Natural Resources and Environmental Issues

Volume 8 *SwarmFest 2000* Article 2

2001

Swarm simulation system

Swarm Development Group
Santa Fe Institute, NM

THE SWARM SIMULATION SYSTEM

SWARM DEVELOPMENT GROUP

624 Agua Fria, Suite 2, Santa Fe, NM 87501, Email: swarm@swarm.org

Swarm is a software package for multi-agent simulations of complex adaptive systems, originally developed at the Santa Fe Institute. Swarm is intended to be a useful tool for researchers in a variety of disciplines. The basic architecture of Swarm is the simulation of collections of concurrently interacting agents: with this architecture, we can implement a large variety of agent based models. The Swarm software is available to the general public under GNU licensing terms. Swarm is experimental software, which means that it's complete enough to be useful but will always be under development.

In the Swarm system, the basic unit of simulation is the swarm, a collection of agents executing a schedule of actions. Swarm supports hierarchical modeling approaches whereby agents can be composed of swarms of other agents in nested structures. Swarm provides object-oriented libraries of reusable components for building models and analyzing, displaying, and controlling experiments on those models. Swarm models may be written using Java, or Objective C, and other languages are on the horizon. Swarm is available as a packaged (binary) distributions for Microsoft Windows (Windows 9x, Windows NT 4, or Windows 2000), Debian GNU/Linux 2.2 (ix86, sparc) Red Hat GNU/Linux (Red Hat 6.1 / x86, Red Hat 6.1 / sparc, LinuxPPC R5 1999), SuSE 6.3 GNU/Linux (x86), and Solaris 2.7 (Sparc). More information about Swarm can be obtained from our web pages, http://www.swarm.org.

COMPUTATIONAL APPROACHES TO COMPLEX SYSTEMS

In the sciences, especially in the study of complex systems, computer programs have come to play an important role as scientific equipment. Computer models provide many advantages over traditional experimental methods, but also have several problems. In particular, the actual process of writing software is a complicated technical task with much room for error and difficult to repeat by other researchers. Thus, computer modeling frequently turns good scientists into bad programmers. Most scientists are not trained as software engineers. As a consequence, many home-grown computational experimental tools are (from a software engineering perspective) poorly designed. The results gained from the use of such tools can be difficult to compare with other research data and difficult for others to reproduce because of the quirks and unknown design decisions in the specific software apparatus. Furthermore, writing software is typically not a good use of a highly specialized scientist's time. In many cases, the same functional capacities are being rebuilt time and time again by various research groups, a tremendous duplication of effort.

A subtler problem with custom-built computer models is that the final software tends to be very specific, a dense tangle of code that is understandable only to the people who wrote it. Typical simulation software contains a large number of implicit assumptions, accidents of the way the particular code was written that have nothing to do with the actual model.

And with only low-level source code it is very difficult to understand the high-level design and essential components of the model itself. Such software is useful to the people who built it, but makes it difficult for other scientists to evaluate and reproduce results. For computer modeling to mature, a standardized set of well-engineered software tools usable on a wide variety of systems. The Swarm project aims to produce such tools through a collaboration between scientists and software engineers.

Swarm is an efficient, reliable, reusable software apparatus for experimentation. If successful, Swarm will help scientists focus on research rather than on tool building by giving them a standardized suite of software tools that provide a well-equipped software laboratory.

Multi-agent Discrete Event Simulation

The modeling formalism that Swarm adopts is a collection of independent agents interacting via discrete events. Within that framework, Swarm makes no assumptions about the particular sort of model being implemented. There are no domain specific requirements such as particular spatial environments, physical phenomena, agent representations, or interaction patterns. Swarm simulations have been written for such diverse areas as chemistry, economics, physics, anthropology, ecology, and political science.

The basic unit of a Swarm simulation is the agent. An agent is any actor in a system, any entity that can generate events that affect itself and other agents. Simulations consist of groups of many interacting agents. For example, an ecosystem simulation could consist of agents representing coyotes, rabbits, and carrots. In an economic simulation, agents could be companies, stockbrokers, shareholders, and a central bank. Simulation of discrete inter- actions between agents stands in contrast to continuous system simulations, where simulated phenomena are quantities in a system of coupled equations.

Agents define the basic objects in the Swarm system, the simulated components. A schedule of discrete events on these objects defines a process occurring over time. In Swarm, individual actions take place at some specific time; time advances only by events scheduled at successive times. A schedule is a data structure that combines actions in the specific order in which they should execute. For example, the coyote/rabbit simulation could have three actions: rabbits eat carrots, rabbits hide from coyotes, and coyotes eat rabbits. Each action is one discrete event: the schedule combines the three in a specific order, e.g. each day, have the rabbits eat carrots, then they hide from the coyotes, then the coyotes try to eat the rabbits. The passage of time is modeled by the execution of the events in some sequence.

Swarms

The fundamental component that organizes the agents of a Swarm model is an object called a swarm. A swarm is a collection of agents with a schedule of events over those agents. For example, a swarm could be a collection of 15 coyotes, 50 rabbits, a garden with carrots, and a simple schedule: the rabbits eating the carrots and hiding and the coyotes eating the rabbits (Figure 1). The swarm represents an entire model: it contains the agents as well as the representation of time.

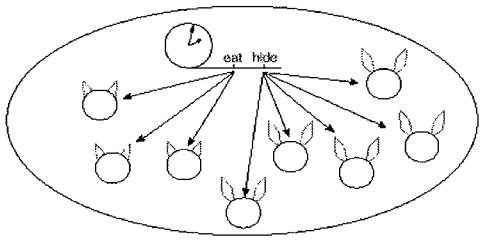

Figure 1. An example of a swarm of rabbits and coyotes.

In addition to being containers for agents, swarms can themselves be agents. A typical agent is modeled as a set of rules, responses to stimuli. But an agent can also itself be a swarm: a collection of objects and a schedule of actions. In this case, the agent's behavior is defined by the emergent phenomena of the agents inside its swarm. Hierarchical models can be built by nesting multiple swarms. For example, one could build a model of a pond inhabited by single celled animals. At the highest level, a swarm is created that contains agents: the swarm represents the pond, and each agent represents one animal. The behavior of cells could be defined simply as some algorithm, but a cell is itself a collection of organelles: a nucleus, mitochondria, endoplasmic reticulum. Another way to represent a cell is as a swarm of agents, the organelles. Two models are being combined: the pond as a swarm of cells and the cell as a swarm of organelles.

The ability to build models at various levels can be very powerful. Swarm allows users to

explicitly build and test multi-level models. A swarm can explicitly represent an emergent structure, a group of agents behaving cohesively as a single agent. Because swarms can be created and destroyed as the simulation executes, Swarm can be used to model systems where multiple levels of description dynamically emerge.

Another use of multiple swarms is to support the modeling of agents that themselves build models of their world. As noted in Hogeweg's MIRROR system (Hogeweg 1989), in some simulations, especially those where the agents have a cognitive component, an important factor in the system dynamics is an agent's own beliefs about its world. In Swarm, agents can themselves own swarms, models that an agent builds for itself to understand its own world. For example, in an economic simulation of a swarm of companies the researcher might be interested in the theory each company has of its competitors' actions. To model this in the Swarm system, the model builder would give each company-agent its own swarm: these private swarms implement each company's model of the world.

The formalism of the swarm is a natural way of encapsulating a simulation: a swarm simply represents a group of agents and their schedule of activity. The modularity and composability of swarms allows for a flexible modeling system. Swarms can be nested to directly represent multi-level simulations, and they can be used by the agents themselves as models of their own world (Figure 2).

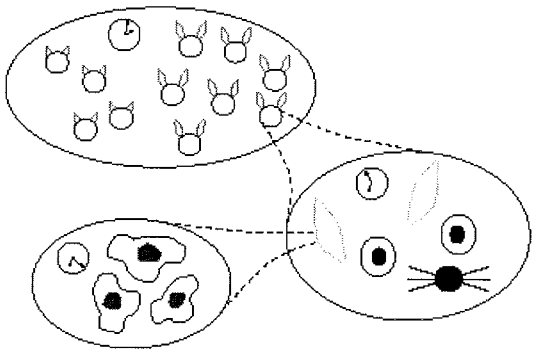

FIGURE 2. Hierarchical swarms: rabbits, rabbit parts, individual cells.

OBJECT ORIENTED TECHNOLOGY

The logical structure of swarms of agents interacting through discrete events is implemented in a straightforward way in Java, Objective C, or another object oriented (OO) language. In OO programming software consists of the definitions of various classes of objects. An object is a combination of instance variables for the object's state and methods that implement the object's behavior (Robson 1985). In Swarm an agent is modeled directly as an object. Types of agents (generic coyotes) are classes, and specific agents (a particular coyote) are objects, instances of the Coyote class. Each object carries with it its own state variables, but the generic definition of its behavior is provided by the class (Figure 3).

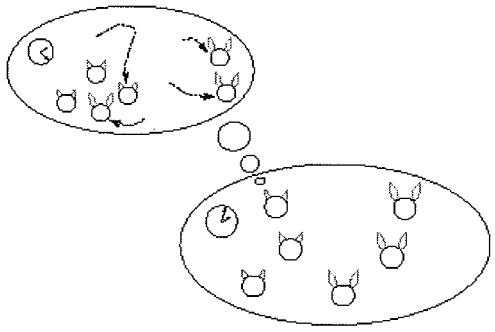

FIGURE 3. Coyote model linked with a model for hunting.

The instance variables for an object directly represent the state of the agent. For example, a particular rabbit's mass could be stored as an integer variable in the class Rabbit. The methods of an object implement the behavior of the agent. Rabbits hide: this is implemented by having a hide method defined on class Rabbit. The schedule of activity for a model is then simply a partially ordered series of such actions to be performed on objects. Each action specifies a method to be executed on a target object.

The Swarm system itself is an object framework: a set of class libraries that are designed to work together and facilitate implementation of agent-based models.

There are seven core libraries in Swarm: defobj, collections, random, tkobjc, activity,

swarmobject, and simtools. The first four libraries are support libraries with potential use outside of Swarm; the last three are Swarm-specific. There are also currently three domain-specific libraries available for Swarm model builders: space, ga, and neuro. The details of each library is discussed below. In addition to being a natural way to implement multiagent simulations, OO programming is also a convenient technology for building libraries of reusable software. Users can start to build models by directly instantiating useful classes from the Swarm libraries. If there is no particular class that has the precisely needed behavior, one can take a preexisting class and specialize it, adding new variables and methods via inheritance. Inheritance and OO encapsulation make writing reusable code easier than traditional procedural programming. This in turn makes it easier for people to share Swarm modeling software they have written, facilitating the exchange of ideas and techniques within simulation communities.

One addition to standard OO programming that Swarm implements is the probe facility. In most computer programs, it is enough that the program does what you want, that Windows doesn't lose your files, and that your word processor can print out your papers. But in simulation it is important that all aspects of the computation be observable, that it be easy for the researcher to measure data from a running model. The Swarm system defines the ability for any object to be probed. Probes allow any object's state to be read or set and any method to be called in a generic fashion, without requiring extra user code. Probes are used to make data analysis tools work in a general way and are also the basis of graphical tools to inspect objects in a running system.

Structure of a Swarm Simulation

The core of a Swarm simulation is the modeled world itself. In the simplest case, a model consists of one swarm inhabited by a group of agents and a schedule of activity for those agents. The agents themselves are implemented as objects. Agents are created by taking a class from the Swarm libraries, specializing it for the particular modeling domain, and then instantiating it, one object per agent. For example, an agent that is a neural network could start by taking a general purpose neural network class from the neuro library, adding extra methods needed for the specific type of network, and then creating an instance of it to be the actual neural network.

In modeling it is common (but not universal) to speak of agents as living in an environment. Many simulation platforms _x the environment as a particular type; two dimensional grids are common. A distinguishing feature of Swarm is that there is no design requirement for a particular kind of environment. For instance, a coyote/rabbit model might have the rabbits living in the environment of a garden. In Swarm, this environment is itself just another agent. The garden is simply an instance of a user-defined garden agent, perhaps based on a cellular automata to simulate growth of carrots. The garden agent might have a special status in the model, but in the underlying software it is treated no differently than any other agent. In the general case, the environment for the agents is the agents themselves: some agents might have a larger influence than others, but in Swarm they are considered fundamentally equivalent.

Once a user has defined the agents and established their relationships, the last step in building the model itself is to put the agents together into a swarm. The user writes a schedule of activity for the agents, defining how time is simulated in the system by creating a set of actions in a specified ordering. Schedules are built by creating instances of data structures from the activity library, filling them in with ordered object/message pairs. Once the schedule is completed the model swarm is ready to be executed.

A model running by itself is not very interesting: data collection tools are needed to observe the model and record what is happening. In Swarm measurement happens by the actions of observer agents, special objects whose purpose it is to observe other objects via the probe interface (Figure 4). For example, one observer agent might be watching the number of rabbits and producing a time series graph of population dynamics. Another observer agent could track the spatial distribution

of the coyotes, storing data to a file for later analysis.

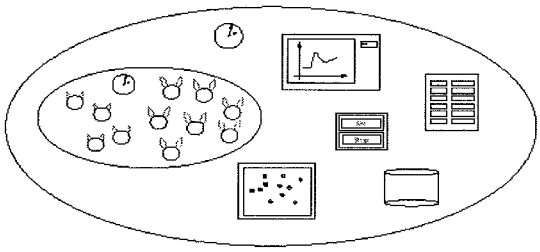

FIGURE 4. A swarm of observer agents measuring a model.

The observer agents themselves are a swarm, a group of agents and a schedule of activity. By combining this swarm with a model swarm running as a subswarm of the observer, a full experimental apparatus is created. By using hierarchical swarms to separate data collection from the model, the model itself remains pure and self-contained, a simulated world under glass. Different observer swarms can be used to implement different data collection and experimental control protocols, but the model itself remains unchanged.

SWARM LIBRARIES

Swarm libraries serve two major functions. The libraries are a set of classes that model builders can use by direct instantiation. For many objects, especially highly technical ones such as schedule data structures, it's likely that all a user will ever do is use the classes as provided. But in addition, one can use Swarm libraries by subclassing them, specializing particular classes for particular modeling needs. Both modes of using the Swarm libraries are important; Swarm is designed to facilitate both as appropriate.

Simulation Libraries

The main novelty in the Swarm system is the design of the simulation-specific libraries themselves. The activity library contains the heart of the simulation mechanism, the scheduling data structures and execution support. The underlying structure of the activity structures are time-ordered schedule, clocks with specific events at particular times. These schedules are implemented in activity as a collection of actions sorted by timestamp.

Ambiguity can occur in partial orders and time-based schedules as a result of two or more actions scheduled at the same time or in the same relative order. Swarm resolves such ambiguity by defining a concurrent group type, an explicit indication of how to execute a group of actions that are defined at the same time. Options include running the group in an arbitrary, fixed order; running the group in a random order every time; or actually running each action concurrently, for future implementation on parallel machines. The explicit notation of a concurrent group type helps to expose and remove any hidden assumptions in the time structure of a model.

Two modes of operation are supported: a fully graphical mode for interactive exploration and a batch mode for offline data collection. simtools also contains the data analysis and display support. The library contains classes that can generate summaries of statistical data, draw time series graphs, etc. Data analysis objects are completely generic in their application; users specify the particular data to collect by creating probes on the observed objects.

Software Support Libraries

One library worthy of mention is the random number library. This gives the user a suite of random number generators. In computer simulation the quality of random number generators is absolutely essential: it is very easy to have subtly wrong results because of a generator with biases or correlation. The Swarm random number library includes several classes of generators drawn from published research. It includes a bibliography as well as tested implementations. Because the library itself is object oriented it is simple to have multiple, independent random number streams, a feature that aids repeatability of experiments.

Model-specific Libraries

In addition to the support and simulation libraries required for all Swarm applications there are several optional libraries that can be used for particular modeling domains. Libraries exist to

support two dimensional spaces, genetic algorithms, and neural networks. Development effort for Swarm so far has concentrated on the required libraries; now that the Swarm infrastructure is in place, effort is shifting towards building more model-specific libraries. There are many opportunities for user-contributed code.

The Swarm distribution comes with a simple space library, a set of classes for two dimensional discrete lattices. These sorts of spaces are common in ecosystem simulations. The base class in space is a two dimensional array that stores objects or integer values at particular grid points. Several classes inherit from this base to provide dynamics, for instance a cellular automata approximation to diffusion. The current space library is merely a suggestion of the kinds of environments a model could use: in the future, we plan to have spaces with continuous values and dynamics defined by differential equations. Spaces with other topologies are also crucial: three dimensions, non-discrete coordinates, and arbitrary graph structures are all needed by applications.

User Community

Swarm is a service to the community of researchers building computers simulations. Computer programs have become an important aspect of experimental science and yet few general purpose tools exist to help people write models. The goal of Swarm is to provide consistent experimental tools in the form of libraries of carefully designed, implemented and tested code. Such libraries can also be developed and exchanged within user communities that organize and maintain the model-building components applicable to their shared needs.

Swarm defines a structure for simulations, a framework within which models are built. The core commitment is to a discrete-event simulation of multiple agents using an object- oriented representation. To these basic choices Swarm adds the concept of the swarm, a collection of agents with a schedule of activity. The swarm is the basic structural element of a model: it is the basic collection that allows scaling within large models, and also supports the modeling of complex, multi-level dynamics including agents that model their own world. In all these choices, the goal of Swarm is to enable a higher level of representation for simulations, thereby making it easier to understand, implement, repeat, and communicate computer models.

Since Swarm's first limited beta release in October 1995, numerous publications have utilized Swarm (See reference list). Our goal is to create a distribution that is simple, complete, well documented, and bug-free. Our long-term goal is to foster a community of modelers, researchers who use computer software for experimental science. Swarm is already helping to provide focal point for discussion of simulation techniques and methodology. Swarm can also facilitate the sharing of modeling components and libraries within particular research communities, fostering an important form of intellectual exchange. Finally, a formalized framework for model definition establishes a necessary standard of specification for computer programs used as tools in experimental science.

Acknowledgments

Swarm was initially developed at the Santa Fe Institute with the support of Grant No. N00014-95-1-1000 from the Office of Naval Research and Grant No. N00014-94-1-G014 from the Naval Research Laboratory, both acting in cooperation with the Defense Advanced Research Projects Agency. Swarm benefited from earlier support from The Carol O'Donnell Foundation, Mr. and Mrs. Michael Grantham, and the National Science Foundation. SFI also gratefully acknowledges invaluable support from Deere & Company for this project. Thanks to the Joint Warfare Analysis Center for continuing support.

Literature Cited

Hogeweg, P. 1989. Mirror beyond mirror, puddles of life. In Chris Langton, (ed.), Artificial Life. Addison-Wesley.

Robson, D. 1985. Smalltalk-80: the language and its implementation. Addison-Wesley, 11.

REFERENCE LIST

Book

Luna, F. and L. Stefansson. 2000. Economic simulations in swarm: Agent-Based Modelling and Object Oriented Programming. In H. Amman and A. Nagurney (ed.). Advances in Computational Economics. Volume 14.Kluwer Academic Publishers. ISBN 0-7923-8665-5.

Papers

Armstrong, A. A., and E. H. Durfee. 1998. Mixing and Memory: Emergent Cooperation in an Information Marketplace. Proceedings of the Third International Conference on Multiagent Systems, July 1998. IEEE Computer Society Press.

Booth, G. 1997. Gecko: A Continuous 2D World for Ecological Modeling Artificial Life. Summer 1997. 3. 3:147--163.

Bruhn, P. 1997. Enterprise Simulation using Multi-Agent System Modeling and the Swarm Toolkit. M.S. Thesis.

Burkhart, R. 1994. The Swarm Multi-Agent Simulation System. Proceedings from (OOPSLA) The Object Engine, Workshop 7 September 1994.

Burkhart, R. 1995. Object-Oriented Programming Systems, Languages, and Applications. Proceedings from (OOPSLA) 1995 Adaptable and Adaptive Software, Workshop 8 October 1995, Create-phase Protocols for Object Customization.

Burkhart, R. 1997. Schedules of Activity in the Swarm Simulation System. Proceedings from OOSPLA '97, Workshop on OO Behavioral Semantics.

Carnahan, J., S. Li, C. Costantini, Y. T. Toure, and C. E. Taylor. 1997. MIT Press. Computer Simulation of Dispersal by em Anopheles Gambiae s.l. in West Africa Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems1997. pp. 387--394. Series: Complex Adaptive Systems. MIT Press. Boston.

Chantemargue, F., T. Dagaeff, M. Schumacher, and B. Hirsbrunner. 1998. Autonomous Agents and Cooperation: Application to Collective Robotics, University of Fribourg. Computer Science Department. Fribourg Switzerland. Internal Note 98-03.

Chantemargue, F., O. Krone, M. Schumacher, T. Dagaeff, and B. Hirsbrunner. 1998. Autonomous Agents: from Concepts to Implementation. Proceedings of the Fourteenth European Meeting on Cybernetics and Systems Research (EMCSR'98) April 14-17 1998. Vienna, Austria. 731-736.

Cohen, M. D., R. L. Riolo, and R. Axelrod. 1999. The Emergence of Social Organization in the Prisoners' Dilemma: How Context-Preservation and other Factors Promote Cooperation PSCS Working Paper 99-01-002.

Dagaeff, T., F. Chantemargue, and B. Hirsbrunner. 1997. Emergence-based Cooperation in a Multi-Agent System. Proceedings of the Second European Conference on Cognitive Science (ECCS'97) Manchester U.K.. April 9-11. pp. 91-96.

Downing, K., and P. Zvirinsky. 1999. The simulated evolution of biochemical guilds: reconciling gaia theory and natural selection. Artificial Life. **5.** 4.

Fulkerson, B., and G. Staffend. 1997. Decentralized Control in the Customer Focused Enterprise Annals of Operations Research. **77**:325-333.

Hinsch, M., and J. J. Merelo. 1998. Coevolving Iterated Prisonner's dilemma strategies in different environments Univ. Granada, Spain. GeNeura Team, Dept. ATC, g-98-1.

Jares, T. 1998. The Survival and Consequences of Noise Traders in Financial Markets: A Numerical Modeling Approach. Ph.D. Thesis. University of Nebraska.

Johnson, P. 1998. Adaptive agents versus rational actors: social science implications.Annual Meeting of the American Political Science Association3-6 September 1998. Marriott Copley Place and Sheraton Boston Hotel and Tower, Boston. Massachusetts.

Johnson, P. 1998. An agent based model of the exchange theory of interest groups. Annual Meeting of the American Political Science Association, 3-6 September 1998. Marriott Copley Place and Sheraton Boston Hotel and Tower, Boston, Massachusetts.

Kohler, T.A., and E. Carr. 1996. Swarm-based modeling of prehistoric settlement systems in southwestern North America. Proceedings of Colloquium II, UISPP, XIIIth Congress September 1996. Series: Sydney University Archaeological Methods. 5. I. Johnson and M. North. School of Archaeology, University of Sydney, Australia.

Kohler, T.A., C.Van West, E. P. Carr, and C. G. Langton. 1996. Agent-based modeling of prehistoric settlement systems in the northern American Southwest. Proceedings of Third International Conference Integrating GIS and Environmental Modeling, Santa Fe, New Mexico. National Center for Geographic Information and Analysis. Santa Barbara, California.

Kohler, T.A., J. Kresl, C. Van West, E.Carr, and R.Wilshusen. 1999. Be There Then: A Modeling Approach to Settlement Determinants and Spatial Efficiency among late Ancestral Pueblo Populations of the Mesa Verde Region, U.S. Southwest Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes. T. Kohler and G. Gumerman. Santa Fe Institute and Oxford University Press.

Kreft, J. U., G. Booth, and J. W. T. Wimpenny. 1998. BacSim, a simulator for individual-based modelling of bacterial colony growth. Microbiology. **144**:3275-3287.

Krone, O., F. Chantemargue, T. Dagaeff, M. Schumacher, and B. Hirsbrunner. 1998. Pages 149-158.Coordinating Autonomous Entities Proceedings of the ACM Symposium on Applied Computing (SAC'98). Special Track on Coordination, Languages and Applications February 27 - March 1. Atlanta, Georgia.

Krone, O., F. Chantemargue, T. Dagaeff, and M. Schumacher. 1998. Coordinating autonomous entities with STL. The Applied Computing Review. Special issue on Coordination Models Languages and Applications.

Krothapalli, N. K. C., and A.V. Deshmukh. 1997. Effects of negotiation mechanisms on performance of agent based manufacturing systems. Proceedings of the Seventh International Conference on Flexible Automation and Intelligent Manufacturing. 704-717.

Krothapalli, N. K. C., and A. V. Deshmukh. 1998. Self-regulating negotiating schemes for robust agent--based manufacturing systems. Proceedings of the Seventh Industrial Engineering Research Conference.

Krothapalli, N. K. C., and A.V. Deshmukh. 1998. Design of negotiation protocols for multi--agent manufacturing systems International Journal of Production Research. 1998 (in press).

Lang, Railsback & Assoc. 1999. Tools for Individual-based Stream Fish Models: Improving the Cost-Effectiveness and Credibility of Individual-based Approaches for Instream Flow Assessment, EPRI, Electric Power Research Institute, Palo Alto, California EPRI TR-114006.

Manuca, R., Y. Li, R. Riolo, and R. Savit. 1998. The structure of adaptive competition in minority games. PSCS Working Paper 98-11-001. 1998.

Marshall, J. A. R., and J.E. Rowe. 1999. The evolution of cooperation through kin selection. International Journal of Systems Science. (submitted).

McMullin, B. 1997. Computational autopoiesis: the original algorithm. Santa Fe Institute Working Paper 97-01-001

McMullin, B. 1997. SCL: an artificial chemistry in swarm, SFI Working Paper 97-01-002.

McMullin, B., and F. J. Varela. 1997. Rediscovering computational autopoiesis. In P. Husbands and H. Inman (eds.), Proceedings of the Fourth European Conference on Artificial Life, July 1997. School of Cognitive and Computing Sciences, University of Sussex. Series: Complex Adaptive Systems. MIT Press.

Minar, N., R. Burkhart, C. Langton, and M. Askenazi. 1996. The swarm simulation system: a toolkit for building multi-agent simulations. SFI Working Paper 96-06-042.

Parunak, H. V. D., R. Savit, and R. L. Riolo. 1998. Multi-agent systems and agent-based simulation. In Sichman, Conte, and Gilbert. Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide Series: LNAI series. Springer-Verlag.

Pepper, J. W., and B. Smuts. 1999. The evolution of cooperation in an ecological context: an agent-based model. In T. Kohler and G. Gumerman (eds.) Dynamics in Human and Primate Societies: Agent-Based Modeling of Social and Spatial Processes. Santa Fe Institute and Oxford University Press.

Pepper, J. W. 2000. The evolution of modularity in genome architecture. Evolvability workshop at Artificial Life VII1-2 August 2000. Reed College. Portland Oregon.

Pitt, W. C., A. Ogawa, F. F. Knowlton, and P. W. Box. 2000. Evaluation of depredation management techniques for territorial animals using a computer model: coyotes as a case study. Proceedings of Vertebrate Pest Control Conference. In press.

Pitt, W. C., P. W. Box, and F. F. Knowlton. 2001. A new approach to understanding canid populations using an individual-based computer model. Endangered Species Update. In press.

Pitt, W. C., P. W. Box, and F. F. Knowlton. Modelling territorial animal populations with social structure: predictions from computer simulation. To be submitted to Ecological Modelling. In preparation.

Polhill, J. G., N. M. Gotts, and A. N. R. Law. 2001. Imitative versus nonimitative strategies in a land-use simulation. Cybernetics and Systems. **32**: 285-307.

Railsback, S. F., R. H. Lamberson, and S. Jackson. 1999. Individual-based models: progress toward viability for fisheries management. Spatial Processes and Management of fish Populations, Proceedings of the 17th Lowell Wakefield Symposium, Anchorage, Alaska.

Railsback, S. F., R. H. Lamberson, B. C. Harvey, Movement rules for individual-based models of stream fish. Ecological Modelling. **123**:73-89.

Railsback, S. F., and B. C. Harvey. Individual-based model formulation for cutthroat trout, Little Jones Creek, California. U.S. Forest Service, Redwood Sciences Laboratory, Arcata, California, in preparation.

Railsback, S. F., and B. C. Harvey. Comparison of salmonid habitat selection objectives in an individual-based model. January 2000 in preparation for Ecology.

Satterfield, T., and M. Murphy. 1999. A computational model of creole genesis. Linguistic Society of America Meeting January 1999. Los Angeles.

Savage, M., and M. Askenazi. 1998. Arborscapes: A Swarm-Based Multi-agent Ecological Disturbance Model Submitted to Geographical and Environmental Modelling, available as SFI Working Paper 98-06-056.

Savage, M., B. Sawhill, and M. Askenazi. 2000. Community Dynamics: What Happens When We Rerun the Tape? Journal of Theoretical Biology. **205**(4):515-526.

Savit, R., R. Manuca, and R. Riolo. 1998. Adaptive competition, market efficiency, phase transitions and spin-glasses. LANL Eprint archives paper adap-org/9712006.

Savit, R., R. Manuca, and R. Riolo. 1998. The dynamics of minority competition. Physical Review Letters.

Schretzenmayr, M. 1998. Strategien zur umnutzung von grossflaechigen innerstaedtischen industrie- und gewerbebrachen (Strategies for the redevelopment of large-scale inner city industrial sites). PhD Thesis, ETH No. 12473 (in German).

Stefansson, B. 1997. Swarm: an object oriented simulation platform applied to markets and organizations evolutionary programming VI1997. In P. Angeline, R. Reynolds, J. McDonnel, and R. Eberhart. Series: Lecture Notes in Computer Science. Springer-Verlag. New York.

Strader, T. J., F. R. Lin, and M. J. Shaw. 1998. Simulation of order fulfillment in divergent assembly supply chains. Journal of Artificial Societies and Social Simulation. **1**. 2.

Strader, T. J., F. R. Lin, and M. J. Shaw. 1999. The impact of information sharing on order fulfillment in divergent differentiation supply chain. Journal of Global Information Management. **7**. 1. January-March 1999.

Terna, P. 1998. Simulation tools for social scientists: building agent based models with swarm. Journal of Artificial Societies and Social Simulation. **1**. 2.

Terna, P. 1998. ABCDE: Agent based chaotic dynamic emergence series: lecture notes in artificial intelligence. Multi-Agent Systems and Agent-Based Simulation, First International Workshop, MABS'98. Springer. Berlin.

Terna, P. 2000. Economic experiments with swarm: a neural network approach to the self-development of consistency in agents' behavior economic simulations. In F. Luna and B. Stefansson (eds.), Swarm: Agent-Based Modelling and Object Oriented Programming. Dordrecht and London, Kluwer Academic.

Terna, P. 2000. The mind or no mind dilemma in agents behaving in a market. Applications of Simulation to Social Sciences. G. Ballot and G. Weisbuch. Paris, Hermes Science Publications.

Villa, F., and R.Costanza. 1998. Design of multi-paradigm integrating modeling tools for ecological research Journal of Environmental Modelling and Software. Multi-paradigm ecological modeling, remote simulation control, simulation interface design, model coordination, Swarm. (submitted).