Organization, Self-Organization, Autonomy and Emergence: Status and Challenges

Sven Brueckner [1] Hans Czap [2]

[1] New Vectors LLC
3520 Green Court, Suite 250, Ann Arbor, MI 48105-1579, USA
Email: sven.brueckner@newvectors.net
http://www.altarum.net/~sbrueckner
[2] University of Trier, FB IV Business Information Systems I
D-54286 Trier, Germany
Email: Hans.Czap@uni-trier.de
http://www.wi.uni-trier.de/

Abstract: Development of IT-systems in application domains is facing an ever-growing complexity resulting from a continuous increase in dynamics of processes, applications- and run-time environments and scaling. The impact of this trend is amplified by the lack of central control structures. As a consequence, controlling this complexity and dynamics is one of the most challenging requirements of today's system engineers.

The lack of a central control instance immediately raises the need for software systems which can react autonomously to changing environmental requirements and conditions. Therefore, a new paradigm is necessary how to build software systems changing radically the way one is used to think about systems and its development: self-organization, autonomy and emergence are the concepts that have to be build into future systems.

This paper shows first steps in order to arrive at a theory of self-organization, autonomy and emergence and provides some of the fundamental principles that should be followed for the design of truly adaptive software systems.

Keywords: systems theory, organization, self-organization, emergence, autonomy.

1. Introduction

This section shows the basic principles of self-organization in natural and artificial systems. The components identified will be explained in the following chapters. The aim is to show, how organizations are able to change their inherent structures respectively how to apply structural learning as a self-organizing process.

The ever-growing complexity of today's IT-supported processes in business, government, military or entertainment, seriously challenges traditional approaches to IT-application design, implementation, deployment, and management. Today's software systems are increasingly decentralized, distributed, and dynamic, just as the problem domains in which they are operating are taking on these characteristics too.

To deal with this growing software and domain complexity and the increasing dynamics of application- and run-time-environments, software systems must be able to react and adapt autonomously to changing requirements. Therefore, approaches that rely on the fundamental principles of self-organization and autonomy are growing in acceptance.

Following such approaches, software system functionality is no longer explicitly designed into its component processes but emerges from lower-level interactions that are purposefully unaware of the system-wide behavior. Furthermore, organization, the meaningful progression of local sensing, processing and action, is achieved by the system components themselves at runtime and in response to the current state of the environment and the problem that is to be solved, rather than being enforced from the "outside" through design or external control.

While such self-organizing and emergent software systems are very attractive for their inherent flexibility, robustness and graceful degradation under component failure – to name just a few commonly cited attributes – the design, implementation, validation and evaluation of such systems remain major challenges for researchers and system developers. Typical questions and issues revolving around such approaches are:

- What principles govern the self-organization of autonomous components and the emergence of desired system-level features?
- How to break down the system into interacting components?
- How to design inter-component interactions and component-internal reasoning to achieve the desired system-level functions?
- How to define a static and dynamic environment that supports and nourishes the goal-driven operation of the system components?
- How to design methods for human operators to interact and guide such systems?
- How to define interfaces for external IT systems to interact with such a system for mutual benefit?
- How to implement robust decentralized and distributed systems?
- How to understand their behavior under a wide range of operational parameters?
- How to formally or even empirically validate and evaluate the emergence of the desired system-level

functions?

- How to deploy, maintain, and update the components of such a complex system?
- How to certify correct functioning also in malicious environments

In the following, we discuss various such issues in more detail to provide context to the various contributions of this special journal edition.

2. Theory of Organization, Self-Organization, Autonomy and Emergence

One of the major challenges of the new approach to engineering complex software system by harnessing principles such as self-organization or emergence is the lack of a formal theory that can underpin such systems, whether engineered or "naturally" grown. The interdisciplinary science of Complex Adaptive Systems (CAS) offers some useful concepts and ideas that are drawn from a wide variety of fields, such as physics, chemistry, biology, economics, or social science. But, in general, our ability to formally model, analyze and evaluate such systems in their structure or dynamics is still very limited. Before presenting a formal definition, a more phenomenal understanding of the concept of an organization and its properties is essential.

2.1 Organizations and Organizational Dimensions

The concept 'organization' has different meanings. Mainly, four versions are distinguished:

(i) An organization A is a set of elements, $A = \{a_1, .., a_n\}$. This definition is in correspondence with the formal definition of a system, but is very unspecific. As it is with universal definitions, they do not help much in differentiating an organization from any other system. Therefore, in the following this version is disregarded.

(ii) An organization can be understood as the process of organizing.

(iii) An organization can be seen as the instrument or mean to achieve specific goals. In this context one talks of the instrumental character of an organization. The concepts (ii) and (iii) are essential when talking about self-organization, since, as is later shown, self-organization is understood as the process of adapting an organization by internal forces to a specific requirement or goal.

(iv) Finally, an organization can be identified with the result of organizing, i.e. an organization is seen as a social institution.

Organizations and Distributed Processing have many common characteristics. The understanding of the fundamental properties of organizations helps in understanding basic principles of distributed processing. For example, the fundamental principles of work-sharing and specialization of different processing entities immediately leads to questions of coordination and control as well as efficiency. Goal-oriented behavior, performance, scaling etc. are of uppermost relevance. Questions of architectural design of any distributed processing system and the question whether a design fits the intended purpose, immediately leads to definition (iii) above, whereas questions of coordination at run-time strongly relate to definition (ii). Our focus, therefore, lies on the concepts (ii) and (iii) of an organization.

In management-theory organizational settings are characterized more specific by the five 'structural dimensions' respectively 'structural variables' [9]. These dimensions immediately relate to architectural design-problems during conceptual phase of systems development:

Specialization. Depending on the kind of problems to be solved by the system, specific capabilities of processing units must be assigned. In general, having specific purpose processing units will result in high performance but low reliability. If a high degree of reliability and flexibility is intended the components of the system should show overlapping capabilities.

Coordination. Different coordination measures relevant for the systems design consist of 'coordination by algorithms', 'coordination by direct communication' and 'self-coordination'. 'Coordination by algorithms' is very efficient in static environments, where the problems to be solved are well-known in advance. In distributed processing one generally prefers 'direct communication' or 'self-coordination'.

In the case of 'direct communication' the client, who requests a problem solution, must know the specific capabilities of the servers in question. Thus, it's a problem of information availability at run-time. The well known contract-net-protocol is an example of 'coordination by direct communication'.

In the case of 'self-coordination' a group of processing units, respectively a subsystem is confronted with the problem to be solved. The processing units decide by their own, which part of the problem will be solved by which unit. The blackboard –paradigm, where a task is announced to the public and processing units declare readiness to solve the problem, is one of the most popular forms of self-coordination. As is known of blackboard-architecture communication likely becomes a bottleneck and limits scalability. This is generally true for self-coordination approaches, if coordination uses a central device. As is shown in the following, stigmergy provides means for self-coordination by changing environmental variables, thus avoiding bottlenecks of hierarchical or other centralized approaches.

Configuration. Configuration relates to the organizational structure, i.e. the division of labor according to the tasks, responsibilities, obligations and accountability of members of the organization. Design decisions address the number of hierarchy levels, the span of control, definition of roles etc.

The dimensional variables coordination and configuration are not independent each other. Typically, by the definition of roles and correspondent responsibilities the need for coordination can be drastically reduced.

Decision distribution. Decision distribution is correlated to the delegation of decisions and the question of responsibility. Self-organization of a subsystem presupposes the possibility of the members of this subsystem to decide on its own. For self-organizing groups the group-manager does not decide about the tasks each member of the group has to do. Rather, the group will assign by self-coordination the tasks to its members. Self-organization and autonomy of members within a group correspond to each other.

Formalization. Formalization relates to the degree at which general rules and formal behavior are implemented within an organization. A high degree of formalization makes sense mainly in static environments. In the case of complex dynamic environments a high degree of flexibility is necessary. Insofar is a high degree of formalization prohibitive for self-organization.

2.2 Self-Organization and Stigmergy

The phenomenon of purposeful self-organization is not an inherent property of systems that just happens by itself. Rather, specific requirements must be fulfilled. In social organizations some of these requirements have been mentioned above, i.e. reduced formalization, decision delegation, self-coordination. First, mainly self-organization will happen, if the team as a group of peers is responsible for the work to be done and not the individual within the team. Second, if there is some incentive for the individual to have success. Third, if each individual knows about the importance for the work to be done or in other words about the incentive in the case of success and punishment in the case of failure.

Since behavior of human beings often does not obey strict laws, it's easier to study primitive insects and their self-organizing behavior in order to gain the necessary knowledge about how to build artificial systems that show self-organization.

[2] in studying self-organization in swarms of social insects found the following ingredients that are relevant for self-organization:

- **Positive and negative feedback**. Positive feedback will be needed to recruit other insects and reinforce specific behavior. Negative feedback counterbalances the positive one and helps to stabilize the collective pattern, for example the exhaustion of some food-source.
- **Amplification of fluctuations**. For insects, there must be some random behavior in order to discover new solutions or to adapt to changing environments. In building artificial systems there must be some means to insure that no uniform behavior of all exemplars will happen. Autonomy on the level of behavior of individuals is one way to achieve this purpose. An other way would be to let behavior depend on some probability distribution, as is frequently done in ant-algorithms.
- **Multiple interactions**. For self-organization it is essential that individuals are able to make use of the results of their own activities as well as of other's activities. Essentially, this means that there is some form of a communication system. This can be sub-symbolic and quantitative as is the case for the pheromone markers of ants or symbolic and qualitative as in negotiations among humans. In either case, it allows collective learning and synchronized behavior.

Engineered Multi-Agent Systems (MAS) are created for a specific purpose. Thus, in general, there is typically some goal determining the behavior of agents that may directly or indirectly providing positive or negative feedback. Clearly, in MAS one has a communication system that allows for collective learning. But autonomy of agents generally is understood as the autonomy to choose autonomously the means to achieve some goal, i.e. autonomy of an agent determines its behavior on a lower level. Therefore, amplification of fluctuations must be given specific attention in order to build self-organizing MAS.

Communication can be based on direct interactions between individuals using communication channels. Communication also can happen indirectly by changing the environment. A soccer-playing robot that kicks the ball changes the environment to all the other players, causing different activities. Stigmergy, from the Greek words *stigma* "sign" and *ergon* "work": the work performed by agents in the environment guides their later actions, has been coined to address these indirect communications by environmental changes that potentially lead to the system's self-organization. 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 [10]. Such techniques are common in biological distributed decentralized systems such as insect colonies [11]. 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 1, the two distinctions are orthogonal.

Table 1. 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

2.3 A Formal View of Self-Organization and Emergence

A very first step towards a theory of self-organizing systems with emergent functionality is to rigorously define the basic concepts. [14] differentiates between self-organization and emergence as two distinct system properties, with self-

organization being a phenomenon of interrelationships between components at the same level (micro-level) and emergence referring to an inter-level process appearing on the macro-level.

As mentioned above and used in expressions such as "self-organization," the word "organization" has distinct, but related, meanings.

Combining the views 1 and 4 of section 2.1 and observing that in real organizations, like enterprises, generally, one has not one formal organizational structure but rather a network of interwoven relationships, formal and informal ones, communication structures, decision structures etc., one has to state that any organizations consists of a multitude of organizational structures. The following formal definition accounts for this.

Organization as a social institution. A System Σ is an ordered pair (V,S), $\Sigma = (V, S)$, where

$V = \{v_1, v_2, \ldots\}, V \neq \varnothing,$

$S = \{S_1,S_2, S_3,.., S_m\}$ a finite set,

each element S_i being a finite set of relations of order i:

$S_i = \{R_1^i, R_2^i,..., R_{mi}^i\}, m_i \geq 0,$

$(m_i = 0$ corresponds to $S_i = \varnothing)$,

$$R_k^i \subseteq \prod_{j=1}^{i} V, \quad k = 1,..,m_i .$$

If V is finite, i.e. $V = \{v_1, v_2, ..v_n.\}$ Σ *is* called an *organization*.

Relations S_0 of order 0 might be interpreted as attributes of the elements v_i. Since different attributes contribute to different degrees of specialization the organizational variable *"specialization"* (see 2.1 above) can be measured by S_0.

This formalization of the concept of an organization permits us to say that one organization respectively a set of elements $V = \{v_1, v_2, ..v_n.\}$ is "more organized" than another (or than the same set V at a different time). Different detailed definitions for the set S are possible and by set-inclusion relate immediately to the concept of the "*degree of organization*", which might be understood as any measure of the number and intensity of set of relationships S.

Also, any subset of $\Sigma = (V, S)$, i.e. a subset of $V = \{v_1, v_2, .., v_n.\}$ or one of $S = \{S_1,S_2, S_3,.., S_m\}$ immediately relates to the concept of a sub-organization.

The term *environment* must be explained too. It always relates to a given system respectively a given organization and denotes the set of variables and influences that act onto the system but cannot be influenced directly by the system itself.

Organizing, i.e. organization as a process. Organization is the process of building or, more generally, of changing the set S of relationships within the set $V = \{v_1, v_2, ...\}$ of organizational members. Also, any changes that might happen to the set V itself, i.e. if any members are entering or leaving V, causes the need of adapting the set S of relationships, and, thus, is some kind of organization.

Definition 1 and 2 do not allow characterizing the *appropriateness* of any organization, i.e. how good the organization is able to respond to any requirements or changes caused by the environment. Clearly, regarding the myriads of consultants who live in selling some organizational concepts, one should recognize that there is no general valid answer which environmental change to the best results in a specific mix of organizational variables. Nevertheless, in the large, some answers might be given. These answers relate to *information availability*, *length of decision paths*, *dynamics of environment* and *internal variety*, as is explained in chapter 3 in more detail.

With this understanding of *"organization"* it is possible to define *"self-organization"* as a process that adapts a given organization by internal forces alone, i.e. without an external change of structures, *because of an observed mismatch to environmental requirements:*

(i) Cause of any self-organization to take place is some environmental change and an information flow from the outside into the system/organization.

(ii) The mismatch of organizational responsiveness to the external requirement is recognized.

(iii) The system re-organizes by itself (not by external intervention) to increase fitness.

Internal variety is a concept used in *cybernetics* to address the ability of a system to respond adequately to external requirements. With respect to the concepts introduced so far one can define *internal variety* as the ability of a system for self-organization

Any organization observed from the outside of the system is called *emergence.* Emergence relates to the behavior of the organization on the macro level, where the relationships S and the specific functioning of the elements of the organization in question cannot be seen. To achieve greater precision, it is proposed 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 (Fig. 1). Self-Organization is the organizing process among elements *within a level.* 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.

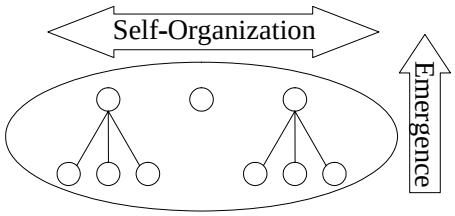

Fig. 1. Comparing Self-Organization and Emergence.

Emergence is defined as a co-notation of organization and in so far of self-organization as well. Emergence (as the term is used here) 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.

2.4 Measuring Self-Organization and Emergence

Since the perception of organization and thus self-organization is highly domain and even system dependent, no globally applicable metric can be provided, though general guidelines may be offered. But, typically, self-organization leads to the emergence of dynamic structures in some topological space. For instance, ants that organize in the

foraging for food create paths that are only visible in the spatial distribution of the ants in space and time. And measures that are based on this distribution would be able to detect the emergence of organized structures, but they would not be able to tell whether the structures emerged through self-organization or from external control.

One such measure that is useful in measuring the emergence of structures in some topology is the Information (or Shannon) Entropy [5]. As demonstrated in [12], entropy measures may be defined over various spatio-temporal distributions that reflect the emerging structures. For instance, one could measure the spatial distribution of the ants across the space around the nest. Initially, before the first path is formed, the entropy would be high because the ants are exploring the space randomly. Once the food is found and ants are recruited to the path, the entropy will collapse.

While such distribution measures at the system level are appropriate for the system architect or evaluator to determine the degree of organization in the system, those classes of measures are not directly accessible to the individual agents with only local access to state information. But, for various local and distributed learning and optimization techniques, it would be very useful to provide the individual agent with a view into the current degree of organization at the system level.

In several previous research projects, we found it useful to apply a derivative of the Information Entropy to an agent's non-deterministic decision process itself. The Option Set Entropy measures the degree to which the agent actually uses information (regardless which information) to select among alternative options in a decision step. If the decision is not using any information, it is essentially random and the entropy is highest. A deterministic decision, whether or not it is the best one for the overall system, results in the lowest entropy.

Assume that in a particular decision step at time t, an agent has N options to choose from. Furthermore, assume that the decision process of the agent assigns each option a probability $p_{1...N}$ to be selected and that the actual decision follows these probabilities (e.g., spins a weighted roulette wheel). Deterministic decisions would assign a probability of 100% to one option and zero to all the others. Then, we define the Option Set Entropy (OSE) as: $OSE(t)=-\Sigma(p_i*log(p_i))/log(N)$. Dividing the standard entropy by the logarithm of the number of options available in one step normalizes the Option Set Entropy to a range of [0,1] and thus provides a clean way to cope with the variation of the number of options over time.

In the case of the foraging ant, the set of options would be a collection of bins for the various direction it could move towards next. In the random exploration mode (no path formed), all bins would be equally weighted and thus the OSE would approach one. Along the path, one direction would clearly stand out (highest pheromone gradient) and the OSE would be significantly lowered. In [4] we demonstrate how such a locally accessible metric can be used to globally optimize the emergent system performance of a distributed graph-coloring algorithm.

3. Design and Modeling of Self-Organizing Software Systems

In the following, we propose a number of application domain features, which may suggest the use of self-organizing approaches to systems design, we discuss the use of stigmergy as a tool for managing large-scale distributed applications, and we present a collection of principles that may guide systems engineers in their application design.

3.1 Where Would One Want to Use Self-Organization and Emergence?

Five domain features indicate the appropriateness of self-organizing approaches: discreteness, deprivation, distribution, decentralization, and dynamism.

Discrete: An organization had been introduced as consisting of a finite and thus discrete set of elements $V = \{v_1, v_2, .., v_n\}$. In so far, it is easiest to apply agents or any other organizational concept (whether self-organizing or not) to a domain if the domain consists of discrete elements that can be mapped onto the set V.

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 nano-technology 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.
- Resource constraints tend to support high degrees of the organizational variable "specialization" since one is forced to take special purpose components. This trend, in turn, forces architectural designs (configuration) which increases the need for coordination.

Several characteristics of self-organizing systems make them good candidates for deprived environments. For example:

- Interactions among system components are typically local in some topology. 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 in geographically distributed systems, 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. In general, this is done by making use of environmental markers (stigmergy) allowing for very simple communication mechanisms. Also, this process guarantees that inconsistencies remove themselves within a specified time horizon, without the need for complex truth-maintenance procedures.

Distributed. The notion of "local interactions" is central to our definition of self-organizing systems with emergent features. 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 self-organizing approaches such as path planning [13], pattern recognition [4], sensor network self-organization, and ant-colony optimization [6], 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 [8], 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.

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.

Self-organization can be a poor choice for applications that require centralization. Particularly, systems that need a high degree of control requiring an extensive set of rules governing systems behavior (high degree of the organizational variable "formalization") are not suited for self-organization and its inherent fault-tolerance.

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.

(i) 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.

(ii) A frequent role of the central element is to mediate interactions among lower-level nodes (as in the mediator architecture [7]). This technique may actually lengthen the communication path between two nodes, leading to undesirable delays as messages travel up, then back down, the hierarchy.

(iii) The central element and the communication paths leading to it are vulnerable to attack or failure, making the system less robust than a self-organizing 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). Techniques for self-organization and emergence are a natural candidate for implementing decentralized architectures.

Dynamic. A system is dynamic if its requirements change during its lifetime. The emergent behavior that is characteristic of self-organization 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 self-organization by way of the distinction between centralized and decentralized architectures. If the system changes slowly, non-self-organizing 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 an architecture for self-organization and emergence.

Obscurity reflects the degree to which the original designer can anticipate later requirements. 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. Self-organizing systems are much better at enabling a system to satisfy requirements that would be surprising to its original designer.

3.2 Stigmergy Supports the Emergence of Self-Organized Structures

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.

Stigmergic mechanisms have a number of attractive features, particularly for self-organizing systems.

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

(i) The agents are easier to program and prove correct at the level of individual behavior.
(ii) They can run on extremely small platforms (such as microchip-based "smart dust" [15]).
(iii) They can be trained with genetic algorithms or particle-swarm methods rather than requiring detailed knowledge engineering.

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 [1]. 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.

3.3 Design Principles for Self-Organizing Systems

From the discussion of application characteristics and in support of stigmergic coordination processes, we derive the following collection of design principles. Naturally, this list is incomplete and the "principles" are more guidelines to be taken into consideration by the system's architect. But, nevertheless, analysis of naturally evolved self-organizing systems with emergent functions (e.g., social insect colonies, human economies) show similar "principles" are favored by the natural selection process too.

3.3.1 Design Principles Regarding the Agent Population(s)

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.

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.

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.

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.

3.3.2 Design Principles Regarding the Agent Interactions

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. This principle corresponds to the requirement of "multiple interactions" mentioned in section 2.2 establishing some kind of communication system that provides social learning.

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").

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 as their decisions are taken probabilistically, breaking the symmetry among them and enabling information flows that can sustain self-organization.

3.3.3 Design Principles Supporting the Emergence of Desired Functions

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.

(i) Let each agent support multiple functions, just as for instance ants may participate in foraging for food, tending to the brood or constructing the nest.
(ii) Let each emergent system-level function require multiple agents to avoid single points of failure and to keep individual agents simple (rather than a single agent being complex enough to provide the entire function).
(iii) 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.

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 [16]. 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 [3] that agents can use to make intelligent decisions about bidding in resource allocation problems. Following our definition of self-organization as a process that adapts the organization of a system to environmental changes, one has to design a mechanism that favors the right kind of local changes at the agent level to optimally change the organization at the system level.

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. Summary

This paper deals with self-organization and with concepts in the context of self-organization. Since self-organization is a particular aspect of organization an understanding of self-organization presupposes that of organization. Following [9], the paper offers a more phenomenal characterization of organization by the organizational dimensions/variables. This formulation provides a more natural understanding of the topic and the particular values of the organizational variables that support self-organization and the verbal characterization of an organization is augmented by a rigorous formal definition, which helps in comparing different organizations and in defining the concepts self-organization, internal variety emergence and stigmergy. It also supports measuring of self-organizations and emergence. For the latter the concept 'Options Set Entropy' is proposed. It is based on Shannon's information entropy and actually is a measure of the information used to take a decision.

In parallel to these more theoretical concepts the concrete question is analyzed, which environmental properties support respectively hinder the deployment of self-organizing software systems. And the question is answered what the characteristic features are that self-organizing software must have. In doing so, specific design principles for self-

organizing software and for the emergence of desired functions are given.

Acknowledgement

We have to thank Professor Manish Parashar, Professor Hong Tang and Professor Xingyu Wang for their valuable comments and improvements.

References

[1] P Ball,The self-made tapestry: Pattern formation in nature, Princeton, NJ, Princeton University Press, 1996.

[2] E Bonabeu, M Dorigo and G Theraulaz, Swarm Intelligence. From Natural to Artificial Systems, Santa Fe Studies in the Sciences of Complexity, New York, Oxford 1999.

[3] S A Brueckner and H V D Parunak, Swarming agents for distributed pattern detection and classification. Proceedings of Workshop on Ubiquitous Computing, AAMAS 2002, Bologna, Italy, 2002, http://www.altarum.net/~vparunak/PatternDetection01.pdf (checked on 2006-08-01).

[4] S Brueckner and H V D Parunak, Information-driven phase changes in multi-agent coordination. Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, 2003, pp. 950-951, http://www.altarum.net/~vparunak/AAMAS03InfoPhaseChange.pdf (checked on 2006-08-01).

[5] Th M Cover, J A Thomas. Elements of information theory, 2nd edition, New York: Wiley-Interscience, 2006.

[6] 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, Vol. 26, No. 1, 1996, pp. 1-13.

[7] B R Gaines, Mediator research program, http://ksi.cpsc.ucalgary.ca/projects/Mediator/, 1995.

[8] M Heusse, S Guérin, D Snyers, and P Kuntz, Adaptive agent-driven routing and load balancing in communication networks. Advances in Complex Systems, Vol. 1, 1998, pp. 234-257.

[9] A Kieser, P Walgenbach, Organisation, 4. überarbeitete und erweiterte Aufl., Stuttgart 2003.

[10] 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, http://citeseer.ist.psu.edu/cache/papers/cs/25995/http:zSzzSzpolaris.ing.unimo.itzSzZambonellizSzPDFzSzSwarm.pdf/co-fields-towards-a.pdf (checked on 2006-08-01)

[11] H V D Parunak, 'Go to the Ant': Engineering principles from natural agent systems. Annals of Operations Research, Vol. 75, 1997, pp. 69-101. http://www.altarum.net/~vparunak/gotoant.pdf (checked on 2006-08-01).

[12] H V D Parunak and S Brueckner, Entropy and self-organization in multi-agent systems. Proceedings of The Fifth International Conference on Autonomous Agents (Agents 2001), Montreal, Canada, ACM, 2001, pp. 124-130, www.altarum.net/~vparunak/agents01ent.pdf (checked on 2006-08-01).

[13] H V D Parunak, M Purcell, and R O'Connell, Digital pheromones for autonomous coordination of swarming UAV's. Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference, Norfolk, VA, AIAA, 2002, www.altarum.net/~vparunak/AIAA02.pdf (checked on 2006-08-01).

[14] H V D Parunak and S A Brueckner. Engineering swarming systems, F Bergenti, M.-P Gleizes, and F Zambonelli, eds., Methodologies and Software Engineering for Agent Systems, Kluwer, 2004.

[15] K Pister, Smart Dust, Autonomous sensing and communication in a cubic millimeter, 2001, http://robotics.eecs.berkeley.edu/~pister/SmartDust/ (checked on 2006-08-01).

[16] D Wolpert and K Tumer, An introduction to collective intelligence. Technical Report NASAARC-IC-99-63, NASA Ames Research Center, 1999.

Author Bios

Sven Brueckner is a Senior Systems Engineer in the Emerging Markets Group at NewVectors. He has been active in the field of multi-agent system's research for more than ten years. His doctoral thesis "Return from the Ant – Synthetic Ecosystems for Manufacturing Control" researches the theoretical foundations of agent system design and applies its findings to complex manufacturing control systems, for which he was awarded the degree Doctor rer. nat. by Humboldt University Berlin, Germany in Summer 2000

After joining ERIM, a predecessor company to NewVectors in spring of 2000, Dr. Brueckner has been a technical lead, Principal Investigator or Project Manager on several agent-focused research and development efforts, such as DARPA-supported research into air-combat coordination, swarming control of unmanned autonomous vehicles, self-organizing management of mobile ad-hoc networks, or polyagent models for adversarial reasoning.

Currently, Dr. Brueckner leads a project in the Office of Naval Research (ONR) Counter-IED Basic Research program, and is a technical lead and project manager for a Disruptive Technology Office (DTO) funded project on intelligence analysis modeling and support. He also spearheads Altarum's work package in the NIST/ATP DBDS project, which develops a novel decision support system for car body design.

Dr. Brueckner has authored over 20 papers on agent-based and complex systems' theory and application and is inventor on four recent patents or preliminary patent filings.

Hans Czap, born in 1945, studied Mathematics at Univ. Wuerzburg (Germany), SUNY (Oneonta, NY, USA), Dundee (Scotland). 1974 he graduated with the Ph-D degree at Univ. of Wuerzburg. 1983 he became professor for Business Information Systems at University of Goettingen.

In 1985 he accepted the chair for Business Information Systems at University of Trier (Germany). He is founding president and during 1986 – 1990 he had been president of Int. Assoc. for Terminology and Knowledge Transfer. 1993 –

1998 he was member of the board of executives of International Society for Knowledge Organization (ISKO). 1997 he became first CEO of Center of Health Care Management at Univ. of Trier (until 2003). For a two year period starting in 2000 he had been chair of Scientific Commission “Public Management” of Assoc. of Univ.-Professors for Business Administration.

In the period 2000 –2004 he was managing director of newly founded Competence Center for E-Business at University of Trier.

His scientific interests cover a broad range: systems theory, cost accounting, information systems, neural networks and multi-agent systems. He is author, editor or co-editor of 15 monographs and has published more than 100 papers.