of a time scheduler, the availability of communications mechanisms, the availability of flexible interaction topologies, a range of architectural choices, facilities for storing and displaying agent states, large-scale development support and in some cases special topic support. Large-scale agent models can be built and run on desktop computers in addition to higher-performance computing systems. However, large-scale agent models generally require more advanced skills and more development resources than desktop environments. Several standards for agent software have influenced agent-based toolkit devel-

opment including the Foundation for Intelligent Physical Agents' (FIPA 2005) architecture specifications, the Object Management Group Agent Platform Special Interest Group, Agent UML (OMG 2005), and the Knowledge-able Agent-oriented System architecture (KAoS) (Bradshaw 1997).

Thanks to substantial public research and development investments, many ABMS software environments are now freely available. These include Repast, Swarm, NetLogo and MASON, among others. Proprietary toolkits are also available. A recent review and comparison of Java-based agent modeling toolkits is by Tobias and Hoffman (2004).

The REcursive Porous Agent Simulation Toolkit (Repast) is the leading free and open source large-scale agent-based modeling and simulation library. Repast seeks to support the development of extremely flexible models of agents with an emphasis on social interactions. Users build simulations by incorporating Repast library components into their own programs or by using the visual Repast for Python Scripting environment (Collier, Howe et al. 2003). More information on Repast, as well as free downloads, can be found at the Repast home page (Repast 2005). Repast is maintained by the Repast Organization for Architecture and Design (ROAD). According to ROAD:

> Our goal with Repast is to move beyond the representation of agents as discrete, self-contained entities in favor of a view of social actors as permeable, interleaved and mutually defining, with cascading and recombinant motives. We intend to support the modeling of belief systems, agents, organizations, and institutions as recursive social constructions. The fuller goal of the toolkit is to allow situated histories to be replayed with altered assumptions. To achieve this goal, it will be necessary for Repast to provide a feast of advanced features, and it is toward that objective that we work. (Repast 2005)

Repast has been used extensively in social simulation applications (North and Macal 2005). There are three production versions of Repast, namely Repast for Python (Repast Py), Repast for Java (Repast J) and Repast for the Microsoft .NET framework (Repast .NET).

Repast Py is a cross-platform visual model construction system that allows users to build models using a graphical user interface and write agent behaviors using Python scripting. All of the features of the Repast system are available in Repast Py, but Repast Py is designed for rapid development of prototype agent models. Repast Py models can be automatically exported to Repast J for large-scale model development.

Repast J is a pure Java modeling environment to support the development of large-scale agent models. It includes a variety of features such as a fully concurrent discrete event scheduler, a model visualization environment, integration with geographical information systems for modeling agents on real maps, and adaptive behavioral tools such as neural networks and genetic algorithms. The Repast J interface running an implementation of a basic social network model is shown in Figure 7. The lines in the lower display window indicate ongoing interactions between connected agents. Agents form and maintain ties over time based on the payoffs they receive from interactions along the links.

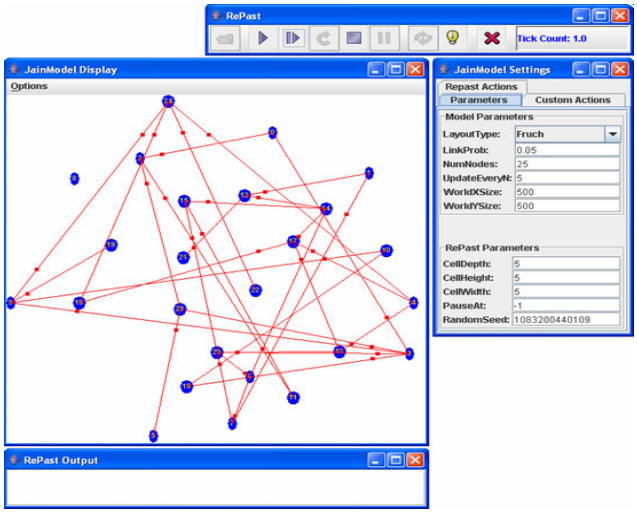


Figure 7: The Repast J "Jain" Network Model

Repast .NET is a pure C# modeling environment that brings all of the features of Repast J to the Microsoft .NET framework. Repast .NET models can be written in any language supported by the Microsoft .NET framework such as Managed C++, C#, Visual Basic or even Managed Lisp or Managed Prolog10.1.

Repast has a sophisticated scheduler that can do both time step and discrete event simulation as well as maintain both global and local views of time. Repast allows a wide range of communications mechanisms to be used with all of the major interaction topologies. Repast includes a full set of tools for storing and displaying agent states and works well with several large-scale development environments. Repast also includes tools for automated integration with both commercial and free open source Geographical Information Systems (GIS) using both Lagrangian and Eulerian representations. The commercial GIS integration includes automated connectivity to the widely used ESRI ArcGIS geographical information system. Furthermore, since Repast is based on the Java language, the Microsoft .NET framework and Python scripting, it is fully object-oriented.

Swarm was the first ABMS software development environment launched in 1994 by Chris Langton at the Santa Fe Institute. Swarm is a free and open source software library (Minar, Burkhart et al. 1996) and is currently main-

tained by the Swarm Development Group (SDG). Swarm seeks to create a shared simulation platform for ABMS and to facilitate the development of a wide range of models. Users build simulations by incorporating Swarm library components into their programs. More information on Swarm, and free downloads, can be found at the SDG home page (SDG 2005). From the Swarm developers:

> Swarm is a set of libraries that facilitate implementation of agent-based models. Swarm's inspiration comes from the field of Artificial Life. Artificial Life is an approach to studying biological systems that attempts to infer mechanism from biological phenomena, using the elaboration, refinement, and generalization of these mechanisms to identify unifying dynamical properties of biological systems… Chris Langton initiated the Swarm project in 1994 at the Santa Fe Institute. The first version was available by 1996, and since then it has evolved to serve not only researchers in biology, but also anthropology, computer science, defense, ecology, economics, geography, industry, and political science.

The Swarm simulation system has two fundamental components. The core component runs general-purpose simulation code written in Objective-C, Tcl/Tk, and Java. This component handles most of the "behind the scenes" details. The external wrapper components run user-specific simulation code written in either Objective-C or Java. These components handle most of the "center stage" work. The Swarm interface uses probes to display and edit the properties of agents. Probes are normally activated by clicking on an agent in one of the display windows.

Unlike Repast, the Swarm scheduler only supports time step scheduling, but it does so at a high level of resolution. The Swarm scheduler can maintain global and local time schedules. Swarm supports a full set of communications mechanisms and can model all of the major interaction topologies. Swarm includes a good set of tools for storing and displaying agent states. Since Swarm is based on a combination of Java and Objective-C, it is object-oriented. However, this mixture of languages causes Swarm to have difficulties with integration into some of the large-scale development environments, such as Eclipse. Swarm has support for GIS through the Kenge library.

NetLogo is another cross-platform multi-agent programmable modeling environment that has extensive use and support (NetLogo 2005). Originally based on the StarLogo system, NetLogo accommodates agent systems having a combination of live and software agent participants.

Proprietary toolkits are also commonly used for agent modeling. These toolkits have the advantage of being custom designed for specific uses. They also can be modified without concern for maintaining synchronization with the public version of the toolkit. However, developing and maintaining such a toolkit can require substantial resources and a long-term organizational commitment. Furthermore, such toolkits lack the community or commercial support often found with publicly available toolkits such as online and in-person help, code sharing, and targeted conferences and user meetings. The lack of experienced developers is also an issue faced with proprietary toolkits.

4.3 The ABMS Modeling Lifecycle

Developing an agent-based simulation is part of the more general software and model development process. The development timeline typically has several highly interleaved stages. The concept development and articulation stage defines the project goals. The requirements definition stage makes the goals specific. The design stage defines the model structure and function. The implementation stage builds model using the design. The operationalization stage puts the model into use. In practice, successful ABMS projects typically iterate over these stages several times with more detailed models resulting from each iteration. Successful projects also begin small using one or more of the desktop ABMS tools and then grow into the larger-scale ABMS toolkits in stages.

5 ABMS AND TRADITIONAL M&S TECHNIQUES

Agent-based modeling can either provide an overarching framework for model components based on other modeling techniques, or it can provide agent models that are embedded into larger systems. All of the modeling approaches currently in use were originally developed to address specific types of problems, for example, optimization for finding the best solution, discrete-event simulation for understanding the effects of uncertainty in a process, and Systems Dynamics for understanding system interconnectedness. Using agent-based models in combination with other techniques is an approach that we call "model blending." Here are some examples of situations where it might be desirable to combine ABMS with other modeling techniques.

Systems Dynamics (SD) is extremely useful for identifying the important variables and causal linkages in a system and for structuring many aspects of model development. Many ABMS modeling projects can benefit greatly by beginning with a systematic identification and analysis of the important variables in the system and their causal relationships as in SD.

Discrete event simulation (DES) offers methods for taking a process view of the system and for dealing with stochastic uncertainty. To the extent that agents are engaged in processes and move through a system, DES techniques can be useful in developing an agent-based model.

Often, it is not realistic to develop a comprehensive agent model in which all the causal factors and relationships leading to an agent's decisions can be fully specified. The alternative is to use statistical estimation techniques for estimating agent decision rules. These are essentially simple models relating situations the agents find themselves in to their actions. It is also possible to go from complex agent models to simpler statistical relationships by identifying the key variables that govern most of the agent behaviors. Sophisticated statistical techniques such as principal component analysis can be used for this purpose.

In traditional risk analysis, there is little apparent connection to agent-based modeling. However, sometimes risk can be considered an emergent property of a system as a whole. A comprehensive, system-wide assessment of the causal factors that lead to risk throughout the system can be addressed. Agent modeling is a natural approach to representing the diverse characteristics and decision making behaviors of companies or individuals that comprise the system or industry. For example, Insurance World is an agent-based risk assessment model of the reinsurance industry (MacKenzie 2002).

Optimization techniques are used to model optimal (normative) individual and organizational decision making. Agent-based modeling can be used in conjunction with agents that optimize on an individual basis or, as described above, agent-based models can be the basis for swarm optimization techniques in which individual agent actions lead to near-optimal system states.

In summary, agent modeling represents a new frontier in creatively combining novel and traditional modeling approaches in ways that have not been previously possible.

6 ABMS RESOURCES

Many resources are available for learning more about ABMS and several ABMS communities support development of toolkits and applications. NAACSOS, the North American Association for Computational Social and Organizational Science, is a recently formed national professional organization devoted to furthering computational social sciences and ABMS <http://www.naacsos.org>. Annual conferences include the NAACSOS conference <http://www.casos.cs.cmu.edu/> that covers social and organizational computation of all kinds, the Agent 200X conferences <http://www.agent2005.anl.gov> that focus on agent modeling in the social sciences and include tracks on toolkits, computational social theory, and applications, SwarmFest <http://www.sdg.org> that is concerned with all aspects of agent-based modeling, especially Swarm-based applications, and the Lake Arrowhead Conference <http://www.hcs.ucla.edu/arrowhead.htm> that focuses on social theory and modeling of human complex systems. Other major conferences have ABMS-related tacks or sponsor specialty conferences including the IEEE, INFORMS, and various simulation conferences (WSC, SCSC, Multi-Conferences), for example, the 2005 IEEE Swarm Intelligence Symposium <http://www.ieeeswarm.org>. Various organizations are devoted to advancing complexity sciences such as the Santa Fe Institute <http://www.santafe.edu> and the Center for the Study of Complex Systems (CSCS) at the University of Michigan <http://www.cscs.umich.edu>. Some organizations are devoted to furthering ABMS in theory, applications, methods and/or education such as Argonne's Center for Complex Adaptive Agent Systems Simulation <http://www.cas.anl.gov>, the Center for Computational Analysis of Social and Organizational Systems (CASOS) at Carnegie Mellon <http://www.casos.cs.cmu.edu/> and the Center for Social Complexity <http://socialcomplexity.gmu.edu/> at George Mason.

7 WHY AND WHEN ABMS

The scope of ABMS continues to expand well beyond its origins in biological systems. ABMS has found application in solving practical problems and advancing research agendas. Why do we do agent-based modeling? The agent representation allows us to ask questions such as "what is the effect of agent diversity on the evolution of the system?", "do certain types of agents dominate?", and "does the system evolve toward a stable mix of agent types?" Agent simulation can be used to study how patterns and organizations emerge and to discover how system-level structures form that are not apparent from the behaviors of individual agents.

Situations for which agent-based modeling can offer distinct advantages to traditional modeling approaches, reveal new insights, and answer long-standing questions are becoming better understood every day. When is it beneficial to think in terms of agents?

- 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 relationships with other agents, and agent relationships form and dissolve

- When it is important that agents form 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
- When process structural change needs to be a result of the model, rather than a model input.

ACKNOWLEDGMENTS

The authors would like to thank David Sallach, Rob Axtell, Eric Bonabeau, Rick Riolo, and Kate Coronges for recent helpful discussions on agent modeling. This work is sponsored by the U.S. Department of Energy under contract W-31-109-ENG-38.

REFERENCES

Arthur, W. B. et al. Eds. 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.

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.

Bandura, A. 2001. Social cognitive theory: an agentic perspective, *Annual Review of Psychology* 52:1-26.

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

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 *Proc. National Academy of Sciences* 99(3): 7280-7287.

Bradshaw, J. 1997. *An introduction to software agents,* Menlo Park, CA: AAAI Press:.

Bryson, J. 2002. The behavior-oriented design of modular agent intelligence: a practical guide to behavior-oriented design (BOD), In *Proc. of Agent Technology and Software Engineering (AgeS 02)*, Ed., Jörg P. Müller, Springer, Nov. 27.

Callen, E. and Shapero, D. 1974. A theory of social imitation, *Physics Today* 27: 23-28.

Casti, J. 1994. *Complexification*, Harper Collins: New York.

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

Cederman, Lars-Erik. 2002. Endogenizing geopolitical boundaries with agent-based modeling, *Proc. National Academy of Sciences* 99 (suppl. 3):7796-7303.

Christiansen, J. H. and M. Altaweel. 2004. Simulation of natural and social process interactions in Bronze Age Mesopotamian settlement systems, presented at *Society for American Anthropology 69th Annual Meeting*, Montreal, Canada.

Collier, N., T. Howe, et al. 2003. Onward and upward: the transition to Repast 2.0. in *Proc. First Annual North American Association for Computational Social and Organizational Science Conference*, Pittsburgh, PA.

Emonet, T., C. M. Macal, M. J. North, C. E. Wickersham and P. Cluzel. 2005. AgentCell: a digital single-cell assay for bacterial chemotaxis, *Bioinformatics* 21(11):2714-2721.

Epstein, Joshua M. 2005. Remarks on the foundations of agent-based generative social science, in *Handbook on Computational Economics II*, Eds., K. Judd and L. Tesfatsion, North Holland Press.

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

FIPA (Foundation for Intelligent Physical Agents). 2005. FIPA Home Page, <http://www.fipa.org/>.

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

Gilbert, N. and A. Abbot. 2005. Introduction to special issue: social science computation, *American Journal of Sociology* 110(4):859-863.

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

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

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

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

Krawczyk, K., W. Dzwinel, and D. Yuen. 2003. Nonlinear development of bacterial colony modeled with cellular automata and agent objects, *Int'l. Journal of Modern Physics C*, 14(10):1385-1404.

Law, A. M. and D. W. Kelton. 2000. *Simulation modeling and analysis*, 3rd ed. New York: McGraw-Hill.

Macal, C. 2003. Effects of global information availability in networks of supply chain agents, in *Proc. Agent 2003: Conf. on Challenges in Social Simulation*, Eds., C. Macal, D. Sallach and M. North, Chicago, IL, Oct. 2-4, 235-252, Argonne National Laboratory.

MacKenzie, D. 2002. The science of surprise, *Discover*, 59-62.

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

MathWorks. 2005. MATLAB home page, <http://www.mathworks.com>.

Mellouli, S., G. Mineau, et al. 2003. Laying the foundations for an agent modelling methodology for fault-tolerant multi-agent systems, in *Fourth International Workshop Engineering Societies in the Agents World*, Imperial College London, UK.

Minar, N., R. Burkhart, et al. 1996. The Swarm simulation system, a toolkit for building multi-agent simulations, <http://www.santafe.edu/projects/swarm/overview/overview.html>.

NetLogo. 2005. NetLogo home page, <http://http://ccl.northwestern.edu/netlogo>.

North, M., G. Conzelmann, V. Koritarov, C. Macal, P. Thimmapuram and T. Veselka. 2002. E–laboratories: agent–based modeling of electricity markets, *2002 American Power Conference*, Chicago, IL, Apr. 15-17.

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

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.

OMG (Object Management Group). 2005. Object Management Group home page, <http://www.omg.org>.

Rao, A. S. and M. P. Georgeff. 1999. Modeling agents within a BDI-architecture, In *Proc. International Conference on Principles of Knowledge Representation and Reasoning (KR)*, Eds., R. Fikes and E. Sandewall, Cambridge, MA: Morgan Kaufmann.

Repast. 2005. Repast home page, <http://repast.sourceforge.nett/>.

Reynolds, Craig. 2005. Boids, <http://www.red3d.com/cwr/boidss/>.

Sallach, D. 2003. Social theory and agent architectures: prospective issues in rapid-discovery social science, *Social Science Computer Review* 21:179-195.

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

Schelling, T. C. 1971. Dynamic models of segregation, *Journal of Mathematical Sociology* 1: 143-186.

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

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

Smith, V. 1989. Theory, experiments and economics, *Journal of Economic Perspectives*, 3(1):151-169.

SDG (Swarm Development Group). 2005. Swarm Development Group home page, <http://www.swarm.org>.

Sterman, John. 1989. Testing behavioral simulation models by direct experiment, *Management Science* 33(12):1572–1592.

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. <http://www.econ.iastate.edu/tesfatsi/ace.htm.

Tobias, Robert and Carole Hofmann. 2004. Evaluation of free Java-libraries for social-scientific agent based simulation, *Journal of Artificial Societies and Social Simulation*, 7(1), Jan. 31.

Troisi, A., V. Wong, and M. Ratner. 2005. An agent-based approach for modeling molecular self-organization, *Proc. National Academy of Sciences,* 102(2):255-260.

Wasserman, S. and K. Faust. 1994. *Social network analysis: methods and applications*, Cambridge, UK: Cambridge University Press.

Wolfram, S. 2002. *A new kind of science*, Wolfram Media.

Wolfram Inc. 2005. Mathematica home page, <http://www.wolfram.com>.

Young, H. P. 1998. *Individual strategy and social structure: an evolutionary theory of institutions,* Princeton, NJ: Princeton University Press.

AUTHOR BIOGRAPHIES

CHARLES M. MACAL, Ph.D., P.E., is the Director, Center for Complex Adaptive Agent Systems Simulation (CAS2), Argonne National Laboratory. He is a member of the INFORMS-Simulation Society, the Society for Computer Simulation Intl., the Systems Dynamics Society and a founding member of NAACSOS. His email address is <macal@anl.gov> and his web address is <http://www.cas.anl.gov>.

MICHAEL J. NORTH, Ph.D., M.B.A., is the Deputy Director of CAS2 at Argonne. Michael has more than 14 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 as well as degrees in business and mathematics. His email address is <north@anl.gov>.