Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

AGENT-BASED MODELING AND SIMULATION: DESKTOP ABMS

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," in addition to traditional deductive and inductive reasoning (Axelrod 1997b). 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 of shopper behavior using spreadsheets.

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, 2007). 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. Practical applications of ABMS are described, ABMS toolkits are presented, and the development of a simple agent-based model of shopper behavior using spreadsheets 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 primitive reactive decision rules to complex adaptive artificial intelligence (AI). Others 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 in some sense 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 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 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 that are of interest.
- 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 for communication, and the capability to respond to the environment. Agents have the ability to recognize and distinguish the traits of 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 relative to its goals.
- 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.

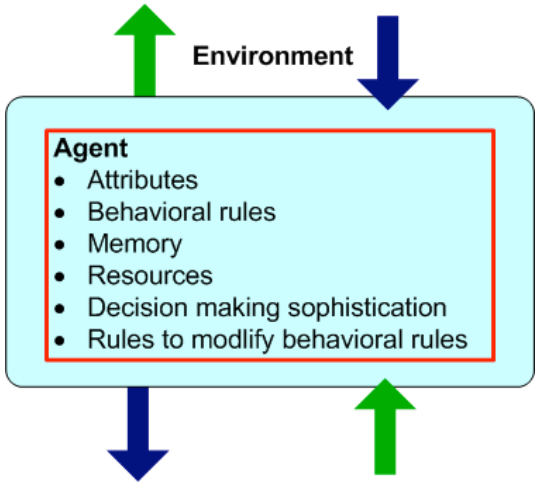


Figure 1: An Agent

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 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), 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 roam over the world-wide web and perform various functions for users and to some extent can behave autonomously.

ABMS has strong roots in the fields of multi-agent systems (MAS) and robotics from the field of AI. But ABMS is not only tied to designing and understanding "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 electric power industry. 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 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.

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. 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

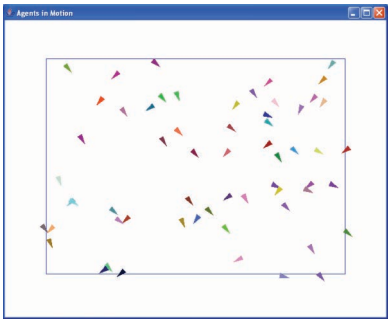
The Boids simulation is a good example of how interacting agents, characterized by simple behavioral rules, lead to emergent and seemingly organized behavior at the system level (Reynolds 2006). Agent behavior is reminiscent of schooling or flocking behavior in fish or birds. In the Boids model, each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

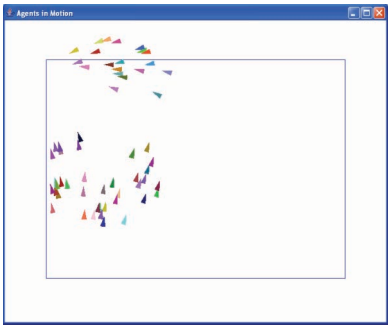
Here, nearby or local refers to agents in the immediate neighborhood of an agent as defined by the straight-line distance. Even with only these three simple rules applied at the individual agent level and only to the agents in its "neighborhood", the agents' behaviors begins to appear coordinated, and a leaderless flock emerges (Figure 2).

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: (1) sustainable patterns can emerge in systems that are completely described by simple deterministic rules based on only local information, and (2) patterns that develop can be extremely sensitive to the initial conditions.

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, effectively 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).



(a) Initial random configuration



(b) After 500 updates

Figure 2: 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: (1) how much do we know about credibly modeling people's behavior?, and (2) how much do we know about modeling human social interaction? 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.

Thomas Schelling is generally credited with developing the first social agent-based simulation in which agents represent people and agent interactions represent a socially relevant process (Schelling 1978). 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?" 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.

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 which 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 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) 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 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, "positive feedback loops" and "increasing returns" have been identified as underlying dynamic processes of rapid exponential growth in economic systems (Arthur et al. 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 help understand the social and environmental factors responsible for the disappearance of the Anasazi in the southwestern U.S. (Koehler et al. 2005) and to explain the prosperity of ancient cities in Mesopotamia (Wilkinson et al. 2007).

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. Agent-based models of "emotion, cognition, and social behavior" are being developed by cognitive scientists and others (Gratch and Marsella 2001). These synthetic social agents model the influence of emotion and cognition on social behavior (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 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 1997a). 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, Sallach 2003).

3 ABMS APPLICATIONS

Practical agent-based modeling and simulation is actively being applied in many areas (Table 1). ABS applications range from modeling agent behavior in the stock market (LeBaron 2002) and supply chains (Fang, Kimbrough et al. 2002, Macal 2004a), to predicting the spread of epidemics (Huang, Sun et al. 2004) and the threat of bio-warfare (Carley 2006), from modeling the growth and decline of ancient civilizations (Kohler, Gumerman et al. 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.

Table 1: Agent-based Modeling Applications

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 & control • Force-on-force Biology • Population dynamics • Ecological networks • Animal group behavior • Cell behavior and sub cellular processes

4 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 cluster, or in 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 the next section we describe an implementation of a spatial shopper agent model in spreadsheets.

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 is particularly well suited for artificial life projects. 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, with at least limited capabilities. 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, map, 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 et al. 2006), Swarm (SDG 2006; Minar et al. 1996), NetLogo (NetLogo 2006) 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 et al., 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.

5 SPREADSHEET SHOPPER AGENT MODEL

The Shopper Agent Model illustrates the main aspects of agent simulation using a spreadsheet environment, albeit in a highly simplified way for the purposes of introducing the components of and the structure of an ABMS. The model illustrates the use of (1) agent behaviors and, (2) the specification of agent interactions with other agents and with the environment. Developing spreadsheet models involves more than simply copying and pasting agents into a spreadsheet environment. Other functions typically found in agent-based models must also be addressed such as time management, input data setup, output results collection or logging, and run specification. The Shopper Agent Model is inspired by John Casti's SimStore model (Casti 2001).

5.1 Shopper Agent Model Concept

For simplicity, we assume each shopper is shopping for one item. The model can be easily extended to consider shoppers shopping for an arbitrary number of items if desired. Shoppers begin their trip at the front entrance of the store. The first time a shopper visits the store, they do not know where their item is located, and must mill through the store to find their item. Once they find the item, they may remember the location on their next visit, but their memories are imperfect. If a shopper does not remember the item location, they mill through the store. If a shopper knows the location of their item, they head in its general direction. Movements of shoppers are impeded by the need to avoid bumping into other shoppers and the store shelves. Once a shopper finds their item of interest, they head to the checkout counters. Shoppers have limited knowledge of where the counters are located and which counters have the shorter checkout lines until they get near the counters. Shoppers leave the store through the main exit after check out. Shoppers return to the store for another purchase trip on a periodic basis.

5.2 Shopper Agent Workbook

The shopper agent workbook consists of four spreadsheets and an Excel macro sheet. The four spreadsheets are as follows:

1. Agent Environment: Defines the physical layout of the store using cells.
2. Agents: Defines agents, as rows in spreadsheet, and their attributes.
3. Time Series Output Log: Records and charts average shopper frustration over time for the course of the simulation.
4. Aggregate Output Log: Records statistics on the simulation, such as successful trips.

5.2.1 Agent Environment

The retail shopping model has a full store floor plan (Figure 3). The floor plan is composed of spreadsheet cells marked with colors and text as follows:

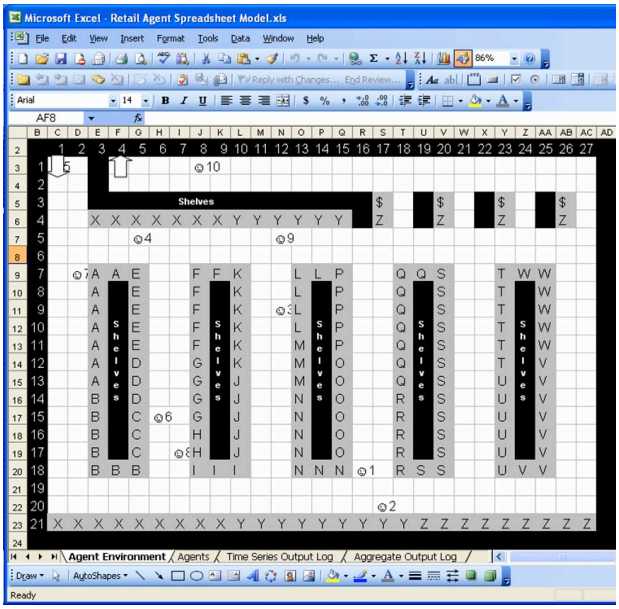


Figure 3: Store Layout in Excel Spreadsheet

- Walls are indicated by black cells,
- Shelves are indicated by gray cells,
- Items for sale are denoted by the letters A through Z,
- Checkout counters are indicated by "$", and
- Entry and exit doors are shown by arrows.

The numbers down the far left column and across the top row (black background) represent relative cell coordinates. Using relative cell coordinates simplifies many calculations. For example, the cell at relative row 7 and relative column 3 (Excel cell E9) contains item A. Agents navigate the store using a named range, [Environment], located at relative location (0, 0) (Excel cell B2). For example, Agent 4 is located at relative location row 5 and column 5 (Excel cell G7).

5.2.2 Agents

The Agents worksheet defines the agents. Agents are the rows in the spreadsheet and agent attributes are denoted in the columns.

- Name
- Active Status (Yes, No)
- Location (Row, Column)
- Symbol
- Status (Checkout, Browsing, Waiting, …)
- Target Item
- Target Location (Row, Column)
- Remembered Item
- Remembered Item Location (Row, Column)
- Frustration (number between 0 and 1)
- Number of items found
- Number of trips made

Some of the agent attributes, are static and do not change as the simulation progresses. Some attributes are dynamic, and are updated during the simulation. Static attributes are Name, Symbol, Target Item, and Target Location. All other attributes are dynamic.

Shopper Frustration. Shoppers who have a hard time finding their item of interest become frustrated and eventually leave the store without purchasing an item. Several factors may be responsible for a shopper being unable to find their item in a reasonable amount of time and becoming frustrated, such as limited knowledge of their item's location, crowds slowing their progress, and a complex store layout that extends search time. Shoppers' frustration drops when they find their item of interest. Shoppers remember their frustration from previous trips when they return to the store.

Agent States. Agents are characterized by their status or "state" which denotes what they are trying to do at any given time. Shopper agents can be in one of five possible states:

- *Waiting*: Shopper is waiting to enter the store.
- *Browsing*: Shopper is moving through the store and looking for the desired item.
- *Found*: Shopper just found the desired item.
- *Checkout*: Shopper is looking for a checkout counter.
- *Leaving*: Shopper is looking for the exit door.

The Agent State Transition diagram shows the possible transitions between the states of each shopper agent (Figure 4). For example, the shopper is initialized in the "Browsing" state, moving through the store in search of its desired item. Either the shopper finds the item, and transitions to the "Found" state, or becomes frustrated and enters the "Checkout" state without an item. In this case, the shopper is looking for a checkout counter in order to exit the store. The names on the arrow links indicate the name of a function associated with the workbook's macro (see below).

Agent interaction is limited in the model, but illustrative. Agents interaction consists of an agent inspecting the adjacent cells at each time step and determining whether other agents occupy adjacent cells. The agent selects an available cell to move into that is not occupied by another agent. At this point in the simulation logic, one could implement more significant forms of agent interaction.

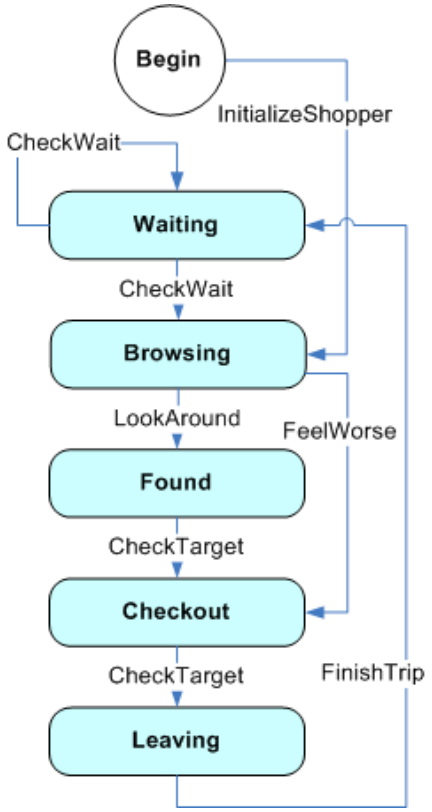


Figure 4: Agent State Transition Diagram

5.2.3 Output Log Worksheet

The Time Series Output Log worksheet records the average frustration level for all shoppers in the store and graphs the frustration level over the simulation time (Figure 5). The graph is initially constructed manually within the spreadsheet and is then updated and dynamically displayed as the simulation progresses.

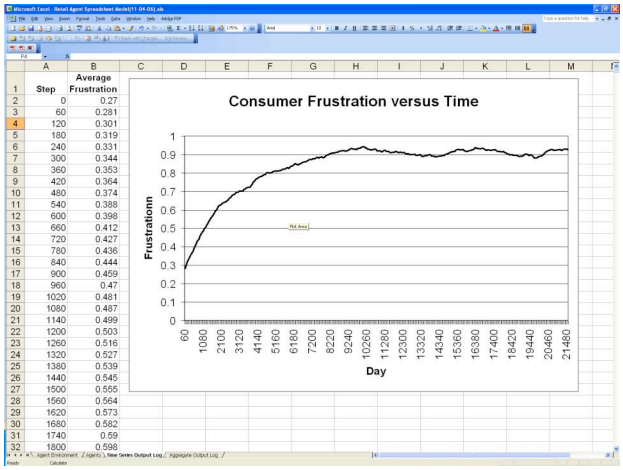
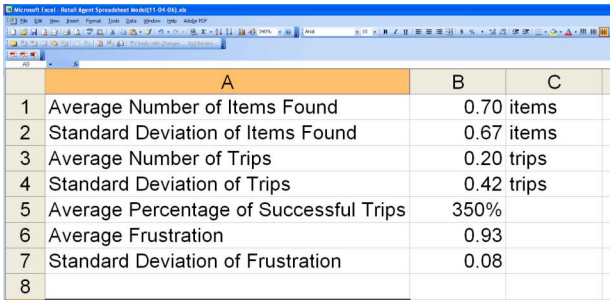


Figure 5: Time Series Output Log Worksheet

5.2.4 Aggregate Output Log worksheet

The Aggregate Output Log worksheet computes and reports on the important statistics of the simulation regarding the shopper agents experiences in the store (Figure 6).



	A	B	C
1	Average Number of Items Found	0.70	items
2	Standard Deviation of Items Found	0.67	items
3	Average Number of Trips	0.20	trips
4	Standard Deviation of Trips	0.42	trips
5	Average Percentage of Successful Trips	350%	
6	Average Frustration	0.93	
7	Standard Deviation of Frustration	0.08	
8			

Figure 6: Aggregate Output Log Worksheet

5.3 Shopper Agent Model Structure

The basic structure of the Shopper Agent Model is shown in Figure 7. The basic functions are (1) initialization of the data and agents in the model, (2) a loop over all the time periods in the model and, within this, another loop that loops over all the agents in the model, and (3) a summary of the statistics generated by the simulation. This loop-within-loop structure is the simplest way to implement time-stepping and agent interactions in a model. Essentially all shoppers execute their behaviors in the same sequential order at each time step, and time steps are uniform in length. Most real-world agent-based models implement a more complex structure for scheduling events and agent interactions, which would be a capability provided by an agent-based toolkit. The behavior of each shopper agent are embedded within the Shop function, as indicated in the figure.

A program written as an Excel macro simulates the tiem stepping and the agent states. The Excel macro is a program written in the VBA language and associated with the Shopper Agent workbook. (VBA is Visual Basic for Applications, the macro programming language for Excel and other Microsoft Office applications.). Readers programming in VBA may benefit from a good book on Excel macro programming such as Simon (2002).

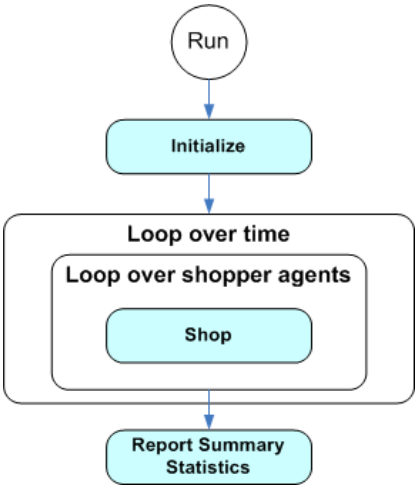


Figure 7: Structure of Shopper Agent Model

The Shop function is the most complex part of the program. It directs the agent behavior, state transitions, and the process of shopper movement throughout the store. Shop calls a collection of simple functions that performs various tasks such as changing the agent state (see Figure 8).

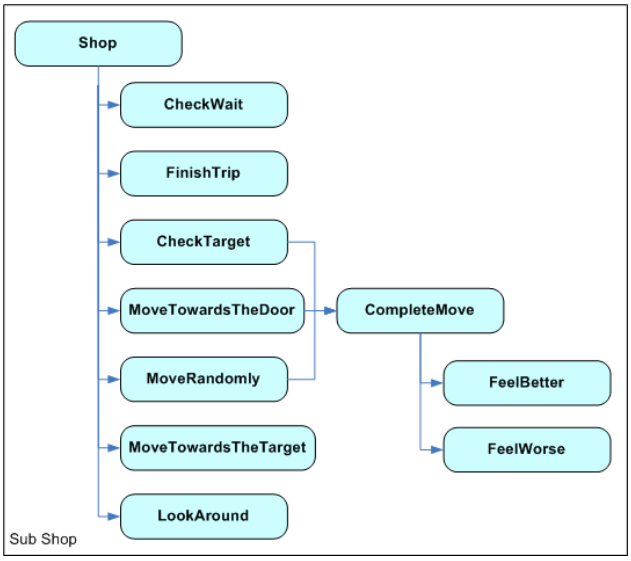


Figure 8: Structure of Subroutine Shop

The VBA code that implements Subroutine Shop is shown in Figure 9 to provide an idea of the complexity of the code. The complete macro is comprised of 18 subroutines consisting of 200 lines of executable code. Spatial reasoning on the part of the agents is accomplished through the use of named ranges, relative cell references, and offsets. A complete description of the model implementation and code is included in North and Macal (2007).

```
' The main shopper shopping routine.
Sub Shop(shopper As Range)

  ' Check to see if we have found the
target.
  If (shopper.Offset(0, 5) = "Waiting")
Then

    ' Check to see if our wait is over.
    Call CheckWait(shopper)

  ' Check to see if the shopper is at
the door
  ' and ready to leave the store.
  ElseIf ((shopper.Offset(0, 2) = 1) And
_
    (shopper.Offset(0, 3) = 4)) Then

    ' Note the finished trip.
    Call FinishTrip(shopper)

  ' Check to see if we have found the
target.
  ElseIf   (shopper.Offset(0,   5)   =
"Found") Then

    ' Check the target item.
    Call CheckTarget(shopper)

  ' Check to see if the shopper should
leave the store.
  ElseIf (shopper.Offset(0, 5) = "Leav-
ing") Then

    ' Move towards the door.
    Call MoveTowardsTheDoor(shopper)

  ' Check to see if we are frustrated or
distracted.
  ElseIf (shopper.Offset(0, 12) > Rnd())
Then

    ' Move randomly.
    Call MoveRandomly(shopper)

  Else

    ' Move towards the target using the
    ' Manhattan distance.
    Call MoveTowardsTheTarget(shopper)

  End If

  ' Check to see if what we are looking
for is nearby.
  Call LookAround(shopper)

End Sub
```

Figure 9: VBA Code for Subroutine Shop

5.4 Extending the Shopper Agent Model

The Shopper Agent Model as presented here is a simple example of an agent-based model. The model illustrates the basic elements of agent representation, interaction, and scheduling of agent actions and events. The model includes a modularized representation for both agent attributes (in the Agents spreadsheet) and for agent behaviors (as implemented in the Shop subroutine of the VBA code). More sophisticated agent behaviors, such as agent learning and adaptation, can readily be implemented in the same framework presented here, but this would require more VBA code to be written. Agent interaction is implemented in this spatial model by using cell references specific to spreadsheets so that agents can sense neighboring agents in adjacent cells. In principal, this logic can be extended in the VBA code to consider more sophisticated types of agent interaction, such as agent communication, contention for resources such as items or floor space, and agent recognition of acquaintances.

Although VBA is technically object-oriented in nature, the VBA code that is developed for the model is largely procedural in nature, that is, operations are implemented on a step-by-step basis. Other types of coding would be required for desktop (such as stylized languages) or large-scale (generally, object-oriented) agent-based environments.

As the scale of the Shopper Agent Model is increased, the spreadsheet implementation of the model may become relatively inefficient, as more agents are added to the model, as agent behaviors become more complex, or as the scheduling of agent interactions become more complicated. At this point, it may be desirable to move to one of the desktop or large-scale environments having facilities specifically designed for agent-based simulation.

6 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 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

This work was supported by the U.S. Department of Energy under contract number DE-AC02-06CH11357.

REFERENCES

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

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

Axelrod, R. 1997b. "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.

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

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

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. 2006. *BioDefense through City Level Multi-Agent Modeling of Bio and Chemical Threats*. Arizona Spring Biosurveillance Workshop, Tucson, Arizona.

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 <http://www.complexity.org.au/ci/vol08/casti01/>.

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, Argonne, IL, ANL-06/16 (report prepared for the Illinois Commerce Commission), April.

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. O. Kimbrough, et al. (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. G. 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:

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

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.

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, et al. (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. A., G. J. Gumerman and R. G. Reynolds. 2005. Simulating ancient societies, *Scientific American*, July.

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

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

Macal, C. 2004a. Emergent structures from trust relationships in supply chains, in *Proc. Agent 2004: Conf. 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 *Proc. Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Macal, C. M., D. L. Sallach, and M. J. North (eds.), Chicago, IL, Oct. 7-9, available at <http://www.agent2004.anl.gov>, pp. 185-204.

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

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

North, M. J., N. T. Collier, and J. R. Vos. 2006. Experiences in Creating Three Implementations of the Repast Agent Modeling Toolkit, *ACM Transactions on Modeling and Computer Simulation*, 16(1):1-25, January.

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.

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.

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.

Reynolds, Craig. 2006. Boids, <http://www.red3d.com/cwr/boids/>.

ROAD (Repast Organization for Architecture and Design). 2004. Repast Home Page, Available at <http://repast.sourceforge.net/>.

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

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, <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. <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), Jan. 31.

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

Wilkinson, T. J., M. Gibson, J. H. Christiansen, M. Widell, D. Schloen, N. Kouchoukos, C. Woods, J. Sanders, K.-L. Simunich, M. Altaweel, J. A. Ur, C. Hritz, J. Lauinger, T. Paulette and J. Tenney. 2007. Modeling Settlement Systems in a Dynamic Environment, in *The Model-Based Archaeology of Socionatural Systems*, Kohler, T. A. and S. E. v. d. Leeuw, eds., pp. 175-208, School for Advanced Research Press: Santa Fe, NM.

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 Int'l., 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>.