

ERIM's Approach to Fine-Grained Agents

H. Van Dyke Parunak, Sven Brueckner, John Sauter

ERIM, PO Box 134001, Ann Arbor, MI 48113-4001
{vparunak, sbrueckner, jsauter}@erim.org

Abstract. Traditional software agents, an extension of Artificial Intelligence, seek human-level intelligence in each agent. For over 15 years, inspired by Artificial Life, ERIM has been devising architectures in which useful intelligence emerges at the system level from interactions of fine-grained agents. We have applied such architectures to a wide variety of domains, including business, industrial, and military. This white paper outlines three major principles that characterize our approach. For each we discuss *what* the principle is, *why* it is important, and *how* it works in practical implementations.

1. Introduction

Traditional agent architectures are exemplified by the BDI approach, which explicitly models the beliefs, desires, and intentions of individually intelligent agents that interact globally through protocols patterned on human language. ERIM's approach favors much smaller, simpler ("fine-grained") agents that interact only with neighbors. There are clear advantages to using fine-grained agents. For example,

- The economics of producing and deploying agents favor small devices that can be constructed using MEMS techniques.
- Local communications permit low-power operation and allow bandwidth to be reused beyond the range of each agent.
- When each agent represents only a small part of the mass of the overall system, overall functionality can be more robust if some of the agents are destroyed.

In spite of the limited individual processing capability of such agents, three principles permit us to use them to construct systems with complex functionality.

Section 2 emphasizes the importance of the *environment* in agent systems. Section 3 argues that we must pay attention to the *dynamics* that emerge from interactions multi-agent systems, with special emphasis on concepts from statistical mechanics. Section 4 suggests the importance of *growing*, rather than building, agents and agent societies. Section 5 concludes.

2. Environmental Mediation

What is it?—Agents only exist in, and only interact through, an environment (Fig. 1) [7, 16]. This insight is often ignored when researchers consider the environment as a

passive communications framework and agents are viewed as interacting directly with one another (dotted lines in Fig. 1).

This pattern of interaction is called “stigmergy” [8], from the Greek words *stigma* “sign” and *ergos* “work”: the work performed by the agents in the environment in turn guides their later actions. Such techniques are common in biological distributed decentralized systems such as insect colonies [18].

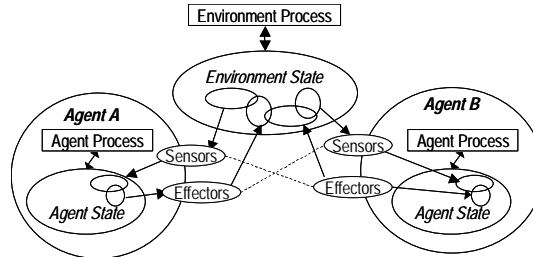


Fig. 1. Agents are bounded processes interacting through a shared environment. The common assumption that they interact directly with one another (*dotted lines*) stumbles when the environment has its own processes that can alter its state

Why is it important?—

Understanding the environmental nature of agent interactions is important both defensively and offensively.

Defensively, designers of multi-agent systems must recognize that the environment often has its own processes. In purely electronic problem domains such as digital libraries, engineers seek to suppress environmental influences through layered protocols. But the environment often intrudes in embarrassing ways when an application is scaled up for real-world use. Agent designs for non-electronic domains (such as factory automation) must consider the environment explicitly. We learned the active role of the environment in negotiation when experimenting with a contract net for manufacturing control [17]. After proving that our protocol was deadlock-free, we ran it on a physical control system, and it promptly deadlocked. The negotiation in question concerned the movement of a physical part from one workstation to another. Our analysis neglected the movement of the part itself. This physical movement conveyed information between the two workstations, and thus between their software agents. This undocumented extension of our protocol invalidated our proof and caused the system to deadlock. The arrival of the part at the receiving workstation gave that workstation information that it would not otherwise have had, namely, that the part had been delivered.

Offensively, environmental mediation enlarges the design space open to the agent engineer, by inviting the exploitation of environmental dynamics in problem-solving. A parade example is insect pheromones, which we have emulated in a system for emergent coordination among moving aircraft [21]. A related system used by wasps to assign tasks within the nest has been applied successfully to manufacturing scheduling applications [5]. Information flows through the environment can complement traditional message-based communications among agents [22]. In addition, processes taking place in the environment (e.g., aggregation, evaporation, and propagation of pheromones) can take some of the processing load from the agents, permitting achievement of a given computational task (e.g., path planning) with simpler agents than would otherwise be possible.

The interaction of agents with the environment has several characteristics that are desirable in multi-agent systems. Using environmental interaction offensively enforces these qualities in the resulting system.

- The environment is typically *distributed* over the problem domain, making it a natural point of contact for agents that are also distributed.
- Agent interaction with a distributed environment is inherently *decentralized*, reinforcing design choices that avoid a single point of failure or vulnerability.
- If agents restrict their interaction with each other to changes in the environment, and if the environment's topology can be embedded in Euclidean space, then agents will tend to interact more with *nearby* agents than with distant ones.
- The environment *integrates* actions performed by different agents.

The interactions of agents with the environment are often subsymbolic. Agents interact by emitting and sensing signals of distinct type and strength, but without the complex syntax associated with more traditional agent protocols. As a result, communications require only low bandwidth, and are robust against interception and interference. Furthermore, agents can be much simpler than those that must manipulate complex symbolic structures.

How does it work?—Taking advantage of environmental mediation in a multi-agent system requires identifying or constructing the environment and the mechanisms through which agents interact with it.

Sometimes agents can manipulate the physical world directly. For example, robots might deposit and sense chemicals, beads, or other markers in the physical world.

More commonly, the environment is computational. In pheromone-based air mission control [21], we map regions of the physical world onto “places,” computational processes that support the basic pheromone dynamics of aggregation, evaporation, and diffusion. Places form a graph in which an edge joins two places just when the regions they represent are adjacent. When a robot is in the physical region associated with a particular place, its agent resides on the place, and interacts with it. In a real-world implementation, the places might run on unattended ground sensors situated in the physical environment.

Such an environment is applicable to any domain that can be modeled as a graph, such as a social network, the organizational chart of a business, a supply network among manufacturing companies, a semantic network, or the wiring diagram of a circuit. If we can formulate a multi-agent system in terms of agents moving over a graph, we have opened the door to endowing the nodes of the graph with useful dynamics that can supplement the agents' functionality.

Once the environment is defined, one needs to define the processes that it supports. The Second Law of Thermodynamics poses a pervasive challenge to the coherence of large agent systems. Adding energy to a system is a necessary condition for overcoming the Second Law, but it is not sufficient. Gasoline in the engines of construction equipment can construct a building out of raw steel and concrete, while the same gasoline in a bomb can reduce a building to a mass of raw steel and concrete. Environmental dynamics can help, as illustrated by pheromones in insects.

In the absence of environmental mediation, agents are considered to perceive one another directly, reason about this perception, and then take rational action. Fig. 2 illustrates a more environmental model [13]. Agents record their (rational) actions by depositing pheromones in the environment. Diffusive processes in the environment generate a gradient that agents perceive, permitting ordered behavior at the agent

level. At the same time, these processes increase disorder at the micro level, so that the system as a whole becomes less ordered over time, as the second law requires.

Research in synthetic pheromones [3, 19, 24] draws directly on this model of coordination, but the model is of far broader applicability. In a multi-commodity market, individual agents follow economic fields generated by myriad individual

transactions, and self-organization in the demand and supply of a particular commodity is supported by an environment that distributes resources based on the other transactions in the system. The movement of currency in such a system provides similar functions to those of pheromones in insect systems.

We hypothesize that a coupling of ordered and disordered systems is ubiquitous in robust self-organizing systems. Environmental processes involving pheromones or currencies are a tractable way to implement such coupling, generating self-organizing agent societies.

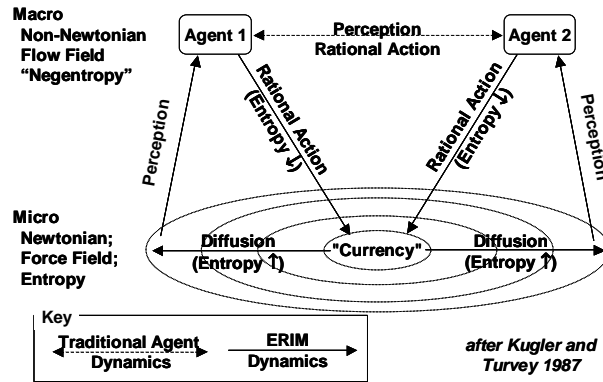


Fig. 2. Environmental processes contribute to agent coordination by aggregating agent interactions into structures, such as gradients, that guide individual agent behavior

3. Emergent Dynamics

What is it?—Classical software is monolithic. Each program addresses a problem or function at the level perceived by the user, and runs on a single thread of control with strong couplings among different modules. Agents present a radically different perspective in which each module (“agent”) has its own thread of control, executes autonomously rather than being explicitly invoked, and is loosely coupled to other agents. User-level functionality is typically not assigned to a single agent, but *emerges* from the interactions of all the agents in a system.

Emergent behavior has two important characteristics. First, it can be much more complex than the behavior of any individual component. (For example, human consciousness emerges from the interactions of much simpler neurons.) Second, it may be qualitatively different than individual behaviors. (A traffic jam moves in the opposite direction to the cars of which it is composed.)

Why is it important?—Like environmental mediation, unmanaged emergent behavior poses a threat to multi-agent systems. This threat is better recognized than that of unmanaged environmental dynamics, but the most common approach is to suppress it by constraining the behavior of individual agents so that the system exhibits only a subset of its total potential behavior [4, 11, 29].

This approach jeopardizes three potential applications of multi-agent systems: simulation, control, and computation.

A common application of multi-agent systems is the *simulation* of physical systems. When those systems consist of interacting components, much of the benefit of an agent approach is the natural mapping of real-world components to individual agents. More often than not, the interactions of the natural components lead to emergent dynamics, so that any agent system intended to model them must be able to generate the same dynamics [15].

Multi-agent systems are proving increasingly useful as a *control* mechanism for distributed systems. The cyberneticists' "law of requisite variety" [1] points out that any control system must have a behavioral repertoire at least as rich as that of the system being controlled. Once again, the emergent nature of the world's dynamics requires that these dynamics not be damped out of the multi-agent system.

Computationally, part of the appeal of a multi-agent system is that the interactions of multiple agents can achieve computations that are formally more complex than those supported by single-agent systems [27]. Kennedy and Eberhart [12] make the extreme case that all cognition is essentially social rather than individual. "*Dumb algorithms* become *smart agents* ... when enhanced by interaction" [28]. Some approaches commonly taken to suppress emergent behavior (for example, serializing interactions) reduce this computational windfall. If we learn to manage emergent behavior, we increase the computational power that a community of agents can provide. Alternatively, we can accomplish a given computational task with simpler individual agents, since much of the information processing now takes place in the interactions *between* agents rather than in computations *within* a single agent.

How does it work?—A fundamental challenge is understanding the relation between individual agent behaviors and the behavior of the system as a whole. Simulation can predict system behavior from specified agent behaviors, but no effective computation is known to derive individual behaviors from desired system behavior.

Physical theory faced a similar crisis a century and a half ago. Classical physics studied matter and energy at the macroscopic level, the level at which we experience nature (thus, the "user level"). The development of atomic theory in the nineteenth century presented a radically different perspective. Familiar macroscopic characteristics of matter such as the distinction among solid, liquid, and gas or the experience of hot and cold disappear when one considers a set of loosely interacting molecules. Nature provided the "simulation" generating macro behavior from micro entities, but there was no analytical mapping, and no evident mechanism for association atomic-level behaviors with an observed macro-level description.

Statistical physics has resulted from the study of the relation between the macroscopic characteristics of matter and the microscopic behaviors of atoms and molecules. It has developed a rich repertoire of system characterizations, concepts, and mathematical tools to bridge the macro/micro gap. These formalisms can be extended to bridge the analogous gap in multi-agent systems. For example:

- A system can exist in different *phases*. The passage from one to another (a *phase transition*) can be reflected in increased computational costs. This particular perspective has already been of great value in understanding computational models such as constraint satisfaction and graph structure [14], and we have detected it in a simple model of multi-agent resource allocation [22].

- Many features of realistic systems result when individual entities that are superficially identical behave differently simply through random effects. This phenomenon, known as *symmetry breaking*, has the effect of distributing entities more widely over a system's state space than one might expect from deterministic arguments alone, and can make a drastic difference in the overall behavior of the system. The existence of such effects in nature suggests a constructive use of stochasticity in multi-agent systems (for example, the use of a roulette wheel to make a weighted but still stochastic choice among several potential actions).
- The trajectory of a complex system through its state space typically has two parts: a *transition* that is not repeated, followed by an *attractor* that describes the system's long-term behavior. The subset of state space that includes a single attractor and all the transitions that lead to it is called a *basin of attraction*, meaning that any trajectory that begins within the basin ends up on the same attractor. Many systems have a number of different basins of attraction, and thus may behave very differently depending on their starting conditions (compare [30]).
- Many phase transitions fall into *universality classes* that are indifferent to details of the material involved. For example, the liquid-gas transition has the same critical exponents in many different materials. These critical exponents depend only on the symmetry and dimensionality of the system, and are independent of other non-trivial characteristics. The notion of a universality class for an agent-based system holds out the hope that generic mathematical formalisms can be applied across a wide range of agent implementations. The physical analogy suggests that mission-critical macroscopic behavior will depend critically on some characteristics of individual agents and their interactions, but not at all on others, and that we can understand these differences in terms of the microscopic and macroscopic characterizations of the system. Our work on a simple model of resource allocation, for example, identifies universal behavior across systems of very different structure [23].
- In statistical mechanics, an *ergodic* system is one that yields the same average value of a system parameter whether the average is taken over time or over the system's state space. This state of affairs reflects a certain "fairness" in the system's trajectory through state space, in the sense that the trajectory visits all regions with comparable frequency. Recognizing the ergodicity (or otherwise) of a multi-agent system has at least two important consequences. First, ergodicity permits us (and non-ergodicity forbids us) to estimate time averages by means of state space averages, which are often much more accessible. Second, various weak search methods such as particle swarm optimization or genetic methods (Section 4) assume that the dynamics of the system will adequately sample the search space, and this assumption depends on whether or not the dynamics are ergodic.
- One of the earliest contributions of statistical mechanics to computer science is the notion of entropy as a measure of statistical disorder. If we index the state of a system by i and represent the probability of finding the system in state i by p_i , the system's entropy is defined by $-\sum p_i \log(p_i)$, where the logarithm is conventionally taken to base 2. Shannon used this concept to formalize the amount of information in a message stream [26]. A number of applications and extensions of this measure are useful in understanding multi-agent systems.
 - Shannon entropy can formalize and make quantitative the Kugler-Turvey model of self-organization outlined in Section 2.3 and Fig. 2 [20].

- The entropy of the message stream among agents is a key universal scaling variable that locates a phase transition in resource allocation [22, 23].
- One may include in the entropy computation not only the relative frequency of tokens in a message stream, but also the range and relative likelihood of the various responses that an agent could take in view of that message stream [9].
- Measuring entropy over a continuous state space requires binning the state space, and bin size can affect the result. This dependency can be overcome by embedding entropy computations in a clustering algorithm, defining a *hierarchical entropy* that has proven useful for measuring the degree of organization in a robot team [2].
- The entropy of a system is subadditive with respect to the entropies of its components. Total entropy is equal to the sum of the individual entropies just when the behaviors of the components are statistically independent of each other. Otherwise it is less than the sum, and the difference is the *joint entropy* or *joint information* of the system, a quantity that is a useful measure of the degree of cooperation exhibited in the system [22].

Currently, we use this perspective from outside the system, looking in. The next step is to learn how an agent can locally monitor these dynamics itself, by monitoring the information flow between itself and the environment (and thus other agents). Such observations would then guide the agent's own behavior, enabling it to ride the emergent dynamics of the environment as a surfer does a wave.

4. Growing vs. Building

What is it?—Recent progress in evolutionary computation makes it feasible to grow agents to satisfy a set of requirements, rather than designing them top-down. There are many varieties of evolutionary computation, including genetic algorithms, genetic programming, evolution strategies, evolutionary programming, and (more remotely) simulated annealing and particle swarm optimization [6, 10]. Their common features include a population of one or more problem solving entities that are evaluated against the problem, then modified on the basis of their performance to define a new population (the “next generation”). The process iterates until it satisfies a stopping criterion (e.g., solution of adequate quality found, no more time, lack of progress).

Why is it important?—Knowledge engineering has long been a major bottleneck in developing intelligent systems. At first glance, our two previous insights appear to reduce this burden, since they both support simple agents with subsymbolic processing. Environmental mediation invites subsymbolic interactions such as pheromones, and the added computational power derived from interaction can make simple agents adequate for complex tasks. Thus the “brains” of our agents are largely driven by structures such as numerical equations or neural networks, rather than semantic networks or rule bases. However, there is no free lunch. Programming subsymbolic agents poses three main challenges: semantic, combinatorial, and telic.

Semantically, determining the form and parameters of a numerical equation, or the structure and weights of a neural network, is non-trivial for a human. In some ways it is more difficult than knowledge engineering. At least superficially, a symbolic sys-

tem invites the human to represent knowledge in terms that are intuitively meaningful, but subsymbolic systems are opaque to intuition.¹ Evolutionary methods are ideally suited to manipulate numerical parameters and formal structures.

Combinatorially, each subsymbolic agent can easily have a hundred or more parameters, and a system may include hundreds or thousands of agents, each with potentially different parameters. Evolutionary methods can rapidly search such large solution spaces.

The telic challenge recognizes that our goal in constructing an agent is not the individual behavior of that agent, but the emergent behavior of the whole system, and there is no known algorithm to derive a specification of individual behavior from overall system behavior. Evolutionary methods bridge this gap by measuring the performance of the population of agents against system-level criteria, and improving that performance step by step until it is good enough.

How does it work?—Growing agent behavior imposes two basic requirements on the designer of an agent-based system: a tractable fitness function and schematic representations.

At each step in an evolutionary computation, the current population must be evaluated against the problem in a way that permits the performance of different members to be compared with one another. The fitness function is the mechanism that does this evaluation. Typically, it is based on a simulator for the problem domain in which agents can be tested.

Once evaluated, the members of the population are selectively modified to produce the next generation. A program in any language can be modified, but in some representations, most modifications are pathological, rendering the agent inoperative. Such representations are not well suited for the growth approach. It is much easier to implement non-pathological modification operators for simple, schematic representations, such as a list of parameters for an equation, or a list of connection weights for a neural network, or an S-expression describing a combination of functional operators of fixed arity. Fine-grained agents lend themselves particularly well to such manipulation.

In our work with synthetic pheromones, a swarm of software agents traveling back and forth between two points generates a path through the environment, based on a numerical equation that combines the strengths of the various pheromones in the environment to define the agent's next step. Hand-tuning the parameters of this equation requires laborious human experimentation over the period of several weeks, and needs to be repeated for different scenarios. The evolutionary solution (Fig. 3) [25] is superior in three ways:

1. The process is independent of the scenario being solved.
2. The evolution takes place in real time, as the agents do the path planning, not in a separate planning phase of the system's operation.

¹ This observation represents an important limitation of fine-grained agents. Applications that generate explanations for humans must at some point invoke a symbolic representation of the problem, at least with current technology. The development of integrated systems that combine our fine-grained emergent approach with more conventional BDI agents that can support generation of explanations is an area of current research.

3. The level of performance attained is an order of magnitude greater than that achieved by hand tuning.

5. Conclusion

Fine-grained agents can accomplish significant computational tasks by taking advantage of three fundamental insights.

1. Active processes in the environment can supplement cognitive processing within individual agents.
2. Interactions among agents produce emergent system-level dynamics that can be understood using methods inspired by statistical mechanics.
3. Agents that exploit the previous two insights can be grown using evolutionary methods, reducing the human labor of designing them and yielding better performance than that attainable by direct human engineering.

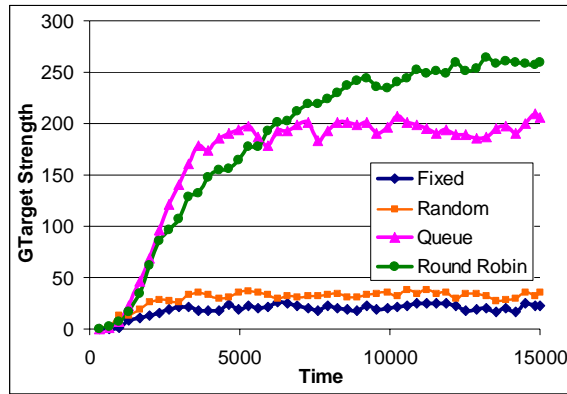


Fig. 3. Performance of evolution on a pheromone system. *Fixed* results are from a population of hand-tuned agents. *Random* are from agents whose parameters are randomly selected around means defined by the hand-tuning process. *Queue* and *Round Robin* show evolved agents, using two different fitness evaluation methods

References

1. Ashby, W. R. Requisite variety and its implications for the control of complex systems. *Cybernetica*, 1(2):83-99, 1958.
2. Balch, T. Hierarchic social entropy: an information theoretic measure of robot team diversity. *Autonomous Robots*, 2000.
3. Brueckner, S. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.
4. Bussmann, S. Agent-Oriented Programming of Manufacturing Control Tasks. In *Proceedings of Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 57-63, IEEE Computer Society, 1998.
5. Cicirello, V. A. and Smith, S. F. Wasp Nests for Self-Configurable Factories. In *Proceedings of Fifth International Conference on Autonomous Agents (Agents 2001)*, 2001.
6. Corne, D., Dorigo, M., and Glover, F., Editors. *New Ideas in Optimisation*. New York, McGraw-Hill, 1999.
7. Ferber, J. and Müller, J.-P. Influences and Reactions: a Model of Situated Multiagent Systems. In *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 72-79, 1996.
8. Grassé, P.-P. La Reconstruction du nid et les Coordinations Inter-Individuelles chez *Belligositermes Natalensis* et *Cubitermes sp.* La théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs. *Insectes Sociaux*, 6:41-84, 1959.

9. Haken, H. *Information and Self-Organization : A Macroscopic Approach to Complex Systems*. 2 ed. Springer, 2000.
10. Jacob, C. *Illustrating Evolutionary Computation With Mathematica*. San Francisco, Morgan Kaufmann, 2001.
11. Jennings, N. R. On Agent-Based Software Engineering. *Artificial Intelligence*, 117:277-296, 2000.
12. Kennedy, J., Eberhart, R. C., and Shi, Y. *Swarm Intelligence*. San Francisco, Morgan Kaufmann, 2001.
13. Kugler, P. N. and Turvey, M. T. *Information, Natural Law, and the Self-Assembly of Rhythmic Movement*. Lawrence Erlbaum, 1987.
14. Martin, O. C., Monasson, R., and Zecchina, R. Statistical Mechanics Methods and Phase Transitions in Optimization Problems. *Theoretical Computer Science*, 265(1-2):3-67, 2001.
15. Moss, S. Editorial Introduction: Messy Systems--The Target for Multi Agent Based Simulation. In Moss, S. and Davidsson, P., Editors, *Multi-Agent-Based Simulation*, pages 1-14. Springer, Berlin, Germany, 2001.
16. Müller, J. P. *The Design of Intelligent Agents*. Berlin, Springer, 1996.
17. Parunak, H. V. D. Manufacturing Experience with the Contract Net. In Huhns, M. N., Editor, *Distributed Artificial Intelligence*, pages 285-310. Pitman, London, 1987.
18. Parunak, H. V. D. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997.
19. Parunak, H. V. D. and Brueckner, S. Ant-Like Missionaries and Cannibals: Synthetic Pheromones for Distributed Motion Control. In *Proceedings of Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 467-474, 2000.
20. Parunak, H. V. D. and Brueckner, S. Entropy and Self-Organization in Multi-Agent Systems. In *Proceedings of The Fifth International Conference on Autonomous Agents (Agents 2001)*, pages 124-130, ACM, 2001.
21. Parunak, H. V. D., Brueckner, S. A., Sauter, J., and Posdamer, J. Mechanisms and Military Applications for Synthetic Pheromones. In *Proceedings of Workshop on Autonomy Oriented Computation*, 2001.
22. Parunak, H. V. D., Savit, R., Brueckner, S. A., and Sauter, J. Experiments in Indirect Negotiation. In *Proceedings of The AAAI Fall 2001 Symposium on Negotiation Methods for Autonomous Cooperative Systems*, 2001.
23. Parunak, H. V. D., Savit, R., Brueckner, S. A., and Sauter, J. A Technical Overview of the AORIST Project. ERIM, Ann Arbor, MI, 2001. Available at http://www.erim.org/cec/projects/aorist/AORIST_Snapshot_0104.pdf.
24. Peeters, P., Valckenaers, P., Wyns, J., and Brueckner, S. Manufacturing Control Algorithm and Architecture. In *Proceedings of Second International Workshop on Intelligent Manufacturing Systems*, pages 877-888, K.U. Leuven, 1999.
25. Sauter, J., Parunak, H. V. D., Brueckner, S. A., and Matthews, R. Tuning Synthetic Pheromones With Evolutionary Computing. In *Proceedings of Genetic and Evolutionary Computation Conference Workshop Program, 2001*, pages 321-324, 2001.
26. Shannon, C. E. and Weaver, W. *The Mathematical Theory of Communication*. Urbana, IL, University of Illinois, 1949.
27. Wegner, P. Why Interaction is More Powerful than Algorithms. *Communications of the ACM*, 40(5 (May)):81-91, 1997.
28. Wegner, P. Interactive Foundations of Computing. *Theoretical Computer Science*, 192(2):315-351, 1998.
29. Wooldridge, M. J. and Jennings, N. R. Pitfalls of Agent-Oriented Development. In *Proceedings of 2nd Int. Conf. on Autonomous Agents (Agents-98)*, pages 385-391, 1998.
30. Wuensche, A. and Lesser, M. *The Global Dynamics of Cellular Automata*. Second ed. 2000.