Forthcoming in F. Bergenti, M.-P. Gleizes, and F. Zambonelli, eds., Methodologies and Software Engineering for Agent Systems. Kluwer, 2004.

Engineering Swarming Systems

H. Van Dyke Parunak and Sven A. Brueckner
Altarum Institute
{van.parunak, sven.brueckner}@altarum.org

Abstract

Most multi-agent systems are inspired by classical AI, whose objective was to realize human-level intelligence in a computer. As the field has moved toward multiple agents, there has been a presumption that individual agents still aspire to high-level intelligence. Swarming systems follow an alternative model, inspired more by artificial life than artificial intelligence. The individual agents in these systems may be non-cognitive, but complex, robust cognition emerges from their interactions. This paper defines swarming and the concepts of self-organization and emergence that underlie it. It describes the kinds of problems for which it is well suited, explores why it functions, and outlines some initial principles of an engineering methodology for developing artificial swarming systems.

1 WHAT is Swarming?

We define swarming as "useful self-organization of multiple entities through local interactions." We begin by reviewing other definitions, then focus in on organization and self-organization, and the relation of these concepts with emergence.

1.1 Swarming

Definitions of swarming have been proposed by insect ethologists, roboticists, and military historians. Of the many definitions that have been proposed, a few will illustrate the main themes.

Students of biological systems use it to describe decentralized self-organizing behavior in populations of (usually simple) animals [9, 10, 14, 38]. Swarming has been defined (e.g., [10]) as "distributed problem-solving devices inspired by collective behavior of social insect colonies and other animal societies." Table 1 lists a few examples that have been studied.

Table 1: Some Examples of Swarming in Nature

Swarming Behavior	Entities
Pattern Generation	Bacteria, Slime Mold
Path Formation	Ants
Nest Sorting	Ants
Cooperative Transport	Ants
Food Source Selection	Ants, Bees
Thermoregulation	Bees
Task Allocation	Wasps
Hive Construction	Bees, Wasps, Hornets, Termites
Synchronization	Fireflies
Feeding Aggregation	Bark Beetles
Web Construction	Spiders
Schooling	Fish
Flocking	Birds
Prey Surrounding	Wolves

The use of the term to describe artificial systems can be traced to Beni, Hackwood, and Wang in the late 1980's [4-7, 26, 27]. Their work focuses on populations of cellular robots, and they use the term to describe self-organization through

local interactions. In the context of unpiloted air vehicles (UAV), Clough defines a swarm as a "collection of autonomous individuals relying on local sensing and reactive behaviors interacting such that a global behavior emerges from the interactions" [16]. He distinguishes swarming (resulting from reactive behaviors of simple homogeneous entities performing simple tasks) from the emergent behavior of heterogeneous teams of deliberative entities performing complex tasks.

Recently, "swarming" has come into vogue in the military to describe a battlefield tactic that involves decentralized, pulsed attacks [2, 20, 21, 30]. Military historians focus less on the process of self-organization and more on the resulting organization itself: "the systematic pulsing of force and/or fire by dispersed, internetted units, so as to strike the adversary from all directions simultaneously" [2]; a "scheme of maneuver" consisting of "a convergent attack of several semi-autonomous (or autonomous) units on a target" [21]. The connection with insect applications is not coincidental. Insect self-organization is robust, adaptive, and persistent, as anyone can attest who has tried to keep ants out of the kitchen or defeat a termite infestation, and military commanders would love to be able to inflict the frustration, discomfort, and demoralization that a swarm of bees can visit on their victims. The linkage between swarming and warfare is ancient. In the Bible, God promises to demoralize the indigenous population of Canaan before the invading Israelites in the words, "I will send the hornet before you" (Exodus 23:28; cf. Deuteronomy 7:20; Joshua 24:12). In the eighteenth dynasty (1550 BC), the ancient Egyptians awarded military heroes a gold and silver medal in the form of a stylized fly (Figure 1) [29], and there is evidence that the ancients sometimes hurled hives of stinging insects against their enemies [35].

For the purpose of this paper, we will define swarming as "useful self-organization of multiple entities through local interactions." This definition highlights elements of the others that have been suggested.

"Useful" emphasizes that we are interested in engineering systems that are answerable to someone outside of the system boundary for their behavior. Some forms of self-organized behavior, such as riots and oscillation, might be interesting to a biologist, but undesirable in a commercial or military application.

Figure 1: Egyptian "fly" medal for military heroes, 1550 BC (National Gallery of Art)

Self-organization is most prominent in the robotic definitions, since the concern there is to distinguish swarming from conventional top-down control schemes. The military definition does not emphasize self-organization, perhaps because of a historic tradition of top-down centralized control. We do not require that the self-organization result from reactive rather than deliberative individual behavior. Thus our definition includes not only Clough's "swarms" but also his "teams," if they meet the other terms of the definition.

The notion of **multiple entities** is common to all definitions, and indeed is intrinsic to the common-sense use of the term. A major motivator for swarming is the proliferation of autonomous platforms, such as vehicles, communications systems, and sensor systems. Although

these systems are often referred to as "unmanned," in current practice it would be more accurate to describe them as "remotely manned." The flight crew for a Predator UAV consists of two people. Housing them in a control van rather than on board the flying platform considerably reduces their risk, but does not reduce the manpower requirements for fielding the vehicle. A major promise of swarming is multiplying the number of platforms that a single person can effectively control.

Our focus on **local interactions** has two motivations: a need and a promise. The *need* is a growing concern about communication congestion. The *promise* is the observation that local interactions suffice to maintain long-range coordination in biological systems, so that we ought to be able to reverse-engineer the underlying mechanisms for use in synthetic systems.

1.2 Organization

As used in expressions such as "self-organization," the word "organization" has at least three distinct, but related, meanings: it can refer to a mapping, a process, or a structure.

Organization$_1$ is a *mapping* from a system to an ordered set, e.g.,

Such a mapping permits us to say that one system is "more organized" than another (or than the same system at a different time).

Different detailed definitions for this mapping are possible. Common themes will include *entropy* and *symmetry*, as illustrated in Figure 2.

Denote a system by an upper-case letter, and its elements as the same letter in lower-case, indexed. Thus $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_k\}$ denote two systems. The entropy of a system A is denoted by S(A). With these concepts, we can meaningfully assert Org(A) > Org(B) if

- S(B) > S(A) or
- B has a higher order of symmetry than A

Entropy can be computed against different bases, such as the spatial distribution of agents, their directions of movement, the behaviors open to them at any moment, or the time series generated by their actions. This variety can lead to the concern that entropy, and thus organization$_1$, is in the eye of the beholder. In fact, methods exist for defining changes in entropy in an unambiguous way [17, 46], though discussing them in detail is beyond the scope of this survey.

Organization$_2$ is a *process* in a single system in which Organization$_1$ increases with time:

$$\mathrm{Org}(A(t_2)) > \mathrm{Org}(A(t_1)),\ t_2 > t_1$$

Organization$_3$ is the *structure* resulting from Organization$_2$, and can be measured with Organization$_1$.

1.3 Self-Organization and Emergence

With this understanding of "organization," it would seem natural to define "self-organization" as a process (Organization$_2$) that reduces the entropy

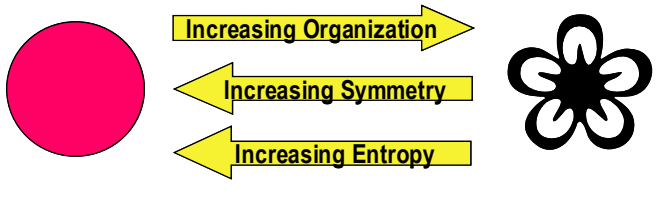


Figure 2: Symmetry vs. Organization

of a system without external intervention (motivating the modifier "self"). This definition is in line with some that have been proposed in the literature, for example:

> Camazine [14]: "Pattern formation occurs through interactions internal to the system, without intervention by external directing influences (leaders, blueprints, recipes, templates)"
>
> Bonabeau [10]: "A set of dynamical interactions whereby structures appear at the global level of a system from interactions among its lower-level components. … The rules specifying the interactions are executed on the basis of purely local information, without reference to the global pattern."

These definitions emphasize the system boundary (through terms such as "local" and "internal"). The second includes three other concepts as well:

- A distinction of multiple levels within a system
- "Interactions" among entities at lower levels of the system
- The "appearance" or "emergence" of properties and structures at higher levels from these interactions

Other definitions of "self-organization" rely only on these three themes, without focusing on the system boundary, for example:

> Biebricher [8]: The process by which individual subunits achieve, through their cooperative interactions, states characterized by new, emergent properties transcending the properties of their constitutive parts
>
> Schweitzer [45]: The emergence of new system properties not readily predicted from the basic equations."

To achieve greater precision, we propose distinguishing between self-organization and emergence on the basis of the contrast between the horizontal concept of system boundary and the vertical concept of levels (Figure 3).

We define **Self-Organization** as organization Among elements *within a level*

- *Without information flow* across the boundary.

The second law of thermodynamics demands that there be *energy flow* across the boundary of any system whose organization increases over time. *Self*-organization requires that this energy flow not contain information. This definition depends critically on the location of the *system boundary*. If the boundary is moved, a system's character as self-organizing or not may change.

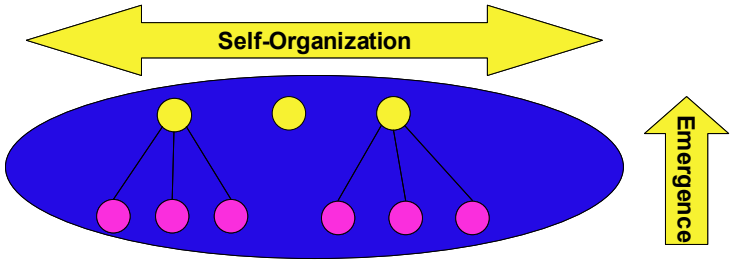


Figure 3: Comparing Self-Organization and Emergence

We define **Emergence** as a subcategory of self-organization (Figure 4). Emergence (as we use the term) describes the appearance of structures at a *higher level* that are not explicitly represented in *lower-level* components. The reliance of swarming systems on locally available information makes it difficult for them to reason explicitly about higher-level structures, so emergence tends to be an important mechanism in swarming systems.

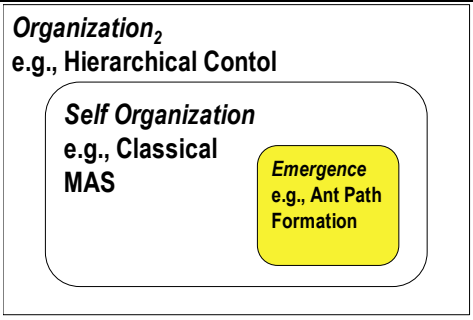


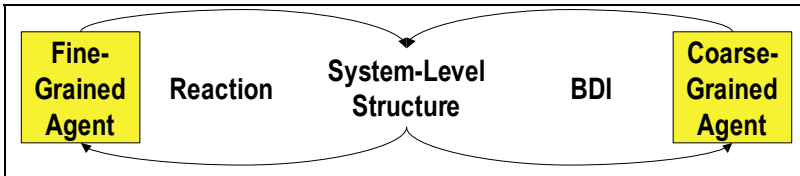
Figure 4: Emergence as a Subcategory of Self-Organization

Neither self-organization nor emergence is necessarily good. The formation of structures will correspond to a reduction in entropy, whether those structures support or frustrate the objectives of the system stakeholders. The fact that emergent structures can be pathological (as in the case of race conditions or herding behavior) may explain the apprehension with which some people view emergence. For example, Wooldridge and Jennings assert [48], "Emergent functionality is akin to chaos …." They urge engineers of agent systems to "severely restrict the way in which agents can interact with one-another … ensure that there are few channels of communication between agents … restrict the *way* in which agents interact" in order to reduce the likelihood of emergent behavior. A consequence of this restriction is that any desired system-level behavior must be explicitly represented in the lower-level components, a requirement that is difficult to meet if the system's requirements include responding gracefully to unanticipated changes in its environment. Our alternative approach is to develop principles for designing and developing systems whose emergent behavior is beneficial or at least benign.

This difference in vision leads to two distinct approaches to building multi-agent systems (Figure 5).

- Classical multi-agent systems achieve self-organization through deliberation among fairly sophisticated ("coarse-grained") agents. Emergent systems can use much simpler reactive agents.
- Reasoning in emergent self-organization is often non-symbolic, while classical systems are usually symbolic.
- Because of the need for representing system-level behavior explicitly at all layers, non-emergent systems are best suited for exploiting well-known environments. The ability of emergent systems to produce new behaviors is appropriate for more exploratory problems.

A front line of our current research is understanding how to hybridize these two families of systems.



Emergence	Deliberation
Non-Symbolic	Symbolic
Exploration	Exploitation

Figure 5: Two Families of Self-Organizing Systems

1.4 Alternatives to Self-Organization

Camazine [14] identifies four alternatives to self-organization: leaders, recipes, blueprints, and templates.

- A *leader* is a single agent that receives status information from the other agents, decides on the action that each should take, and issues commands. This paradigm is sometimes called "centralized control." If the leader is not part of the system, the flow of the leader's commands to the other agents crosses the system boundary.
- A *recipe* is a script (a process description that is sequenced in time) that is constructed by the system's designer and installed at compile-time. If the designer is not part of the system, this script crosses the system boundary when it is installed.
- A *blueprint* is a map (a spatial prescription) that is constructed by the system's designer and installed at compile-time. If the designer is not part of the system, this map crosses the system boundary when it is installed.
- A *template* is a structure in the system's environment (e.g., the walls of a soccer arena) that constraints the system's behavior. If this structure is not part of the system, information about it crosses the system boundary as the system interacts with it.

It is useful to keep these alternatives in mind for two reasons. First, self-organization is not the best answer for every problem. In some cases, an alternative approach may be preferable, and responsible engineering requires awareness of these alternatives. Second, in each case, the system can be made self-organizing by expanding the system boundaries to include the source of the information. Thus many agent architectures include a (software) leader that directs the actions of the other agents. The engineer must carefully specify the system boundary, and may be able to adjust the behavior of the system significantly by shifting the boundary.

2 WHERE would you want to use Swarming?

Five domain features indicate the appropriateness of swarming: discreteness, deprivation, distribution, decentralization, and dynamism.

2.1 Discrete

It is easiest to apply agents (whether self-organizing or not) to a domain if the domain consists of discrete elements that can be mapped onto the agents. Some forms of organization$_3$ also are achieved most naturally in a discrete system, for example, those that are characterized as a graph structure of some sort.

2.2 Deprived (Resource-Constrained)

We say that a system is "deprived" (or resource constrained) when limits on resources (such as processing power, communications bandwidth, or storage) rule out brute-force methods. For instance, if enough communications bandwidth is available, every agent can communicate directly with every other agent. If agents have enough processing power, they can reason about the massive input they will receive from other agents. If they have enough storage, they can maintain arbitrarily large sets of instructions telling them what to do in each circumstance.

Under such assumptions, swarming architectures would seem to have little benefit. Some futurists extrapolate the historically exponential increases in hardware processing power, storage, and bandwidth, and claim that these constraints will quickly disappear. At the hardware level, Moore's law and its analogs for bandwidth and storage give good reason to be optimistic. However, a computer system is more than hardware. It is constrained by theoretical, psychological, commercial, and physical issues as well. For example:

- No matter how much storage is available, the knowledge engineering effort required to construct large knowledge bases remains a formidable *psychological* obstacle to completely defining the behavior of every agent.
- No matter how fast processors get, the *theory* of NP-completeness points out that the time required to solve reasonably-sized problems in many important categories will still be longer than the age of the universe. An important instance of this challenge is the truth maintenance problem, the challenge of detecting inconsistencies in a knowledge base that result from changes in the world, which is NP-hard for reasonably expressive logics.
- No matter how much bandwidth the hardware can support, the *market* may not make it available in the configuration needed for a specific problem. Military planners, for instance, have long counted on the availability of commercial satellite channels, but the commercial market has moved toward land-based fiber backbones, resulting in a major shortfall in projected available bandwidth for military deployments in underdeveloped areas.
- The growing emphasis on Pervasive Computing and nanotechnology requires the deployment of computation on very small devices. The *physical* limitations of such devices will not permit them to support the level of processing, storage, and communications that can be realized on unconstrained devices.

Several characteristics of swarming systems make them good candidates for deprived environments. For example,

- Interactions among system components are typically local. If information needs to move long distances, it does so by propagation rather than direct transfer. Local interactions limit the number of neighbors about whom each agent must reason at a time, and enable the use of low-power transmissions that permit bandwidth to be reused every few kilometers.
- Because system-level behaviors do not need to be specified at the level of each element, the knowledge engineering and storage requirements are greatly reduced.
- Emergent systems commonly maintain information by continuously refreshing current information and letting obsolete information evaporate. This process guarantees that inconsistencies remove themselves within a specified time horizon, without the need for complex truth-maintenance procedures.

2.3 Distributed

The notion of "local interactions" is central to our definition of swarming. Keeping interactions local is a powerful strategy for dealing with deprived systems, but requires that the entities in the problem domain be distributed over some topology within which interactions can be localized.

The most common topology is a low-dimensional Euclidean manifold, or a graph that can be embedded in such a manifold. For example, insect stigmergy takes place on physical surfaces that, at least locally, are embedded in two-dimensional manifolds. Most engineered applications of swarming such as path planning [42], pattern recognition [13], sensor network self-organization, and ant-colony optimization [18], follow this pattern. In these applications, locality can be defined in terms of a distance metric, and enforced by physical constraints on communications (e.g., a node's neighbors are all the other nodes with whom it has radio contact).

More recent work (for instance, in telecommunications [28], or in our laboratory, on semantic structures) successfully mediates agent interactions via scale-free small-world graphs. Such graphs have long-range shortcuts and so are typically not embeddable in low-dimensional manifolds. These shortcuts pose problems for classical definitions of distance, but locality of interaction can still be defined in terms of nearest-neighbor graph connectivity, and the empirical success of these latter efforts shows that this form of locality is sufficient to achieve coordination.

2.4 Decentralized

As a system characteristic, decentralization is orthogonal to distribution. In a centralized system, all transactions require the services of a single distinguished element. If the system is not distributed, the central point and the system are identical. If it is distributed, the central point is one of the elements, with which the others must communicate. A common extension of centralization in a distributed system is the hierarchy, in which the central element for a small group of nodes joins with other nodes at its level in reporting to a yet higher central element, and so on until the top node is reached.

Swarming can be a poor choice for applications that require centralization. The restriction to local interactions means that communications between peripheral elements and the central element is an emergent behavior of the system, which may not meet the quality of service requirements or the need for detailed predictability that often lead to a requirement for central control. However, systems designers should be cautious about accepting a centralized architecture. Such architectures have at least three weaknesses.

1. They are inherently resistant to increases in *scale*. As the system grows, the capacity of the central element must also grow. In decentralized approaches, new elements can be added without changing any of the existing elements.
2. A frequent role of the central element is to mediate interactions among lower-level nodes (as in the mediator architecture [23]). This technique may actually lengthen the communication path between two nodes, leading to undesirable *delays* as messages travel up, then back down, the hierarchy.
3. The central element and the communication paths leading to it are vulnerable to attack or failure, making the system less *robust* than a swarming system.

Centralized architectures often result more from tradition than from absolute system requirements, and a growing body of cases suggests that acceptable functionality can be achieved, with improved scalability, timeliness, and robustness, in a decentralized way. In addition, centralization is impossible in some cases (such as achieving coordination among a population of entities whose members are not known in advance and who do not all have access

to a common element). Swarming techniques are a natural candidate for implementing decentralized architectures.

2.5 Dynamic

A system is dynamic if its requirements change during its lifetime. The emergent behavior that is characteristic of swarming is a powerful way for dealing with changed requirements. The system elements do not need to encode the system-level behavior explicitly, and so do not need to be modified when those requirements change. Three aspects of such change affect the need for emergence: scope, speed, and obscurity.

Scope characterizes the amount of change to which a system's requirements are susceptible. The less the scope of change, the more likely it is that the system as originally configured will deliver acceptable performance. The greater the degree of change, the more value there is in the ability of the elements to reorganize to produce new emergent behaviors that were not active in the initial configuration.

Speed characterizes the rapidity of change, and affects the desirability of swarming by way of the distinction between centralized and decentralized architectures. If the system changes slowly, non-swarming techniques that rely on centralized organizations can tolerate the time delays imposed by hierarchical communications. As the rate of change begins to outpace the communications time through the hierarchy, centralized organizations find themselves perpetually providing the answers to yesterday's problems, and unable to respond rapidly enough. A common response is to flatten the organization and empower lower-level nodes to act on local information, essentially moving toward a swarming architecture.

Obscurity reflects the degree to which later requirements can be anticipated by the original designer. Even if changes are rapid and wide in scope, if they follow along the lines anticipated by the designer, simple parameter adjustments in a non-emergent architecture may be able to cope with them. Swarming systems are much better at enabling a system to satisfy requirements that would be surprising to its original designer.

3 WHY does Swarming work?

Swarming is a discovery, not an invention. It is a naturally occurring phenomenon that we seek to imitate in engineered systems. Design principles for effective artificial swarming systems must be developed from an understanding of why swarming works in natural systems.

We analyze these underlying principles of swarming in terms of three restrictions on the space of all possible multi-process systems, outlined in Figure 6.

- The various processes must be *coupled* with one another so that they can interact.
- This interaction must be self-sustaining, or *autocatalytic*. Autocatalysis enables self-

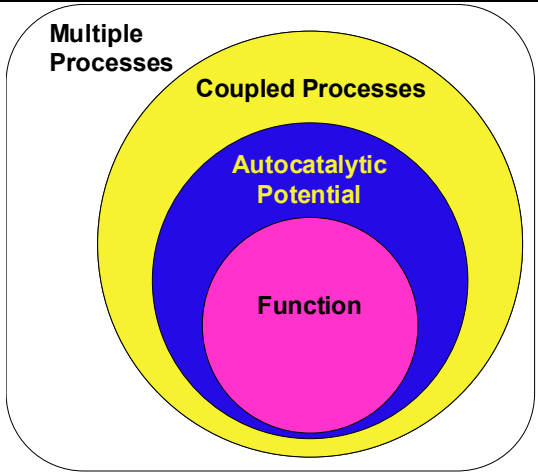


Figure 6: Three Enablers for Swarming Systems

organization, but it is not necessarily useful.

- The self-organizing system must produce *functions* that are useful to the system's stakeholders.

In discussing each of these, we first review the concept and its mechanisms, then discuss design principles to which it leads.

Table 2: Categories of Information Exchange

		Topology of Inter-Agent Relationships	
		Centralized (between Distinguished and Subordinate agents)	**Decentralized** (among Peer agents)
Information Flow	**Direct** (messages between agents)	***Construction*** (Build-Time) ***Command*** (Run-time)	***Conversation***
	Indirect (non-message interaction)	***Constraint***	***Stigmergy*** (generic) ***Competition*** (limited resources)

3.1 Coupled Processes

3.1.1 How Processes can be Coupled

Agents must exchange information if they are to self-organize. Different patterns of information exchange are possible, and can be classified along two dimensions: Topology and Information Flow (Table 2).

The Topology dimension depends on two different kinds of relations that agents can have with one another. When the agents can say "No" to one another within the rules of the system, they are "peer agents." When one of them (say agent A) can say "No" to the other (B), but B cannot say "No" to A, we call A the "distinguished agent" and B the "subordinate." The relationship between two agents may be fairly fixed (for example, the relationship between a human programmer and her software agent). Or it may vary over time (as when peer agents negotiate a work plan that calls for one of them to supervise the other, resulting in a distinguished-subordinate relationship during execution). These concepts can be developed more formally through dependency and autonomy theory [15, 37]. Centralized information exchange is between a distinguished and a subordinate agent, while decentralized information exchange is between peer agents.

The Information Flow dimension relies on environmental state variables that the agents can manipulate and sense. All information exchange is ultimately mediated by the environment, but the role of the environment is sometimes not modeled explicitly. The information flow from one agent to another is Direct if no intermediate manipulation of information is modeled, and Indirect if it is explicitly modeled.

Centralized mechanisms all involve communication between the distinguished agent and its subordinates. This flow may be direct (when the distinguished agent *constructs* or *commands* the subordinates) or indirect (when the distinguished agent *constrains* the subordinates by manipulating exogenous environmental variables visible to the subordinates). In correlation through command, used commonly in robot soccer, holonic manufacturing, and some simulation applications, agents behave much like objects, executing methods invoked by incoming messages. The focal point algorithm advocated by [22] and the common utility functions implicit in [24] both rely on construction (common programming). In indirect centralized mechanisms, subordinates jointly sense changes in a shared exogenous environmental variable. The variable's

dynamics are independent of agent actions, so it cannot move information between subordinates. But it may serve as a synchronizing signal that correlates the agents' actions. The experimenter who configures targets and obstacles in an experimental testbed is constraining the subordinates, supporting correlation through indirect centralized action.

Decentralized mechanisms all involve communication among peers. Most negotiation research focuses on direct peer-to-peer information flows ("conversation"). Indirect decentralized flows occur when peers make and sense changes to environmental variables. This class of coordination is called "stigmergy," [25], from the Greek words *stigma* "sign" and *ergon* "work": the work performed by agents in the environment guides their later actions. The information stored in the environment forms a field that supports agent coordination, leading to the term "co[ordination]-field" for this class of technique [33]. Such techniques are common in biological distributed decentralized systems such as insect colonies [38]. A common form of stigmergy is resource *competition*, which occurs when agents seek access to limited resources. For example, if one agent consumes part of a shared resource, other agents accessing that resource will observe its reduced availability, and may modify their behavior accordingly. Even less directly, if one agent increases its use of resource A, thereby increasing its maintenance requirements, the loading on maintenance resource B may increase, decreasing its availability to other agents who would like to access B directly. In the latter case, environmental processes contribute to the dynamics of the state variables involved.

Different varieties of stigmergy can be distinguished. One distinction concerns whether the signs consist of special markers that agents deposit in the environment ("marker-based stigmergy") or whether agents base their actions on the current state of the solution ("sematectonic stigmergy"). Another distinction focuses on whether the environmental signals are a single scalar quantity, analogous to a potential field ("quantitative stigmergy") or whether they form a set of discrete options ("qualitative stigmergy"). As shown in Table 3, the two distinctions are orthogonal.

Stigmergic mechanisms have a number of attractive features, particularly for swarming systems.

Simplicity.—The logic for individual agents is much simpler than for an individually intelligent agent. This simplicity has three collateral benefits.

1. The agents are easier to program and prove correct at the level of individual behavior.
2. They can run on extremely small platforms (such as microchip-based "smart dust" [43]).
3. They can be trained with genetic algorithms or particle-swarm methods rather than requiring detailed knowledge engineering.

Table 3: Varieties of Stigmergy

	Marker-Based (Artificial signs inserted in the domain)	**Sematectonic** (Domain elements only)
Quantitative (Scalar quantities)	Gradient following in a single pheromone field	Ant cemetery clustering
Qualitative (Symbolic distinctions))	Decisions based on combinations of pheromones	Wasp nest construction

Scalable.—Stigmergic mechanisms scale well to large numbers of entities. In fact, unlike many intelligent agent approaches, stigmergy *requires* multiple entities to function, and performance typically improves as the number of entities increases. Stigmergy facilitates scalability because the environment imposes locality on agent interactions.

Agents interact with the environment only in their immediate vicinity. Increases in the number of agents are typically associated with an extension of the environment. The density of agents over the environment, and thus the processing load on each agent, usually does not increase.

Robustness.—Because stigmergic deployments favor large numbers of entities that are continuously organizing themselves, the system's performance is robust against the loss of a few individuals. Such losses can be tolerated economically because each individual is simple and inexpensive.

Environmental Integration.—Explicit use of the environment in agent interactions means that environmental dynamics are directly integrated into the system's control, and in fact can enhance system performance. A system's level of organization is inversely related to its symmetry (Figure 2), and a critical function in achieving self-organization in any system made up of large numbers of similar elements is breaking the natural symmetries among them [3]. Environmental noise is usually a threat to conventional control strategies, but stigmergic systems exploit it as a natural way to break symmetries among the entities and enable them to self-organize.

We make extensive use of stigmergy in our applications, building on the theoretical foundation and pheromone infrastructure outlined in [11].

3.1.2 Design Principles Derived from Coupled Processes

Coupling 1: Use a distributed environment.—Stigmergy is most beneficial when agents can be localized in the environment with which they interact by sensing and acting. A distributed environment enhances this localization, permitting individual agents to be simpler (because their attention span can be more local) and enhancing scalability.

Coupling 2: Use an active environment.—If the environment supports its own processes, it can contribute to overall system operation. For example, evaporation of pheromones in the ants' environment is a primitive form of truth maintenance, removing obsolete information without requiring attention by the agents who use that information.

Coupling 3: Keep agents small.—Agents should be small in comparison with the overall system, to support locality of interaction. This criterion is not sufficient to guarantee locality of interaction, but it is a necessary condition. The fewer agents there are, the more functionality each of them has to provide, and the more of the problem space it has to cover.

Coupling 4: Map agents to Entities, not Functions.—Choosing to represent domain entities rather than functions as agents takes advantage of the fact that in our universe, entities are bounded in space and thus have intrinsic locality. Functions tend to be defined globally, and making an agent responsible for a function is likely to lead to many non-local interactions. For example, in a factory, each machine (an entity) has fairly local interactions with other machines, parts, and workers in its area of the plant, but a function (such as scheduling) must take into account all of the machines in the entire plant.

3.2 Autocatalytic Potential

3.2.1 What is Autocatalysis?

The concept of autocatalysis comes from chemistry. A catalyst is a substance that facilitates a chemical reaction without being permanently changed. In autocatalysis, a product of a reaction

serves as a catalyst for that same reaction. An autocatalytic set is a set of reactions that are not individually autocatalytic, but whose products catalyze one another. The result is that the behaviors of the reactions in the set are correlated with one another. If reaction A speeds up (say, due to an increased supply of its reagents), so does any reaction catalyzed by the products of A. If A slows down, so do its autocatalytic partners. This correlation causes a decrease in the entropy of the overall set, as measured over the reaction rates, so we would describe such a system as self-organizing. Figure 7 summarizes these concepts using reaction schemata.

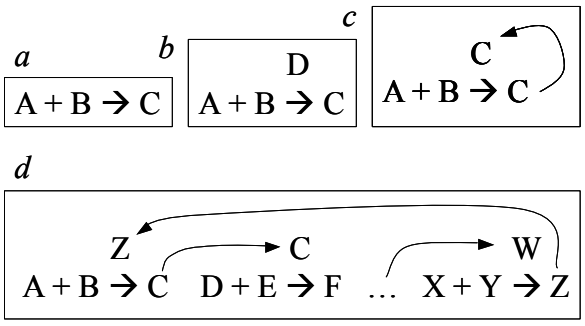


Figure 7: Relations among Processes.--a) A simple reaction. b) D catalyzes the conversion of A and B to C. c) An autocatalytic reaction. d) An autocatalytic set of processes (shown as a ring, but other topologies are possible).

Not all processes that are coupled, are autocatalytic. Autocatalyticity requires a continuous closed flow of information among the processes to keep the system moving. If the product of process A catalyzes process B, but process B's products have no effect (either directly or indirectly) on process A, the system is not autocatalytic. Furthermore, a system might be autocatalytic in some regions of its state space but not in others.

It is natural to extend this concept from chemistry to any system of interacting processes, such as a multi-agent system. A set of agents has *autocatalytic potential* if in some regions of their joint state space, their interaction causes system entropy to decrease (and thus leads to increased organization). In that region of state space, they are *autocatalytic*.

Figure 8 exhibits two different approaches to achieving the closed information flow that supports autocatalysis in multi-agent systems.

- The dashed line between the two agents represents conventional agent interactions: the agents perceive one another, reason about how to coordinate their activities, and then act. The information flow in this case is maintained by the agents' perception of one another. As the

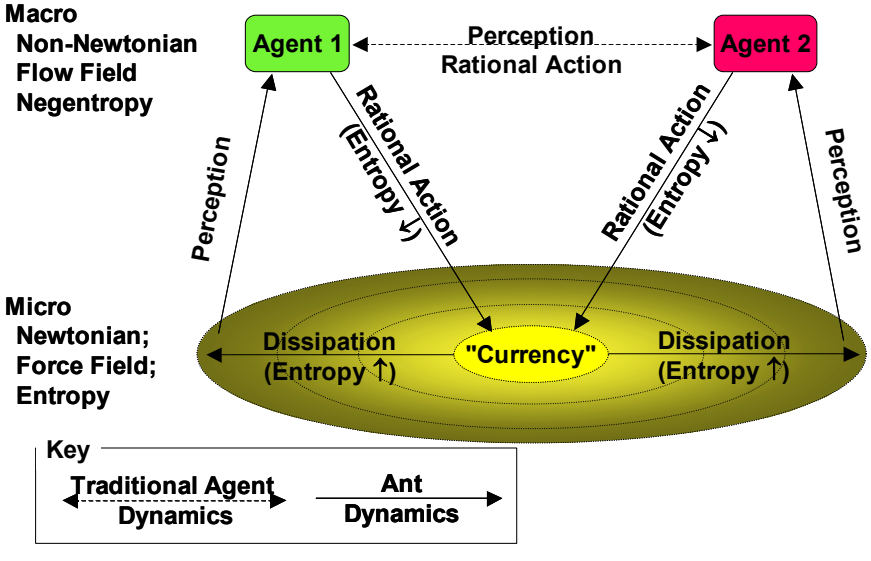


Figure 8: Autocatalytic Flows in Pheromone Dynamics

number of agents increases, this approach requires an increasing amount of processing power on the part of each agent. If an agent is unable to sense or be sensed by other agents in the system, the information flow is broken, and the system's ability to self-organize will be reduced.

- The stigmergic approach used in swarming is represented by the solid lines forming a triangle of "Rational Action," "Dissipation," and "Perception." Agents deposit a "currency" (pheromone in the case of ants; money in the case of a market) into a shared environment. The aggregation of this currency from multiple deposits, and dissipative forces in the environment (evaporation in the case of pheromones), generate gradients that each agent can sense and to which it can respond. This continuous cycle provides the information flow that keeps the processes coordinated. Because the environment integrates the information from each agent, and because agents need sense only their local environment, the number of agents can increase without requiring more processing power on the part of each agent.

Two points are important to understand about autocatalyticity.

1. In spite of the reduction of entropy, autocatalyticity does not violate the Second Law of Thermodynamics. The rationalization is most clearly understood in the stigmergic case (Figure 8). Entropy reduction occurs at the macro level (the individual agents), but the dissipation of pheromone at the micro level generates more than enough entropy to compensate. This entropy balance can actually be measured experimentally [39].
2. Information flows are necessary to support self-organization, but they are not sufficient. A set of coupled processes may have a very large space of potential operating parameters, and may achieve autocatalyticity only in a small region of this space. Nevertheless, if a system does not have closed information flows, it will not be able to maintain self-organization.

3.2.2 Design Principles Derived from Autocatalysis

Autocatalysis 1: Think Flows rather than Transitions.—Our training as computer scientists leads us to conceive of processes in terms of discrete state transitions, but the role of autocatalysis in supporting self-organization urges us to pay attention to the flows of information among them, and to ensure that these flows include closed loops.

Autocatalysis 2: Boost and Bound.—Keeping flows moving requires some mechanism for reinforcing overall system activity. Keeping flows from exploding requires some mechanism for restricting them. These mechanisms may be traditional positive and negative feedback loops, in which activity at one epoch facilitates or restrains activity at a successive one. Or they may be less adaptive mechanisms such as mechanisms for continually generating new agents and for terminating those that have run for a specified period ("programmed agent death").

Autocatalysis 3: Diversify agents to keep flows going.—Just as heat will not flow between two bodies of equal temperature, and water will not flow between two areas of equal elevation, information will not flow between two identical agents. They can send messages back and forth, but these messages carry no information that is new to the receiving agent, and so cannot change its state or its subsequent behavior. Maintaining autocatalytic flows requires diversity among the agent population. This diversity can be achieved in several ways. Each agent's *location in the*

environment may be enough to distinguish it from other agents and support flows, but if agents have the same movement rules and are launched at a single point, they will not spread out. If agents have different experiences, *learning* may enable them to diversify, but again, reuse of underlying code will often lead to stereotyped behavior. In general, we find it useful to incorporate a *stochastic element* in agent decision-making. In this way, the decisions and behaviors of agents with identical code will diversify over time, breaking the symmetry among them and enabling information flows that can sustain self-organization.

3.3 Function

3.3.1 How Systems Adjust to Required Function

Process interaction must support autocatalysis if a system is to support ongoing self-organization. From an engineering perspective, a further step is necessary. Self-organization in itself is not necessarily useful. Autocatalysis might sustain undesirable oscillations or thrashing in a system, or keep it locked in some pathological behavior. We want to construct systems that not only organize themselves, but that yield structures that solve some problem we need to address. There are two broad approaches to this problem, broadly corresponding to the distinction in classical AI between the scruffy and the neat approaches. It is likely that as the use of self-organizing systems matures, a hybrid of both approaches will prove necessary.

One approach, exemplified by work in amorphous computing [1, 34], is to build up, by trial and error, a set of programming metaphors and techniques that can then be used as building blocks to assemble useful systems.

An alternative approach is to seek an algorithm that, given a high-level specification for a system, can compute the local behaviors needed to generate this global behavior. State-of-the-art algorithms of this sort are based not on *design*, but on *selection*. Selection in turn requires a system with a wide range of behavioral potential, and a way to exert pressure to select from this wide range of behaviors the ones that are actually desired.

One way to ensure a broad range of behavioral potential is to construct nonlinear systems that can exhibit formally chaotic behavior. From a classical engineering perspective, chaos is undesirable because it is unpredictable in the long range. However, from an emergent perspective, chaos is desirable because it offers a simple way to sample a broad subset of the system's space of possible behaviors.

We can illustrate this somewhat nonintuitive insight with the simple logistic equation

$$x_{t+1} = g\, x_t\, (1 - x_t)$$

for $0 \le x \le 1$ and $1 \le g \le 4$. Figure 9 shows a plot of the 201st through 500th iterates of this function, starting at $x = 0.5$, for various values of g. The plot has three distinct regions.

- For $g < 3$, x converges to a single value, which depends on g. In this region, the system has only a single behavior for each value of g. If x is perturbed away from this

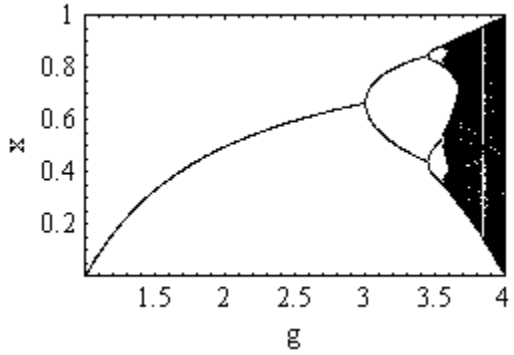


Figure 9: Behavioral Diversity in the Logistic Function

value, it will quickly return. The system has no behavioral diversity.

- For $3 \leq g < 3.569945672\ldots$, x oscillates among a number of discrete alternatives. The Figure clearly shows regions with two, then four alternatives. In fact, as g approaches the upper limit of this range, the number of alternatives doubles repeatedly, so that a value of g can be found to yield any number of alternatives that is an integer power of two.
- For $3.569945672\ldots \leq g$, the system is formally chaotic. In this region, x varies widely over its range, and (left to its own behavior) never repeats its position exactly. This region offers the broadest behavioral potential for x.

The behavioral diversity evident in the chaotic regime is useful only if some way can be found to lock the system down to a particular behavior, but the basic mechanisms for such control have been known for over a decade [36]. The basic idea is to let the chaotic dynamics explore the state space, and when the system reaches a desirable region, to apply a small control force to keep the system there.

It may seem that chaos is a complicated way to generate potential behaviors, and that it would be simpler to use a random number generator. In fact, virtually all such generators *are* in fact nonlinear systems executing in their chaotic regime.

In a multi-agent system, the key to applying this generate-and-test insight is finding a way to exert selective pressure to keep the system balanced at the desired location. Natural systems have inspired two broad classes of algorithm for this purpose: synthetic evolution, and particle swarm optimization.

Synthetic evolution is modeled on biological evolution. Many different algorithms have been developed [31], but they share the idea of a population of potential solutions that varies over time, with fitter solutions persisting and less fit ones being discarded. The variational mechanisms (which usually include random mutation) explore the system's potential behaviors, while the death of less fit solutions and the perpetuation of more fit ones is the control pressure that selects the desired behavior.

Particle swarm optimization [32] is inspired by the flocking behavior of birds. In adaptations of this behavior to computation, solutions do not undergo birth and death (as in evolutionary mechanisms). Instead, they are distributed in some space (which may be the problem space, or an arbitrary structure), and share with their nearest neighbors the best solutions they have found so far. Each solution entity then adjusts its own behavior to take into account a blend of its own experience and that of its neighbors.

Market-based bidding mechanisms may be considered a variation on particle swarm optimization. The similarity lies in selection via behavioral modification through the exchange of information rather than changes in the composition of the population. The approaches differ in the use that is made of the shared information. In the particle swarm, agents imitate one another based on the information they receive, while in bidding schemes, they use this information in more complicated computations to determine their behavior.

3.3.2 Design Principles Derived from Functional Adjustment

Function 1: Generate behavioral diversity.—Structure agents to ensure that their collective behavior will explore the behavioral space as widely as possible. One formula for this objective has three parts.

1. Let each agent support multiple functions.
2. Let each function require multiple agents.
3. Break the symmetry among the agents with random or chaotic mechanisms.

The first two points ensure that system functionality emerges from agent interactions, and that any given functionality can be composed in multiple ways. The third ensures a diversity of outcomes, depending on which agents join together to provide a given function at a particular time.

Function 2: Give agents access to a fitness measure.—Agents need to make local decisions that foster global goals, an insight that is supported by formal analysis in Wolpert's Collective Intelligence (COIN) research [47]. A major challenge is finding measures that can be evaluated by agents on the basis of local information, but that will correlate with overall system state. Determining such measures is a matter for experimentation, although thermodynamic concepts relating short-range interactions to long-term correlations have the potential to yield a theoretical foundation. In one application, we have found the entropy computed over the set of behavioral options open to an agent to be a useful measure of the degree of overall system convergence [12] that agents can use to make intelligent decisions about bidding in resource allocation problems.

Function 3: Provide a mechanism for selecting among alternative behaviors.—If an adequate local fitness metric can be found, it may suffice to guide the behavior of individual agents. Otherwise, agents should compare their behavior with one another, either to vary the *composition* of the overall population (as in synthetic evolution) or to enable individual agents to vary their *behavior* (as in particle swarm optimization).

4 HOW can we apply these principles in engineered systems?

To illustrate the use of these principles, we briefly review several systems, described in more detail in other publications, that produce high-level cognitive behavior from swarming. In each case we review the problem being solved, summarize the behavior of the local elements, and discuss how they reflect the ten design principles outlined in Section 3.

4.1 Pattern Recognition in a Sensor Network [13]

4.1.1 The Problem

Driven by the need for greater efficiency and agility in business and public transactions, more and more data is becoming digitally available in real time on computer networks. These heterogeneous data streams reflect many aspects of the behavior of groups of individuals in a population (e.g., traffic flow, shopping and leisure activities, healthcare needs). A new generation of active surveillance systems that integrate a large number of spatially distributed heterogeneous data streams may be used in various applications, for instance, to protect a civilian

population from bioterrorist attacks, to support real-time traffic coordination systems, to trace collaboration structures in terrorist networks, or to manage public healthcare efficiently.

Active surveillance of population-level activities includes the detection and classification of spatio-temporal patterns across a large number of real-time data streams. Approaches that analyze data in a central computing facility tend to be overwhelmed with the amount of data that needs to be transferred and processed in a timely fashion. Also, centralized processing raises proprietary and privacy concerns that may make many data sources inaccessible. Our architecture avoids these problems through decentralization. Instead of transferring the data to a centralized processing facility, we transfer the processes (fine-grained agents) to the data sources. This architecture addresses both of these concerns. Access restrictions may be guaranteed through proven local processes. Bandwidth is reduced because long-distance communication of data is needed only when the network detects a pattern and needs to invoke a higher authority for action. (Ultimately, one would like the response itself to be a distributed emergent response, but political realities suggest that in the immediate future self-organizing recognition systems will be much more acceptable than self-organizing systems that take action on people and property.)

4.1.2 Summary of Architecture

We consider a distributed swarming agent architecture the most appropriate answer to the challenge of detecting spatio-temporal patterns in a network of heterogeneous sources of potentially proprietary real-time data. Instead of attempting to stream a tremendous amount of data into a central processing facility, we integrate the external sources into a network for mobile agent computing. Essentially, this network of agent processing nodes is a massively parallel computer for pattern detection and classification with a unique way of self-organizing the processing tasks.

Into our network of processing nodes we deploy large populations of simple mobile agents that coordinate their activities using stigmergy. Each node generates agents at a constant rate, and agents die after a fixed lifetime, thus ensuring coverage of the entire area under surveillance. Using artificial pheromones, the agents dynamically organize themselves around patterns observed in the data streams. The emergence of globally coordinated behavior through stigmergic interactions among many fine-grained software agents in a shared computational environment is facilitated by a component of the distributed runtime environment that emulates actual pheromone dynamics (aggregation, evaporation, dispersion) in the physical world. Our heterogeneous agent system continuously executes two parallel processes: pattern detection and pattern classification. More populations of agents could be deployed at any time, for instance to introduce additional criteria in the detection process, or to add more classification schemes.

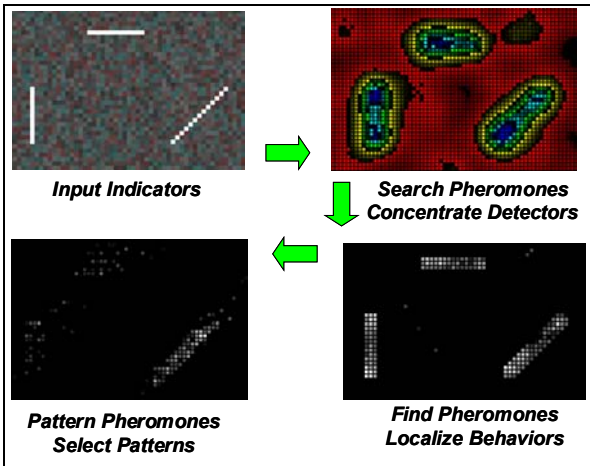


Figure 10: Stigmergic Pattern Detection and Classification

The agents executing the detection process

("Detectors") continuously process the input data and search for spatio-temporal structures, using two sets of flavors of pheromones. Detectors use Search pheromones to mark suspicious areas of the network and attract other detectors to confirm their discovery. A second set of Find pheromones, which require more deposits to stabilize, is used to record this confirmation, informing a local node that it is likely to be an instance of the pattern in question and enabling it to take appropriate action. Detectors search for unusually high differences in the data streams of neighboring locations in the network.

"Classifier" agents are responsible for the classification of the detected patterns according to a particular classification scheme. The pattern classification scheme used in our demonstration correlates the detected patterns with a particular, dynamically changing geographic direction (wind, modeling the dispersion of a bioterrorist weapon). The Classifiers move in a way that models the pattern being sought, and deposit a Pattern pheromone when they encounter a pattern that matches their behavior.

Figure 10 shows the performance of the algorithm. The upper-left display is a grid in which each cell is set either to a random mixture of Red-Green-Blue, or to white. Viewing the overall display, we can see that the white cells are different from the mass of the other cells, and that they are arranged in extended patterns. However, a single cell with only local knowledge of its neighborhood can know neither of these facts. The upper-right display shows the Search pheromones deposited by Detectors searching for unusual cells, based on their recent experience. The high propagation of these pheromones creates gradients that attract other Detectors for confirmation. As more and more Detectors agree that the cells are indeed unusual, Find pheromone (lower right) accumulates to mark the location of the unusual cells. Finally, "Classifier" agents moving diagonally across the field sense repeated Find pheromones aligned with their movement and mark them with Pattern pheromone to indicate an instance of a particular structure of interest.

4.1.3 Principles

Coupling 1: Use a distributed environment.—The network of data collection nodes is distributed over space. For spatio-temporal pattern recognition, each data collection node maintains a temporal data structure to distribute agent interactions in time as well.

Coupling 2: Use an active environment.—The environment implements the basic pheromone dynamics of Aggregation (fusion of observations from multiple agents), Propagation (communication), and Evaporation (truth maintenance).

Coupling 3: Keep agents small.—Both data nodes and mobile agents are small compared with the overall system, and all interactions are local. No single agent can solve the problem. No data node can know on its own that it is part of a pattern being sought, nor can any individual Detector or Classifier confirm the detection or classification without collaboration by its colleagues.

Coupling 4: Map agents to Entities, not Functions.—The data nodes correspond to distributed data sources in the physical domain. The Detectors and Classifiers are not domain entities, but neither does any one of them implement a function by itself. Detection and Classification emerge from the interactions of multiple Detectors and Classifiers. It is perhaps best to think of the Detectors and Classifiers as instances of hypotheses about structures in the environment, hypotheses that are confirmed or discredited through the stigmergic interactions.

Autocatalysis 1: Think Flows rather than Transitions.—The fundamental information flow in this application is the pheromone loop illustrated in Figure 8.

Autocatalysis 2: Boost and Bound.—Search pheromone builds up through positive feedback: the more is deposited, the more Detectors come to that area and the more they deposit Search pheromone. If unrestrained, this reinforcement could lead to all Detectors becoming concentrated in one area, leaving other regions unexplored. Bounds on system dynamics are provided by the programmed death of agents and their continual rebirth at nodes distributed throughout the area.

Autocatalysis 3: Diversify agents to keep flows going.—This architecture has three main species of agents among which information flows: data nodes, Detectors, and Classifiers.

Function 1: Generate behavioral diversity.—Each function (detection and classification requires multiple agents. It is less clear in this case that each agent performs multiple functions. However, agents do differ from one another.

- The birth location of each Detector or Classifier varies across the search area.
- A key behavioral parameter of mobile agents in this application is a threshold that indicates how distinct a data node must be from others that the agent has seen recently before it will deposit a pheromone. This threshold is randomly generated.
- Agents' movements, while influenced by local pheromone gradients, always incorporate a stochastic component. The pheromone strength in nearby nodes is used to weight a roulette wheel that determines the probability that the agent will move to each of those nodes in the next step.

Function 2: Give agents access to a fitness measure.—The pheromone fields accumulate information about outlying nodes and extended patterns that combine the observations of many mobile agents that have followed different individual trajectories. Thus they are locally accessible repositories of information gathered over a much broader area, providing a local view of the global state of the problem.

Function 3: Provide a mechanism for selecting among alternative behaviors.—Mobile agents adjust their detection thresholds using a variation of particle swarm optimization.

4.2 Searching and Imaging with Unmanned Air Vehicles [40]

4.2.1 The Problem

Some sensing problems (e.g., three-dimensional imaging with synthetic aperture radar) requires the coordination of multiple sensing platforms. Consider a swarm of unpiloted air vehicles (UAV's) whose task is to locate and image potential targets hiding under dense foliage. The swarm must achieve three objectives that require different behaviors on the part of individual UAV's.

In **searching**, each UAV must effectively cover a large search space and revisit locations regularly, maximizing detection probability based on known characteristics of the target (e.g., visibility angle), while not exhibiting any obvious systematic search patterns that would permit mobile targets to execute simple avoidance strategies. A single sensor can generate enough information to suggest the presence of a target, though it cannot image the target by itself.

When a vehicle detects a target, it announces the location of the target, and vehicles that receive this announcement begin a coordinated **imaging** task**.** In this phase, each vehicle must collect data from varying angles along linear trajectories (box) while minimizing both the effort (the number of required vehicles and the distance they must move) and the data collection time (by collecting data in parallel).

In addition, individual vehicles require periodic refueling or other **maintenance**, and the swarm must ensure that individual vehicle requirements are met without compromising the ability of the overall swarm to continue functioning.

Figure 11: Visitation Pheromone Map of one UAV in the Swarm

4.2.2 Summary of Architecture

Our stigmergic approach to this problem uses digital pheromones. An important contrast with the pheromone mechanisms in our other two example applications is that while those applications envisioned a network of physical nodes maintaining the pheromone field *externally* to the agents, in this case each agent maintains an *internal* pheromone map that tiles the search space into discrete cells. Each cell is a place in a pheromone infrastructure, which means that the agent that controls the vehicle may deposit and sense digital pheromones of different flavors in that cell. In principle, agents could propagate these maps to one another through local interaction, thus achieving a stigmergic analog to the DAI technique of partial global planning [19], but even without generating such a "global distributed view," the local iconic representation has significant benefits over more conventional robotic techniques such as occupancy maps.

During **search,** when a vehicle passes through the area in the search space that is assigned to a particular cell, it deposits a unit of the visitation pheromone into that cell in its internal map. In addition, the agent broadcasts its location, and the agents of any other vehicle within communications reach then deposit a visitation pheromone into their maps too. Thus, the agents mark cells that some member of the swarm has already visited. Figure 11 shows a snapshot of the visitation pheromone map of one agent in the swarm.

Local concentrations of pheromones lose strength over time, which enables the swarm to "forget" visitations to locations that occurred a long time ago. This knowledge management process ensures that the search process keeps revisiting locations in case targets have moved in.

The individual agent decides its vehicle's trajectory based on its internal map of visitation pheromones. Once it has reached its previous goal, the agent probabilistically selects a new location. The probability of the selection of a particular location is inversely proportional to its distance to the vehicle's current location and to the strength of the visitation pheromone concentration in the cell that covers this location. Thus the agents tend to prefer nearby locations that have not been visited recently, and collectively explore the whole search space.

An agent that detects a potential target dynamically forms an **imaging** team. Team formation is a collaborative process in which agents bid for a role in the team depending on the match of the vehicles' imaging capabilities with the role's requirements (hard constraint) as well as the current distance of the eligible vehicles to the detected target (soft constraint).

Once roles are assigned, the team members plot the optimal trajectories for their respective data acquisition flight and execute the imaging task. Depending on the imaging modality (coherent vs. non-coherent), the data acquisition may be executed individualistically or synchronized across the team. Once the task is completed, the team disbands and the agents resume their search behavior.

A team-based approach to **maintenance** can accommodate UAV's with different fuel consumption rates, as well as variations in the availability of maintenance resources at the base. UAV's deposit a pheromone flavor that communicates the intensity of their current desire for maintenance, while the base propagates a pheromone indicating its current level of load. A UAV's decision to shift into the maintenance role is promoted by its own desire for refuel and inhibited by the level of refueling pheromone it senses from neighboring UAV's and the load pheromone propagated from the base.

Figure 12 shows a screen shot of this system. Most of the UAV's are scanning the area in search mode, but four have formed a verification team to image a suspected target, while one is one its way back to the refuel station.

4.2.3 Principles

Coupling 1: Use a distributed environment.—The pheromone environment maintained by each UAV is not distributed, but locality of interaction among UAV's is enforced by their geographical dispersion over the search area.

Coupling 2: Use an active environment.—The pheromone environment implements the usual pheromone dynamics of aggregation, propagation, and evaporation. In addition, each UAV's physical environment includes the other UAV's, whose behaviors change based on their individual experiences.

Coupling 3: Keep agents small.—No single UAV can do the entire task. At least four are needed to image a target, and even more are required to maintain a high level of search.

Coupling 4: Map agents to Entities, not Functions.—The agents in this system correspond to physical UAV's.

Autocatalysis 1: Think Flows rather than Transitions.—The main information flows in this system are the pheromone flows of Figure 8, and the communications flows between a UAV that has detected a target and the other UAV's whom it seeks to recruit to perform imaging.

Autocatalysis 2: Boost and Bound.—A UAV's attraction to an imaging team is based on positive feedback, while the visitation map approach to dispersing the UAV's is an example of negative feedback.

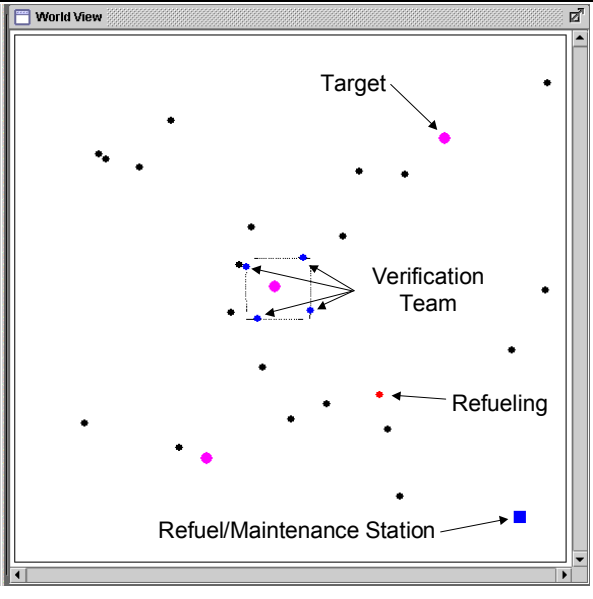


Figure 12: Stigmergic Role Differentiation

Autocatalysis 3: Diversify agents to keep flows going.—UAV's are diverse in their location. In addition, each UAV decision (the angle at which to traverse the search area in search mode, whether to join an imaging team, whether to return for refueling) is stochastically weighted.

Function 1: Generate behavioral diversity.—Each function requires multiple agents, and each agent supports all three functions. Symmetry among agents is broken by making decisions with weighted stochastic functions.

Function 2: Give agents access to a fitness measure.—Several fitness functions influence agent behavior. A UAV's proximity to another that has sensed a target, and its sensory configuration, influence whether it joins an imaging team. Its current fuel level and the load level of the base influence whether it enters maintenance mode.

Function 3: Provide a mechanism for selecting among alternative behaviors.—Agents decide whether to image based on a collective sharing of information in a bidding process. Agents decide whether to enter maintenance mode based on their own fuel level and the load at the base.

4.3 Dynamic Target Selection and Path Planning [41, 42]

4.3.1 The Problem

The current generation of UAV's reduces the threat to human operators, but leaves several problems unresolved.

- It does not decrease the manpower requirements. Each aircraft requires a flight crew of one to three people, so deploying large numbers of UAV's requires committing and coordinating many human warfighters.
- The high-bandwidth needed for linking the flight crew to the aircraft places severe constraints on available communications resources.
- Fusion of information from multiple sources (satellite imagery, sensors on UAV's, unattended ground sensors, information from special forces in the field) is a continuing challenge.

We want a UAV to be able to manage the details of its own mission, avoiding dynamic threats as soon as they arise and planning its path to optimize its movement through the battlespace.

4.3.2 Summary of Architecture

Like our other two examples, this application uses digital pheromones. These pheromones live in a network of *place agents,* which represent regions of the battlespace. All place agents can run on a single computer for simulation purposes, but in actual deployment each place agent might run on an enhanced unattended ground sensor (UGS) placed in the battlespace by air drops or artillery and responsible for any location to which it is closer than any other UGS. We refer to such an enhanced UGS as a HOST (Hostility Observation and Sensing Terminal). Each place agent is a neighbor to a limited set of other place agents, those that are responsible for adjacent regions of space, and it exchanges local information with them. In addition to place agents, the system includes *walker agents,* representing physical resources such as UAV's. Walker agents move through the battlespace by interacting with the place agent for each region that they visit.

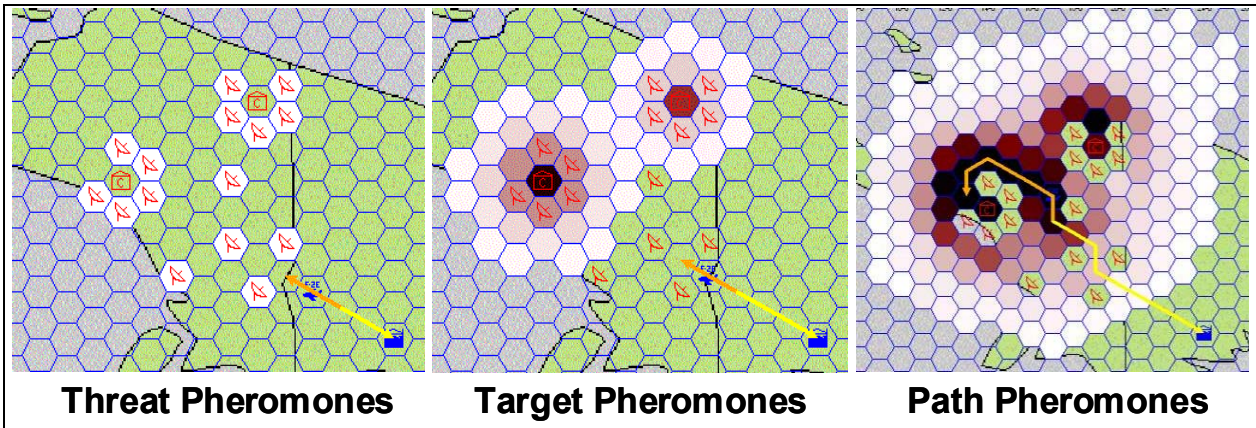


Figure 13: Pheromone Flavors in Emergent Path Planning

Place agents and walker agents are software entities, while HOST's and UAV's are the hardware in which they run.

Each place agent maintains a scalar variable corresponding to each pheromone flavor. It augments this variable when it receives additional pheromones of the same flavor (whether by deposit from a walker agent, from its own sensors, or by propagation from a neighboring place agent). It also evaporates the variable over time, and propagates pheromones of the same flavor to neighboring place agents based on the current strength of the pheromone. Different flavors may indicate the presence of a *threat* that should be avoided in the place's region or the presence of a *target* that should attract UAV's.

The development of a path by a natural ant colony depends on the stochastic interaction of many ants, some of whom wander off and die. Current UAV's such as the Predator and the Global Hawk are far too expensive to use in a stochastic search mode. Instead, each UAV's walker agent periodically emits *ghost agents*, software agents without a corresponding hardware resource. These ghost agents are attracted by target pheromone and repelled by threat pheromones, and lay down a path pheromone to store the results of their explorations. Reinforcement of this path pheromone by multiple ghosts leads to the emergence of a path that the UAV then follows. (Recent advances in inexpensive micro-UAV's opens up the potential for having the UAV's themselves swarm, as in the example discussed in Section 4.2.)

Figure 13 illustrates the functions of the different pheromones in this process. On the left, intelligence about threats is translated into threat pheromones that propagate only a short distance, since their purpose is not to attract distant ghosts, but to prevent nearby ones from wandering into danger. In the center, intelligence about targets results in target pheromones that propagate widely, attracting ghost agents. The higher-priority target (to the west) emits pheromone at a higher rate, thus generating a broader field. The right-hand display shows the path pheromones deposited by the ghost. A UAV following the ridge of this field will be attracted to the appropriate target, while avoiding intervening threats.

4.3.3 Principles

Coupling 1: Use a distributed environment.—The network of HOST's provides an environment that is physically distributed throughout the battlespace.

Coupling 2: Use an active environment.—The HOST's implement the pheromone dynamics of aggregation, propagation, and evaporation.

Coupling 3: Keep agents small.—Intelligence about the battlespace is not concentrated in a single machine, but maintained across many HOST's, each responsible for a small region. The path planning is done by ghost agents, which are small compared with the UAV's walkers. (In our experiments, each walker has about 300 concurrent ghosts.)

Coupling 4: Map agents to Entities, not Functions.—Agents correspond to physical regions and resources.

Autocatalysis 1: Think Flows rather than Transitions.—The basic flow is the pheromone cycle of Figure 8.

Autocatalysis 2: Boost and Bound.—Path emergence among the ghosts is the result of positive feedback as they respond to path pheromones already in place, combined with the bounding influence of pheromone evaporation over time. The ghost population is maintained by continuous birth and programmed death.

Autocatalysis 3: Diversify agents to keep flows going.—The system uses three main species of agents: place agents, walker agents, and ghost agents. In addition, different resources have different walkers, different regions have different place agents, and ghosts diversify themselves through stochastic movement. Walkers and ghosts deposit and sense pheromones in the place agents, and thus pass information among themselves.

Function 1: Generate behavioral diversity.—The system's main function is path planning, in which all agents participate without any of them dominating. An important class of diversity among ghost agents is the equation by which they translate pheromone levels that they sense in their immediate environment into movement decisions. Originally, we hand-tuned the parameters of this equation. We found improved performance when we allowed the parameters to vary around the hand-tuned mean, and even more improvement when we evolved the parameters [44].

Function 2: Give agents access to a fitness measure.—The speed with which a ghost reaches a potential target and returns home is a good measure of the fitness of its search parameters, so we use the lifetime remaining to a ghost as its fitness measure.

Function 3: Provide a mechanism for selecting among alternative behaviors.—We use a variety of the genetic algorithm for adjusting the distribution of search parameters in the ghost population. An important characteristic of our application is that this adaptation happens as the system operates, not in an off-line planning process.

5 Conclusion and Prospect

Swarming systems have demonstrated their effectiveness as an alternative model of cognition. This experience is leading to a growing body of engineering knowledge for the deployment of such systems. They are best suited for resource-constrained systems of discrete interacting elements that exhibit distribution, decentralization, and dynamic change. The self-organization that gives these systems their power requires not only interaction among the agents, but the potential for autocatalytic loops, and some mechanism (such as synthetic evolution or particle swarm optimization) for selecting appropriate behaviors from a wider repertoire based on some fitness function. We have deployed these mechanisms successfully in a number of applications,

including distributed pattern recognition, team formation and management, dynamic target selection and path formation, resource allocation, document search and retrieval, and ecosystem management.

This engineering perspective on swarming systems recognizes that for some applications or problems, conventional cognitive techniques may be more appropriate. Now that we understand where swarming systems are appropriate and some of the principles that enable them, the next challenge is integrating them with more conventional cognitive systems. We are pursuing several lines of research in support of hybrid agent systems, including

- using swarming systems as internal "brains" for more conventional cognitive systems;
- integrating fine-grained and coarse-grained agents as peers in a single system, with fine-grained agents providing ease of implementation and reduced need for knowledge engineering, while coarse-grained agents provide a clearer cognitive interface to human stakeholders;
- developing mathematical methods for imputing cognitive behavior to non-cognitive agents in support of integration with cognitive agents;
- developing a design and specification methodology at a sufficiently abstract level that it can be applied to either class of agent.

6 Acknowledgments

This work is supported in part by DARPA under the JFACC program and WASP seedling. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

7 References

[1]H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous Computing. *Communications of the ACM*, 43(5):74-82, 2000. URL citeseer.nj.nec.com/abelson95amorphous.html.

[2]J. Arquilla and D. Ronfeldt. Swarming and the Future of Conflict. DB-311, RAND, Santa Monica, CA, 2000. URL http://www.rand.org/publications/DB/DB311.

[3]P. Ball. *The Self-Made Tapestry: Pattern Formation in Nature*. Princeton, NJ, Princeton University Press, 1996.

[4]G. Beni. The Concept of Cellular Robotic System. In *Proceedings of IEEE Int. Symp. on Intelligent Control*, Los Alamitos, CA, pages 57-62, IEEE Computer Society Press, 1988, 1988.

[5]G. Beni and S. Hackwood. Stationary Waves in Cyclic Swarms. In *Proceedings of IEEE Int. Symp. on Intelligent Control*, Los Alamitos, CA, pages 234-242, IEEE Computer Society Press, 1992, 1992.

[6]G. Beni and J. Wang. Swarm Intelligence. In *Proceedings of Seventh Annual Meeting of the Robotics Society of Japan*, Tokyo, pages 425-428, RSJ Press, 1989, 1989.

[7]G. Beni and J. Wang. Theoretical Problems for the Realization of Distributed Robotic Systems. In *Proceedings of IEEE International Conference on Robotic and Automation*, Los Alamitos, CA, pages 1914-1920, IEEE Computer Society Press, 1991, 1991.

[8]C. K. Biebricher, G. Nicolis, and P. Schuster. Self-Organization in the Physico-Chemical and Life Sciences. 16546, European Union, 1995.

[9]E. Bonabeau. Swarm Intelligence. In *Proceedings of Swarming: Network Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. New York, Oxford University Press, 1999.

[11] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Dr.rer.nat. Thesis at Humboldt University Berlin, Department of Computer Science, 2000. URL http://dochost.rz.hu-berlin.de/dissertationen/brueckner-sven-2000-06-21/PDF/Brueckner.pdf.

[12] S. Brueckner and H. V. D. Parunak. Information-Driven Phase Changes in Multi-Agent Coordination. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, Melbourne, Australia, pages 950-951, 2003. URL http://www.erim.org/~vparunak/AAMAS03InfoPhaseChange.pdf.

[13] S. A. Brueckner and H. V. D. Parunak. Swarming Agents for Distributed Pattern Detection and Classification. In *Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002*, Bologna, Italy, 2002. URL http://www.erim.org/~vparunak/PatternDetection01.pdf.

[14] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton, NJ, Princeton University Press, 2001.

[15] C. Castelfranchi. Founding Agent's 'Autonomy' on Dependence Theory. In *Proceedings of 14th European Conference on Artificial Intelligence*, Berlin, Germany, pages 353-357, IOS Press, 2000.

[16] B. Clough. Emergent Behavior (Swarming): Tool Kit for Building UAV Autonomy. In *Proceedings of Swarming: Network Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003.

[17] J. P. Crutchfield. The Calculi of Emergence: Computation, Dynamics, and Induction. *Physica D*, 75:11-54, 1994. URL ftp://ftp.santafe.edu/pub/CompMech/papers/CalcEmerg1_46.ps.Z.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):1-13, 1996.

[19] E. H. Durfee and V. R. Lesser. Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167-1183, 1991. URL citeseer.nj.nec.com/durfee91partial.html.

[20] S. J. A. Edwards. Swarming on the Battlefield: past, Present, and Future. MR-1100-OSD, RAND, Santa Monica, CA, 2000.

[21] S. J. A. Edwards. Military History of Swarming. In *Proceedings of Swarming: Network Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003.

[22] M. Fenster, S. Kraus, and J. S. Rosenschein. Coordination without Communication: Experimental Validation of Focal Point Techniques. In *Proceedings of International Conference on Multi-Agent Systems (ICMAS'95)*, San Francisco, CA, pages 102-108, AAAI, 1995.

[23] B. R. Gaines. Mediator Research Program. 1995. Web page, http://ksi.cpsc.ucalgary.ca/projects/Mediator/.

[24] M. R. Genesereth, M. Ginsburg, and J. S. Rosenschein. Cooperation without Communication. In *Proceedings of National Conference on Artificial Intelligence (AAAI'86)*, pages 51-57, AAAI, 1986.

[25] P.-P. Grassé. La Reconstruction du nid et les Coordinations Inter-Individuelles chez *Bellicositermes Natalensis et Cubitermes sp.* La théorie de la Stigmergie: Essai d'interprétation du Comportement des Termites Constructeurs. *Insectes Sociaux*, 6:41-84, 1959.

[26] S. Hackwood and G. Beni. Self-Organizing Sensors by Deterministic Annealing. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robot and Systems*, Los Alamitos, CA, pages 1177-1183, IEEE Computer Society Press, 1991, 1991.

[27] S. Hackwood and G. Beni. Self-organization of Sensors for Swarm Intelligence. In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pages 819-29, 1992.

[28] M. Heusse, S. Guérin, D. Snyers, and P. Kuntz. Adaptive Agent-Driven Routing and Load Balancing in Communication Networks. *Advances in Complex Systems*, 1:234-257, 1998.

[29] E. Hornung and B. M. Bryan, Editors. *The Quest for Immortality: Treasures of Ancient Egypt*. Washington, DC, National Gallery of Art, 2002.

[30] D. Inbody. Swarming: Historical Observations and Conclusions. In *Proceedings of Swarming: Network Enabled C4ISR*, Tysons Corner, VA, ASD C3I, 2003.

[31] C. Jacob. *Illustrating Evolutionary Computation With Mathematica*. San Francisco, Morgan Kaufmann, 2001.

[32] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. San Francisco, Morgan Kaufmann, 2001.

[33] L. Leonardi, M. Mamei, and F. Zambonelli. Co-Fields: Towards a Unifying Model for Swarm Intelligence. DISMI-UNIMO-3-2002, University of Modena and Reggio Emilia, Modena, Italy, 2002. URL http://polaris.ing.unimo.it/didattica/curriculum/marco/Web-Co-Fields/stuff/Swarm.pdf.

[34] MIT. Amorphous Computing Home Page. 2003. Web Site, http://www.swiss.ai.mit.edu/projects/amorphous/.

[35] E. Neufeld. Insects as Warfare Agents in the Ancient Near East. *Orientalia*, 49(1):30-57, 1980.

[36] E. Ott, C. Grebogi, and J. A. Yorke. Controlling Chaos. *Physical Review Letters*, 64(11):1196-1199, 1990.

[37] H. V. D. Parunak. Distributed AI and Manufacturing Control: Some Issues and Insights. In Y. Demazeau and J.-P. Müller, Editors, *Decentralized AI*, pages 81-104. North-Holland, 1990.

[38] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997. URL http://www.erim.org/~vparunak/gotoant.pdf.

[39] H. V. D. Parunak and S. Brueckner. Entropy and Self-Organization in Multi-Agent Systems. In *Proceedings of The Fifth International Conference on Autonomous Agents (Agents 2001)*, Montreal, Canada, pages 124-130, ACM, 2001. URL www.erim.org/~vparunak/agents01ent.pdf.

[40] H. V. D. Parunak and S. Brueckner. Swarming Coordination of Multiple UAV's for Collaborative Sensing. In *Proceedings of Second AIAA "Unmanned Unlimited" Systems, Technologies, and Operations Conference*, San Diego, CA, AIAA, 2003. URL http://www.erim.org/~vparunak/AIAA03.pdf.

[41] H. V. D. Parunak, S. A. Brueckner, and J. Sauter. Digital Pheromone Mechanisms for Coordination of Unmanned Vehicles. In *Proceedings of First International Conference on*

Autonomous Agents and Multi-Agent Systems (AAMAS 2002), Bologna, Italy, pages 449-450, 2002. URL www.erim.org/~vparunak/AAMAS02ADAPTIV.pdf.

[42] H. V. D. Parunak, M. Purcell, and R. O'Connell. Digital Pheromones for Autonomous Coordination of Swarming UAV's. In *Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems,Technologies, and Operations Conference*, Norfolk, VA, AIAA, 2002. URL www.erim.org/~vparunak/AIAA02.pdf.

[43] K. Pister. Smart Dust: Autonomous sensing and communication in a cubic millimeter. 2001. Web Page, http://robotics.eecs.berkeley.edu/~pister/SmartDust/.

[44] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. Brueckner. Evolving Adaptive Pheromone Path Planning Mechanisms. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS02)*, Bologna, Italy, pages 434-440, 2002. URL www.erim.org/~vparunak/AAMAS02Evolution.pdf.

[45] F. Schweitzer and J. Zimmermann. Communication and Self-Organization in Complex Systems: A Basic Approach. In M. M. Fischer and J. Fröhlich, Editors, *Knowledge, Complexity and Innovation Systems*, pages 275-296. Springer, Berlin, Germany, 2001. URL http://summa.physik.hu-berlin.de/~frank/p_webwien.html.

[46] C. R. Shalizi. *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. Ph.D. Thesis at University of Wisconsin, Department of Physics, 2001.

[47] D. Wolpert and K. Tumer. Collective Intelligence. 2002. Web site, http://ic.arc.nasa.gov/projects/COIN/index.html.

[48] M. J. Wooldridge and N. R. Jennings. Pitfalls of Agent-Oriented Development. In *Proceedings of 2nd Int. Conf. on Autonomous Agents (Agents-98)*, Minneapolis, MN, pages 385-391, 1998. URL http://citeseer.nj.nec.com/wooldridge98pitfalls.html.