Software Engineering for Self-Organizing Systems

H. Van Dyke Parunak and Sven A. Brueckner

Vector Research Center, Jacobs Engineering Group
3520 Green Court, Suite 250
Ann Arbor, MI 48105
{van.parunak, sven.brueckner}@jacobs.com

Abstract. Self-organizing software systems are an increasingly attractive approach to highly distributed, decentralized, dynamic applications. In some domains (such as the Internet), the interaction of originally independent systems yields a self-organizing system *de facto*, and engineers must take these characteristics into account to manage them. This review surveys current work in this field and outlines its main themes, identifies challenges for future research, and addresses the continuity between software engineering in general and techniques appropriate for self-organizing systems.

Keywords: software engineering, self-organization, distributed systems, decentralized computing, emergent behavior

1 Introduction

A few decades ago, the idea of self-organization was an intriguing option in the design of a computer application, and its proponents could engage in spirited debate with more classical views of software structure. Today, in many domains (particularly those based on computer networks), the question is no longer whether to use self-organization. Real-world open systems with thousands of autonomous components do in fact organize themselves, for better or for worse. The challenge before us is to understand this dynamic and learn how to manage it [123].

There is no lack of activity around software systems that in one way or another control themselves without direct human intervention. In an attempt to focus this review, we distinguish three kinds of systems: autonomous, self-adaptive, and self-organizing.

An *autonomous* system is one that senses and responds to its environment. The vast research world of agent-based and multi-agent systems is concerned with such systems. Self-organizing systems are made up of autonomous systems, but not every autonomous system is self-organizing.

A *self-adaptive* system is an adaptive system that responds to change without intervention by its creator (thus the "self"). The change may be in the environment, or it may be within the system itself (for example, a fault condition). The difference between a self-adaptive system and an autonomous system is a matter of perspective. If

the environment were static, there would be nothing for an autonomous system to respond do, so every autonomous system in fact adapts in one way or another to environmental change. When we call a system *self-adaptive*, we imply that the change with which the system must cope is unusually large and potentially disruptive.

The *self-organizing* system is a special kind of self-adaptive system. We emphasize two points of refinement.

First, as the use of the term "organize" suggests, a self-organizing system consists of multiple components that can change their interrelations. A single agent could be self-adaptive, but we would not call it self-organizing. Definitions of self-organization often invoke the notion of disorder or "entropy" across the population of elements [51, 100].

Second, we are particularly interested in systems whose response to change does not require centralized reflection. Much work on self-adaptive software requires the system to have an internal representation of its goals [60], or a model of its own architecture [87], or a set of explicit policies ("in case of X, do Y") [40, 48], to guide its adaptation. We are focused on systems that require neither such explicit representations nor a central module to manage the change in the system in response to disruption. Because of this distinction, a hierarchical feedback control system, while composed of many different parts, would still be considered self-adaptive rather than self-organizing.

While this distinction is important and useful [83], we will consider some work that does not completely meet this objective. After all, we are dealing with software *engineering*, not software *science*, and progress often depends on drawing inspiration from many sources [40, 142].

In Section 2, we review the state of the art in software engineering for self-organizing systems. Section 3 summarizes some major trends that we see in current practice. Section 4 outlines directions for future research. Section 5 summarizes how this particular flavor of software engineering relates to the broader field, and Section 6 concludes.

2 State of the Art

In this section, we begin by reviewing some of the immense literature in this field, then survey applications of self-organizing systems and some of the main mechanisms that they employ.

2.1 Literature

Our focus here is on survey articles or programmatic discussions. Later sections of this review will consider more specific studies.

While we distinguish self-organization from self-adaptation, we stand on the shoulders of extensive work in autonomous and self-adaptive software. The classic notion of a feedback control loop can be traced back to the nineteenth century [82], and it was natural for the idea to be applied to computer programs, largely under the

inspiration of Norbert Wiener [146]. The flavor of adaptive control in robotic and manufacturing systems was captured in NIST's Real-time Control System (RCS) reference architecture [1, 2]. Later, IBM's Autonomic Computing Initiative [69, 73] sought to apply these techniques to purely informational systems.

Autonomous systems are the focus of much robotic research, and application concerns have led to recent efforts to define a scale of autonomy [67] and develop methods to test a system's autonomy [70].

Self-adaptive software has been the object of two recent seminars at Schloss Dagstuhl [29, 52], and a special issue of the Journal of Systems and Software is in preparation on this topic [145]. The topic is the object of a careful review article [114], whose approach (focusing on the functions of monitoring, detecting, deciding, and acting) very clearly captures the reflective nature of self-adaptation as opposed to self-organization.

The design and control of self-organizing software per se was the focus of four editions of the ESOA (Engineering Self-Organising Applications) workshop [21, 24, 25, 41], and is treated in Gershenson's recent dissertation [50], and a wide range of shorter studies will be identified in later sections of this review. The areas of self-adaptive and self-organizing systems together are the focus of the ongoing IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO) [116].[1]

2.2 Applications

Self-organization has been applied to a wide range of problems. As noted in the introduction, self-organization is unavoidable in distributed systems, especially open ones, such as networks [15, 61, 64, 123] and water distribution [42], and highly desirable in managing large numbers of robots [53-55, 118, 120] and in agile manufacturing settings [19, 94, 109, 132], where it competes with hierarchical control systems, including holonic schemes [26, 133] that we would consider self-adaptive but not self-organizing. In purely informational settings self-organization has been used to coordinate multiple theorem provers [36], to enable documents to organize themselves [104] and find likely users [20, 62], and to reassign tasks among agents [31, 88]. Mechanisms inspired by wasps and termites have been demonstrated for self-organized construction of physical systems [140].

2.3 Mechanisms

A wide range of instances of self-organization in nature have been isolated and characterized to the point that they can be applied in artificial systems [14, 27, 93]. These derive mostly from social animals (pheromone systems [72, 99, 108, 109, 117, 119, 134, 137], stimulus-based load balancing [140], insect clustering [58, 59, 76, 84, 104, 138], firefly synchronization [130]), but markets [32, 106, 107] and physical systems

[1] Not all studies that take the name "self-organizing" satisfy our definition of the field as distinct from self-adaptation." We would class some of the work reported in venues devoted to "self-organizing software" as in fact only self-adaptive.

such as potential fields [46, 80, 81, 128, 143] have also been invoked. While these mechanisms can be characterized in terms of feedback control, in their natural settings they are highly decentralized and do not rely on explicit models of the structure, goals, or policies of the overall system, thus qualifying as self-organizing and not just self-adaptive.

2.4 Reflection on the State of the Art

While self-organizing solutions have been widely explored, they tend to have two limiting characteristics [142]. First, most applications demonstrate the capabilities of a single mechanism, and do not consider the potential interaction of a toolkit of mechanisms. Second, among software engineers there is relatively little work on the theoretical foundations of these mechanisms. We will return to these themes when we outline directions for future work.

3 Outline of Main Trends

Several general trends are apparent from this brief and highly selective review: decentralization, openness, imitation of nature, and reliance on simulation. To a large degree, these characteristics reflect our definition of "self-organization," but they are common enough in practice to justify focusing on systems that exhibit them.

3.1 Decentralization

Since we distinguish self-organization from self-adaptation partly by the multi-component decentralized nature of the former, decentralization is not surprising, but it is worth refinement and reflection.

Decentralization is not a black-or-white dichotomy. It is useful to distinguish three levels, and in a highly populous system with heterogeneous members, one can imagine gradations and combinations along this spectrum.

At one extreme, and outside our purview, are centralized systems, in which all decisions are made at a single location. We include here not only monolithic systems, but also hierarchical feedback control systems [1]. Holonic systems were originally motivated by emergent dynamics over a hierarchy that defines scale rather than control [75], but engineered holonic applications often look very much like hierarchical control [16, 26, 113]. We can view such systems as centralizing two kinds of information: declarative information about the current state of the system, and imperative information that determines next steps.

At the other extreme are systems in which each entity interacts only with those in its local vicinity. Its neighborhood defines its view of the state of the world (declarative information), and it can only act within that narrow purview (imperative information).

At an intermediate level, the state of the system (declarative information) is collected centrally and made available to all components, but all action is taken locally.

In some cases, one can detect movement along this cline. For example, one team began working with multiple interacting theorem provers in a centralized setting [37], but then revised the system to use global information but only local decisions [36]. In recent work in other domains, their design has moved to a distribution of both declarative and imperative information [42]. A motivation for this movement is the increasing need for real-time response [38], which can be hindered if multiple layers of hierarchy need to be queried to make a decision [144].

There is a limitation to complete localization of interaction: it limits look-ahead. "It only functions acceptably when the (recent) past is representative for the (near) future" [131]. Predictive mechanisms have been proposed to address this problem [30]. A particularly interesting class of problems uses a second-level self-organizing system to make these predictions through a model of the world in which interactions are spatially local but are allowed to evolve faster-than-real-time into the future [63, 97, 102].

Market systems represent an interesting segment of the centralized-decentralized spectrum. Classical Walrasian markets depend on posting bids centrally so that agents can make local decisions [32], thus embodying our midpoint of global declarative information and local imperative information. However, an alternative form of market, Edgeworth barter [4], allows agents to interact pairwise, and still guarantees global convergence. This form of market is completely distributed, and has been applied to problems of distributed constraint optimization [106, 107].

3.2 Openness

In building a self-organizing system from the ground up, one can impose homogeneity on the elements. However, the kinds of self-organization that are being imposed on us (say, through the internet) force us to deal with systems whose elements do not conform to a single blueprint.

Openness greatly increases the complexity of a system. Any single element needs to be prepared to interact with everything outside of its own boundary, which now includes not only other elements that are like itself, but also technical, geographic, political, social and economic realities [39, 123]. Because we cannot predict all of these influences in advance, the line between the preparation of the system (its specification, design, implementation, and testing) and its operation is greatly blurred, a distinction to which we shall return.

In a closed system, entities can be designed to interact directly with each other. The need to cope with an open system has led researchers to focus on a common framework or infrastructure. Any agent that can interact with this infrastructure can be included in the system. At the most primitive level, the physical world is the infrastructure, and agents must have physical sensors and actuators to deal with it (an approach exploited in the axiom that "the world is its own best model" [18]. In the natural world, animals sometimes use the physical world to hold arbitrary markers (for instance, insect pheromones), which is one form of stigmergy [56] (the other being functional changes in the world). Disembodied agents require a computational

framework, and a major line of research [3, 135] is focused on designing such frameworks and their component mechanisms [90-92, 136].

The framework approach to openness imposes a "lowest-common denominator" on all interacting components. There is a trade-off between the simplicity of the common interface to the framework and the range of entities that can interact. A very simple interface supports the widest range of entities, but also limits the amount of information that the entities can exchange [131]. For example, a market is a framework that permits open interaction among a wide range of economic actors by reducing all considerations to a single scalar, price, discarding much detailed information along the way. This consideration has led to the development of relatively sophisticated interaction languages, such as tuple spaces [28, 78] and highly structured symbolic "pheromones" [109].

Openness has implications for the security of a system, in two opposing directions. On the one hand, the more open a system is, the fewer restrictions are imposed on an element that seeks to participate in it, and the easier it is for malicious elements to insert themselves into the system's operation. On the other hand, the more decentralized and localized a system's decisions are, the harder it will be for a malicious element to understand and manipulate the overall state of the system. Roughly, open systems are easier to infiltrate than closed ones, but tend to limit the extent of damage that can be done. On this subject, engineering of self-organizing systems needs to draw extensively on work on cyber-security and trust.

3.3 Imitation of Nature

We have already observed (Section 2.3) that mechanisms for self-organizing systems tend to be drawn from nature, and in particular from biological systems. This tendency can be traced directly to the problem of openness, which organisms must confront in order to survive. The more sophisticated the organism, the more structure it can impose on its own environment, and the less open that environment becomes to other entities. A parade example is the rich linguistic mechanisms that humans use to coordinate with one another. Computational mechanisms modeled on human consciousness and linguistic interaction are the holy grail of AI research, but still beyond our grasp. Artificial versions of cognition have been described as autistic [131] and schizophrenic [65, 125], "idiot savants" with focused capability but lacking adaptability. This may lie behind the preference for simpler insect models in self-organizing software [131], though in fact a more careful analysis suggests humans often use the same kind of simple mechanisms that insects do [95].

3.4 Simulation

Simulation, rather than formal analysis, plays a prominent role in the engineering of most current self-organizing systems [148]. The complexity of these systems makes the development of formal models difficult [65]. In fact, a set of even very simple agents interacting with one another has the computational power of a Turing machine

[44], or perhaps even more [139], and by Rice's theorem [112], any non-trivial feature of such a system is formally undecidable.

Some proponents of simulation argue that a simulation, being a computer program, is a partial recursive function, and thus refuse to recognize any distinction between simulation and formal analysis [45]. The issue is not one of formal structure, but of insight. A computer program, while every bit as formal as a proof, has a very different structure. Most program structures are algorithmic: first do X, then do Y, and then do Z.[2] Such a structure does not lend itself to determining properties such as whether the system will halt, how rapidly it converges, how thoroughly it explores the space of possible behaviors, and whether its equilibria are stable or unstable. The results of Edmonds and Bryson [44] warn that in general such characterizations are unattainable, but as with many formal results, there are special cases where formal methods can support the engineering of self-organizing systems, as we shall see in the next section.

4 Challenges for Future Research

The themes of current systems highlight a number of opportunities for future research. These opportunities are not unexplored, but represent the cutting edge of current work in this field. We consider first the problem of composing more complex systems, then the challenge of characterizing and controlling an existing system, and finally the objective of understanding self-organizing systems formally.

4.1 System Composition

In Section 2.4, we observed that most current applications focus on a single mechanism or phenomenon. Weyns [141] demonstrates the integration of multiple mechanisms in an industrial application, but such hybrid approaches are the exception rather than the rule. Beal suggests that "the composition of phenomena into a larger complex system is rather understudied" [9], and identifies three areas that must be pursued.

First, self-organizing phenomena must be reduced to primitives with well-characterized properties and interfaces. The idea of method fragments [111] is to decompose an approach into fragments using SPEM [89] as the underlying formalism, so that they can be reused and combined with each other. A small but growing circle of activity in defining self-organization mechanisms as software design patterns [35, 47, 63, 72] is also a step in this direction.

Second, we need means of composition that allow self-organizing phenomena to be combined with predictable results. Current efforts that emphasize the centrality of frameworks [3] and architectures [141] are seeking to address this problem, but the need for "predictable results" awaits advances in formal analysis (Section 4.3).

Third, we need means of abstraction that allow details of a complex self-organizing system to be hidden when engineering or analyzing larger subunits. This characteris-

[2] Declarative languages are an exception, and represent an important research topic.

tic, identified by Simon as critical to artificial systems [126], is also likely to depend on further formal insights.

The imitation of nature that is so common in identifying individual mechanisms for self-organization holds promise here as well, if we shift our focus from the individual organisms or species to the level of the ecosystem [9, 19, 71, 74, 105, 115, 137]

4.2 System Characterization and Control

People build systems to perform some task, and need to be able to characterize their behavior and control them.

At design time, we need to understand a range of trade-offs that self-organizing mechanisms impose. These include [38] locality vs. optimality, optimality vs. flexibility, scalability vs. efficiency, efficiency vs. centralization, centralization vs. decentralization, exploration vs. exploitation, and greediness vs. purposefulness. (All of these trade-offs presume that we have well-defined measures of each property, itself a major research challenge). Depending on the requirements of the application, certain regions of each of these scales may qualify as faulty behavior, and techniques of safety engineering can be adopted to identify and avoid them [39].

As the system is operating, we need ways to characterize its behavior. Observing and analyzing the series of events that it generates is one way to gain this insight [68]. One important challenge in this task is that while the nature of the system's behavior as acceptable or unacceptable manifests at the system level, our self-organizing agenda requires us to focus on locally observable phenomena. Information theoretic measures such as the entropy over agent options [22] or over signals passing between agents [64, 66] have proven a promising local window into global system behavior.

Like behavior characterization, behavior control is difficult in a decentralized setting, and is not widely explored [142]. A system of local constraints with attributes defined over component interfaces [49] is one promising way forward. Another is to deploy a control swarm in parallel with the functioning swarm [83]. In some cases centralization may be unavoidable, and a fruitful avenue of exploration is how to combine centralized control where necessary with local control most of the time [40].

4.3 Formal Analysis of Self-Organizing Systems

Attempts to gain a formal purchase on self-organizing systems usually involve one or more of three critical dimensions: a vertical dimension ("emergence") that relates lower-level and higher-level behaviors, a horizontal dimension ("organization") that relates entities at a single level to one another, and a temporal dimension ("dynamics") that explores how the system develops through time.

Emergence.—Perhaps the most widely recognized problem in dealing with self-organizing systems is emergence [147], which we define [103] as system-level behavior that is not explicitly specified in the individual components. Abstracted from software, the problem has a long history, forming the central focus of the discipline of

statistical mechanics (which seeks to relate the observed characteristics of materials at human scale to the interactions of atoms and molecules). This perspective allows the application of concepts such as entropy [57, 64, 100, 122], phase shifts [23, 121, 124], master equations [13, 79], and universality [101] to multi-agent systems. There are further insights to be gained from this approach. For example, the renormalization group [12] has the potential to illuminate understand discontinuities in the behavior of a self-organizing system. By considering the system as it approaches certain limits (for example, low agent density and high number of agents, allowing the use of a gas model [5]), we can place bounds on system characteristics of interest, offering "thermodynamic guarantees" [123] of system behavior.

The mapping from micro to macro behaviors is not symmetrical. To derive the macro behavior from the micro, we run simulations, or (in the appropriate limits) apply techniques from statistical mechanics, and these techniques are useful in system verification. Earlier in the design process, given a specified macro behavior, we need to find micro behaviors that will yield it. The best approach to this problem that we know consists of various forms of generate and test, such as synthetic evolution, which has been applied successfully to define local agent behaviors satisfying a macro specification [17, 96, 117]. This approach requires a system architecture whose representation lends itself to such evolutionary search [19].

An interesting facet of the vertical problem is the level at which goals are satisfied. Individually selfish agents may not yield good results at the group level. We need to develop ways to define and achieve "group-selfish behavior" [131], in which the system as a whole pursues objectives that may not be optimal from the point of view of the components. Insights into this objective may come from biology. The notion of the gene, rather than the individual, as the focus of natural selection [34] can be viewed as a process for favoring a well-defined group of agents (those agents possessing the gene) as opposed to individuals.

Organization.—The horizontal dimension explores the implications of studying which agents can interact with which other ones on the behavior of the whole system. The resulting graph structure is amenable to a variety of formal tools [86]. For example, useful definitions of autonomy and emergence can be formalized in terms of entropy on signals over the edges in the interaction graph [64], and usability can be defined in terms of similar measures on edges connecting the system to users [66]. It has been suggested [122] that the Laplacian spectrum of a network, which captures aspects of the graph's modular and hierarchical structures, may facilitate formalization of the relation between these structures and dynamical processes such as distributed consensus, decentralized coordination and information dissemination [123].

Temporal.—Self-organization is a process that takes place through time, and an adequate formalization of self-organizing systems must support reasoning about the temporal dimension. In many cases, systems need to predict their own behavior in order to adapt appropriately [30, 63, 97, 102, 131], but the nonlinear nature of component interactions means that trajectories diverge over time, leading to a prediction

horizon [98] beyond which any prediction is essentially random. Estimating this horizon is critical to scoping the predictive activity of a system, and quantifying the uncertainty that is inevitable in a self-organizing system [123].

One approach to formalizing the temporal dimension is in defining formal languages to specify system development [7]. (At this point, one may invoke Epstein's insistence on the formal nature of any computer program [45], since higher-level primitives nominated by a programming language do offer a useful abstraction that can give insight to the behavior of the system programmed in the language.) There are a number of examples that could inspire further work in this area, including languages modeling gene network development [43], term-rewriting systems modeling plant growth [110] and its generalization in MGS [129], Coore's Growing Point Language for interconnect topologies [33], Nagpal's Origami Shape Language [85], Werfel's system for distributed adaptive structure generation [140], and Beal's Proto system for spatial computing [8].

5 Relation to Conventional (Software) Engineering

Engineering of self-organizing systems has drawn much from the engineering of conventional software. In this section, we highlight some of the points of continuity and contrast.

Let's begin with **engineering in general**. We have already noted the inappropriateness of the feedback control metaphor for a decentralized approach to self-organization. Nevertheless, the engineering of physical systems has a great deal to teach us. One example is how one handles **noise**. Engineering of physical systems, unlike conventional software engineering, devotes much attention to modeling and quantifying noise in the interfaces between components. Traditional software engineering assumes that noise (i.e., errors) can be eliminated, while other disciplines recognize that it is unavoidable and seek to damp it or provide for graceful degradation [9, 10]. Another example is the adaptation of methods for safety engineering to increasing the robustness and dependability of self-organizing software [39].

The notion of an **architecture** is a powerful way to engage the challenge of system composition and openness, providing a framework for algorithms and identify complementarities and system-level issues [141, 145]. Research on frameworks to provide interaction environments for components [3, 135] is a way to instantiate insights from an architectural approach.

There has been a historical shift in system analysis away from functional analysis and toward **object-oriented** system decomposition. Self-organizing systems benefit from this shift: system functions are usually distributed over many components, and even if some group of components specializes to support a function, that association happens dynamically, rather than being specified in the design [131].

The notion of **design patterns** provides a useful way to abstract individual self-organizing mechanisms so that they can subsequently be recombined in novel ways [127]. The approach has been applied to a number of mechanisms, including market based control [35], gradient fields [35, 72], predictive swarms [102, 144], replication,

collective sort, evaporation, aggregation, and diffusion [47]. It is instructive to observe that the last three patterns are sub-components of a pheromone approach to constructing gradient fields, highlighting both the value of this approach and the need for further development.

Closely related to this work is the application of **SPEM** [89] to facilitate the isolation and integration of **method fragments** [111], illustrated by isolating fragments from Adelfe, CUP, MetaSelf, General Methodology, and SDA, and then recombining them using PASSI.

Finally, engineers of self-organizing systems can take advantage of recent advances in **iterative and incremental development** [9, 77]. It is impossible to anticipate in advance the states accessible to a self-organizing system as it evolves in an open environment. As a result, the line between development and operation inevitably blurs [6]. The system must be specified in terms of desired performance and means of incrementally correcting deficiencies [7], constructing systems that grow and react rather than being constructed and controlled [122]. The need for this perspective is particularly strong in the verification and validation (V&V) of a system. Traditionally, successful completion of V&V is necessary before a system is deployed. Self-organizing systems require mechanisms for "run-time V&V" [11] that can continuously monitor the system's performance as it reorganizes itself in response to unanticipated conditions. It is an open question whether run-time V&V can in fact be done in a fully decentralized manner, or whether some reference to an explicit model of system objectives is necessary. That is, a system can be engineered to organize itself to meet system objectives without carrying a model of those objectives, but it may be the case that it cannot report whether or not it is in fact meeting the objectives unless it has such a model, since the latter task is intrinsically reflective.

6 Conclusion

The engineering of self-organizing software is a challenging domain that has attracted a wide range of creative talent. In spite of the difficulty of the problem and the wide range of approaches, there are consistent themes and well-defined problems to focus future research. As the information universe becomes more distributed and decentralized, the difference between the engineering of self-organizing systems and that of other software will shrink, and the themes that are beginning to manifest themselves in the self-organizing community will be increasingly recognized as staples of software engineering in general.

7 Acknowledgments

This review relies heavily on many colleagues who were kind enough to share their observations on the field, and their own work, with us.[3] We particularly appreciate the

[3] Alphabetically by last name: Bernhard Bauer, Jake Beal, Olivier Buffet, François Charpillet, Jörg Denzinger, Regina Frei, Kurt Geihs, Arnaud Glad, Nicolas Höning,

detailed reviews of the field that several respondents contributed [9, 38, 65, 123, 127, 131, 135, 142]. Even though these reviews are not publicly available, we have borrowed extensively from their ideas and in some cases their wording, and have cited them in order to give appropriate credit. Naturally, we are responsible for how we have combined the ideas that they have so generously shared with us. Think of this exercise as an example of an "open system," in which the components, in this case the contributions of our informants, are allowed to interact in ways that they perhaps did not anticipate. We provide the "infrastructure" for the interaction, and as is often the case in self-organizing systems, the infrastructure makes a great deal of difference in the overall outcome.

In selecting the studies that we cite, we draw heavily on the suggestions of our informants, so our citations should be understood as examples and make no claim to be exhaustive. We look forward to revising this study for publication, and invite the nomination of other work that should be included.

8 References

[1] J. S. Albus. RCS: A Reference Model Architecture for Intelligent Control. *IEEE Computer*, 25(5):56-59, 1992.

[2] J. S. Albus. The NIST Real-time Control System (RCS): an approach to intelligent systems research. *J. Exp. Theor. Artif. Intell.*, 9(2-3):157-174, 1997.

[3] aliCE. aliCE (agents, languages and infrastructures for Complexity Engineering) Home. Bologna, Italy, 2008. http://alice.unibo.it/xwiki/bin/view/aliCE/.

[4] R. Axtell and J. Epstein. Distributed Computation of Economic Equilibria via Bilateral Exchange. Brookings Institution, Washington, DC, 1997.

[5] J. Bachrach, J. Beal, and J. McLurkin. Composable continuous space programs for robotic swarms. *Neural Computing and Applications*, 19(6):825–847, 2010.

[6] L. Baresi, N. Bencomo, B. Cukic, A. Gorla, P. Inverardi, O. Nierstrasz, S. Park, D. Smith, T. Vogel, R. de Lemos, and J. Andersson. Dagstuhl Group C: Process. Dagstuhl, 2010. Available at http://www.dagstuhl.de/Materials/Files/10/10431/10431.SWM12.Slides.ppt.

[7] J. Beal. Functional Blueprints: An Approach to Modularity in Grown Systems. In *Proceedings of the Seventh International Conference on Swarm Intelligence (ANTS 2010)*, 2010.

[8] J. Beal. MIT Proto. MIT, Cambridge, MA, 2010. http://stpg.csail.mit.edu/proto.html.

[9] J. Beal. Software Engineering for Self-Organizing Systems. 2011.

[10] J. Beal and T. F. Knight Jr. Analyzing Composability in a Sparse Encoding Model of Memorization and Association. In *Proceedings of the Seventh IEEE International Conference on Development and Learning (ICDL 2008)*, 2008.

[11] B. Becker, G. Karsai, S. Mankovskii, H. Müller, M. Pezze, W. Schäfer, J. P. S. L. Tahvildari, G. Tamura, N. M. Villegas, and K. Wong. Dagstuhl Group A: Towards

Practical Run-Time V&V (For Self-Adaptive Systems). Dagstuhl, 2010. Available at http://www.dagstuhl.de/Materials/Files/10/10431/10431.SWM10.Slides.ppt.

[12] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman. *The Theory of Critical Phenomena - An Introduction to the Renormalization Group*. Oxford, UK, Clarendon Press, 1992.

[13] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99 (Supplement 3):7280-7287, 2002.

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

[15] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in Telecommunications Networks with "Smart" Ant-Like Agents. In *Proceedings of Second International Workshop on Intelligent Agents for Telecommunications Applications (IATA98)*, Springer, 1998.

[16] L. Bongaerts. *Integration of Scheduling and Control in Holonic Manufacturing Systems*. Thesis at K.U. Leuven, Department of PMA, 1998.

[17] L. Booker. Learning Tactics for Swarming Entities. In *Proceedings of Swarming: Network Enabled C4ISR*, ASD C3I, 2003.

[18] R. A. Brooks. Intelligence Without Representation. *Artificial Intelligence*, 47:139-59, 1991.

[19] S. Brueckner. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control*. Thesis at Humboldt University Berlin, Department of Computer Science, 2000.

[20] S. Brueckner, E. Downs, R. Hilscher, and A. Yinger. Self-Organizing Integration of Competing Reasoners for Information Matching. In *Proceedings of ECOSOA Workshop at SASO 2008*, 2008.

[21] S. Brueckner, S. Hassas, M. Jelasity, and D. Yamins, Editors. *Engineering Self-Organising Systems*. Lecture Notes on Computer Science, Springer, 2007.

[22] S. Brueckner and H. V. D. Parunak. Resource-Aware Exploration of Emergent Dynamics of Simulated Systems. In *Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, pages 781-788, ACM, 2003.

[23] S. Brueckner and H. V. D. Parunak. Information-Driven Phase Changes in Multi-Agent Coordination. In *Proceedings of Workshop on Engineering Self-Organizing Systems (ESOA, at AAMAS 2005)*, pages 104-119, Springer, 2005.

[24] S. A. Brueckner, G. Di Marzo Serugendo, and D. Hales, Editors. *Engineering Self-Organising Systems*. Lecture Notes in Computer Science, 2006.

[25] S. A. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, and R. Nagpal, Editors. *Engineering Self-Organising Systems*. Lecture Notes in Computer Science, 2005.

[26] H. V. Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers In Industry*, 37(3):255-276, 1998.

[27] 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.

[28] M. Casadei, M. Viroli, and L. Gardelli. On the Collective Sort Problem for Distributed Tuple Spaces. *Science of Computer Programming*, 74(9):702-722, 2009.

[29] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Editors. *Software Engineering for Self-Adaptive Systems*. Lecture Notes in Computer Science, 5525, Heidelberg, Springer, 2009.

[30] S.-W. Cheng, V. V. Poladian, D. Garlan, and B. Schmerl. Improving Architecture-Based Self-Adaptation through Resource Prediction. In B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Editors, *Software Engineering for Self-Adaptive Systems*, vol. 5525, pages 128-145. Springer, Heidelberg, 2009.

[31] V. A. Cicirello and S. F. Smith. Wasp-like Agents for Distributed Factory Coordination. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3 (May)):237-266, 2004.

[32] S. H. Clearwater, Editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. Singapore, World Scientific, 1996.

[33] D. Coore. *Botanical Computing: A Developmental Approach to Generating Inter connect Topologies on an Amorphous Computer*. Thesis at MIT, 1999.

[34] R. Dawkins. *The Selfish Gene*. Oxford University Press, 1976.

[35] T. De Wolf and T. Holvoet. Design patterns for decentralised coordination in self-organising emergent systems. In *Proceedings of the Fourth International Workshop on Engineering Self-Organising Applications (ESOA) at AAMAS 2006*, pages 28-49, Springer, 2007.

[36] J. Denzinger and D. Fuchs. Cooperation of Heterogeneous Provers. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999)*, pages 10-15, Morgan Kaufmann, 1999.

[37] J. Denzinger, M. Fuchs, and M. Fuchs. High Performance ATP Systems by Combining Several AI Methods. In *Proceedings of IJCAI-97*, pages 102-107, Morgan Kaufmann, 1997.

[38] J. Denzinger, H. Kasinger, and B. Bauer. Software Engineering for Self-Organizing Systems. 2011.

[39] G. Di Marzo Serugendo. Robustness and Dependability of Self-Organising Systems – A Safety Engineering Perspective. In *Proceedings of the 11th International Symposium on Stabilization, Safety and Security of Distributed Systems (SSS 2009)*, pages 254–268, Springer, 2009.

[40] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky. MetaSelf—An Architecture and Development Method for Dependable Self-* Systems. In *Proceedings of the 25th Symposium on Applied Computing (SAC 2010)*, 2010.

[41] G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Editors. *Engineering Self-Organising Systems*. Lecture Notes in Computer Science, 2004.

[42] F. Dötsch, J. Denzinger, H. Kasinger, and B. Bauer. Decentralized Real-time Control of Water Distribution Networks Using Self-organizing Multi-Agent Systems. In *Proceedings of the 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010)*, pages 223-232, IEEE, 2010.

[43] R. Doursat. The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. *InterJournal: Complex Systems*:1809, 2006.

[44] B. Edmonds and J. J. Bryson. The Insufficiency of Formal Design Methods: The Necessity of an Experimental Approach for the Understanding and Control of Complex

MAS. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 938-945, IEEE 2004.

[45] J. M. Epstein. *Generative Social Science*. Princeton, NJ, Princeton University Press, 2006.

[46] F. Flacher and O. Sigaud. Spatial coordination through social potential fields and genetic algorithms. In *Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior (From Animals to Animats)*, MIT Press, 2002.

[47] L. Gardelli, M. Viroli, and A. Omicini. Design Patterns for Self-organizing Multiagent Systems. In *Proceedings of the 5th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2007)*, pages 123-132, Springer, 2007.

[48] J. C. Georgas and R. N. Taylor. Policy-Based Architectural Adaptation Management: Robotics Domain Case Studies. In B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Editors, *Software Engineering for Self-Adaptive Systems*, vol. 5525, pages 89-108. Springer, Heidelberg, 2009.

[49] I. Georgiadis, J. Magee, and J. Kramer. Self-organising software architectures for distributed systems. In *Proceedings of the First workshop on Self-healing systems (WOSS '02)*, ACM, 2002.

[50] C. Gershenson. *Design and Control of Self-organizing Systems*. Thesis at Vrije Universiteit Brussel, 2007.

[51] C. Gershenson and F. Heylighen. When Can we Call a System Self-organizing? 2003. http://arxiv.org/pdf/nlin.AO/0303020.

[52] H. Giese, H. Müller, M. Shaw, and R. d. Lemos. Abstracts, Dagstuhl Seminar 10431, Software Engineering for Self-Adaptive Systems. 2010. http://www.dagstuhl.de/Materials/index.en.phtml?10431.

[53] A. Glad, O. Buffet, O. Simonin, and F. Charpillet. Self-Organization of Patrolling-ant Algorithms. In *Proceedings of the International Conference on Self-Adaptive and Self-Organizing Systems (SASO09)*, pages 61-70, 2009.

[54] A. Glad, O. Buffet, O. Simonin, and F. Charpillet. Influence of Different Execution Models on Patrollin Ant Behaviors: from Agents to Robots. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 1173-1180, 2010.

[55] A. Glad, O. Simonin, O. Buffet, and F. Charpillet. Theoretical Study of Ant-based Algorithms for Multi-Agent Patrolling. In *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI'08)*, pages 626-630, 2008.

[56] 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.

[57] S. Guerin and D. Kunkle. Emergence of Constraint in Self-organizing Systems. *Nonlinear Dynamics, Psychology, and Life Sciences*, 8(2):131-146, 2004.

[58] A. Hamdi, V. Antoine, N. Monmarché, A. Alimi, and M. Slimane. Artificial Ants for Automatic Classification. In N. Monmarché, F. Guinand, and P. Siarry, Editors, *Artificial Ants: From Collective Intelligence to Real-life Optimization and Beyond*, pages 265-290. John Wiley and Sons, Hoboken, NJ, 2010.

[59] J. Handl, J. Knowles, and M. Dorigo. Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. TR-IRIDIA-2003-24, IRIDIA, 2003. http://wwwcip.informatik.uni-erlangen.de/~sijuhand/TR-IRIDIA-2003-24.pdf.

[60] W. Heaven, D. Sykes, J. Magee, and J. Kramer. A Case Study in Goal-Driven Architectural Adaptation. In B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Editors, *Software Engineering for Self-Adaptive Systems*, vol. 5525, pages 109-127. Springer, Heidelberg, 2009.

[61] 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.

[62] R. Hilscher, S. Brueckner, T. C. Belding, and H. V. D. Parunak. Self-Organizing Information Matching in InformANTS. In *Proceedings of Self-Adaptive and Self-Organizing Systems (SASO07)*, pages 277-280, 2007.

[63] T. Holvoet, D. Weyns, and P. Valckenaers. Delegate MAS Patterns for Large-Scale Distributed Coordination and Control Applications. In *Proceedings of EuroPlop*, 2010.

[64] R. Holzer, H. de Meer, and C. Bettstetter. On Autonomy and Emergence in Self-Organizing Systems. In *Proceedings of Intern. Workshop on Self-Organizing Systems (IWSOS)*, Springer, 2008.

[65] N. Höning. Comments on Software Engineering for Self-Organizing Systems. 2011.

[66] N. Höning and H. La Poutre. Designing comprehensible self-organising systems. In *Proceedings of the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010)*, pages 233-242, IEEE Computer Society, 2010.

[67] H.-M. Huang, K. Pavek, B. Novak, J. Albus, and E. Messina. A Framework For Autonomy Levels For Unmanned Systems (ALFUS). In *Proceedings of AUVSI Unmanned Systems 2005*, 2005.

[68] J. Hudson, J. Denzinger, H. Kasinger, and B. Bauer. Efficiency Testing of Self-Adapting Systems by Learning of Event Sequences. In *Proceedings of the 2nd International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE 2010)*, pages 200-205, IARIA, 2010.

[69] IBM. An Architectural Blueprint for Autonomic Computing. IBM, 2006. http://www-03.ibm.com/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf.

[70] ITEA. Agenda In *Proceedings of the Developing And Testing Autonomy (DATA) Workshop*, International Test and Evaluation Association (ITEA), 2010.

[71] M. A. Janssen, Editor. *Complexity and Ecosystem Management: The Theory and Practice of Multi-Agent Systems*. Cheltenham, UK, Edward Elgar, 2002.

[72] H. Kasinger, B. Bauer, and J. Denzinger. Design Pattern for Self-Organizing Emergent Systems Based on Digital Infochemicals. In *Proceedings of EASe 2009*, pages 45-55, 2009.

[73] J. O. Kephart and D. M. Chase. The Vision of Autonomic Computing. *Computer*, vol. 36, pages 41-50, 2003. Available at http://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf.

[74] J. O. Kephart, T. Hogg, and B. A. Huberman. Dynamics of Computational Ecosystems. *Physics Review*, 40A:404-421, 1989.

[75] A. Koestler. *The Ghost in the Machine*. 1967.

[76] P. Kuntz and P. Layzell. An Ant Clustering Algorithm Applied to Partitioning in VLSI Technology. In *Proceedings of Fourth European Conference on Artificial Life*, pages 417-424, MIT Press, 1997.

[77] C. Larman and V. Basili. Iterative and Incremental Development: A Brief History. *IEEE Computer*, pages 2-11, 2003. Available at http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf.

[78] M. Lejter and T. Dean. A Framework for the Development of Multiagent Architectures. *IEEE Expert*, 11(December):47-59, 1996.

[79] K. Lerman, A. Martinoli, and A. Galstyan. A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In S. E. and S. W., Editors, *Swarm Robotics Workshop: State-of-the-art Survey*, pages 143-152. Springer-Verlag, Berlin, Germany, 2005.

[80] M. Mamei and F. Zambonelli. *Field-based coordination for pervasive multiagent systems*. Springer, 2008.

[81] S. A. a. M. Masoud, A. A. Constrained Motion Control Using Vector Potential Fields. *IEEE Trans. on Systems, Man, and Cybernetics*, 30(3):251-272, 2000.

[82] J. C. Maxwell. On Governors. *Proceedings of the Royal Society of London*, 16:270–283, 1867.

[83] D. Merkle, M. Middendorf, and A. Scheidler. Swarm Controlled Emergence - Designing an Anti-Clustering Ant System. In *Proceedings of IEEE Swarm Intelligence Symposium*, pages 242-249, 2007.

[84] N. Monmarché. On data clustering with artificial ants. In *Proceedings of AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, pages 23-26, 1999.

[85] R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics*. Thesis at MIT, 2001.

[86] M. E. J. Newman. *Networks: An Introduction*. Oxford, UK, Oxford University Press, 2010.

[87] O. Nierstraz, M. Denker, and L. Renggli. Model-Centric, Context-Aware Software Adaptation. In B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Editors, *Software Engineering for Self-Adaptive Systems*, vol. 5525, pages 128-145. Springer, Heidelberg, 2009.

[88] J. J. Odell, H. V. D. Parunak, S. Brueckner, and J. Sauter. Temporal Aspects of Dynamic Role Assignment. In *Proceedings of Workshop on Agent-Oriented Software Engineering (AOSE03) at AAMAS03*, pages 201-213, Springer, 2003.

[89] OMG. Software & Systems Process Engineering Meta-Model Specification. Object Management Group, 2008. http://www.omg.org/spec/SPEM/2.0/PDF.

[90] A. Omicini. Towards a notion of agent coordination context. In D. Marinescu and C. Lee, Editors, *Process Coordination and Ubiquitous Computing*, pages 187–200. CRC Press, 2002.

[91] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini. Coordination Artifacts: Environment-based Coordination for Intelligent Agents. In *Proceedings of 3rd*

international Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), pages 286-293, ACM, 2004.

[92] A. Omicini and F. Zambonelli. Coordination for Internet application development. *Autonomous Agents and Multi-Agent Systems*, 23(3):251-269, 1999.

[93] H. V. D. Parunak. 'Go to the Ant': Engineering Principles from Natural Agent Systems. *Annals of Operations Research*, 75:69-101, 1997.

[94] H. V. D. Parunak. From Chaos to Commerce: Practical Issues and Research Opportunities in the Nonlinear Dynamics of Decentralized Manufacturing Systems. In *Proceedings of Second International Workshop on Intelligent Manufacturing Systems*, pages k15-k25, Katholieke Universiteit Leuven, 1999.

[95] H. V. D. Parunak. A Survey of Environments and Mechanisms for Human-Human Stigmergy. In D. Weyns, F. Michel, and H. V. D. Parunak, Editors, *Proceedings of E4MAS 2005*, vol. LNAI 3830, *Lecture Notes on AI*, pages 163-186. Springer, 2006.

[96] H. V. D. Parunak. Real-Time Agent Characterization and Prediction. In *Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'07), Industrial Track*, pages 1421-1428, ACM, 2007.

[97] H. V. D. Parunak. Generation and Analysis of Multiple Futures with Swarming Agents. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2010)*, pages 1549-1550, IFAAMAS, 2010.

[98] H. V. D. Parunak, T. C. Belding, and S. A. Brueckner. Prediction Horizons in Agent Models. In *Proceedings of Engineering Environment-Mediated Multiagent Systems (Satellite Conference at ECCS 2007)*, pages 88-102, Springer, 2008.

[99] H. V. D. Parunak and S. Brueckner. Ant-Like Missionaries and Cannibals: Synthetic Pheromones for Distributed Motion Control. In *Proceedings of Fourth International Conference on Autonomous Agents (Agents 2000)*, pages 467-474, 2000.

[100] 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)*, pages 124-130, ACM, 2001.

[101] H. V. D. Parunak, S. Brueckner, and R. Savit. Universality in Multi-Agent Systems. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 930-937, ACM, 2004.

[102] H. V. D. Parunak, S. Brueckner, D. Weyns, T. Holvoet, and P. Valckenaers. E Pluribus Unum: Polyagent and Delegate MAS Architectures. In *Proceedings of Eighth International Workshop on Multi-Agent-Based Simulation (MABS07)*, pages 36-51, Springer, 2007.

[103] H. V. D. Parunak and S. A. Brueckner. Engineering Swarming Systems. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, Editors, *Methodologies and Software Engineering for Agent Systems*, pages 341-376. Kluwer, 2004.

[104] H. V. D. Parunak, R. Rohwer, T. C. Belding, and S. Brueckner. Dynamic Decentralized Any-Time Hierarchical Clustering. In *Proceedings of Proceedings of the Fourth International Workshop on Engineering Self-Organizing Systems (ESOA'06)*, Springer, 2006.

[105] H. V. D. Parunak, J. Sauter, and S. J. Clark. Toward the Specification and Design of Industrial Synthetic Ecosystems. In M. P. Singh, A. Rao, and M. J. Wooldridge, Editors,

Intelligent Agents IV: Agent Theories, Architectures, and Languages, *Lecture Notes in Artificial Intelligence 1365*, pages 45-59. Springer, Berlin, 1998.

[106] H. V. D. Parunak, A. C. Ward, M. Fleischer, and J. A. Sauter. The RAPPID Project: Symbiosis between Industrial Requirements and MAS Research. *Journal of Autonomous Agents and Multi-Agent Systems*, 2:2 (June):111-140, 1999.

[107] H. V. D. Parunak, A. C. Ward, and J. A. Sauter. The MarCon Algorithm: A Systematic Market Approach to Distributed Constraint Problems. *AI-EDAM: Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 13(3):217-234, 1999.

[108] D. Payton, Daily, M., Estowski, R., Howard, M., and Lee, C. Pheromone Robotics. *Journal Autonomous Robots*, 11(3):319-324, 2001.

[109] P. Peeters, H. Van Brussel, P. Valckenaers, J. Wyns, L. Bongaerts, M. Kollingbaum, and T. Heikkila. Pheromone based emergent shop floor control system for flexible flow shops *Artificial Intelligence in Engineering*, 15(4 (Oct)):343-352, 2001.

[110] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. New York, NY, Springer-Verlag, 1990.

[111] M. Puviani, G. Di Marzo Serugendo, R. Frei, and G. Cabri. A method fragments approach to methodologies for engineering self-organising systems. *ACM Transactions on Autonomous and Adaptive Systems*, forthcoming, 2011.

[112] H. G. Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. *Trans. Amer. Math. Soc.*, 74:358-366, 1953.

[113] S. Ricketts. Holonic Manufacturing Systems. 1996. Web

[114] M. Salehie and L. Tahvildari. Self-adaptive software: landscape and research challenges. *ACM Transactions on Autonomic and Autonomic Systems (TAAS)*, 4(2):1-42, 2009.

[115] SAPERE. EU Sapere Project (Self-Aware Pervasive Service Ecosystems). 2011. http://www.sapere-project.eu/.

[116] SASO. Self-Adaptive and Self-Organizing Systems. 2011. http://www.saso-conference.org/.

[117] 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)*, pages 434-440, ACM, 2002.

[118] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. A. Brueckner. Performance of Digital Pheromones for Swarming Vehicle Control. In *Proceedings of Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 903-910, ACM, 2005.

[119] J. A. Sauter, R. Matthews, H. V. D. Parunak, and S. A. Brueckner. Effectiveness of Digital Pheromones Controlling Swarming Vehicles in Military Scenarios. *Journal of Aerospace Computing, Information, and Communication*, 4(5):753-769, 2007.

[120] J. A. Sauter, R. S. Matthews, J. S. Robinson, J. Moody, and S. P. Riddle. Swarming Unmanned Air and Ground Systems for Surveillance and Base Protection. In *Proceedings of AIAA Infotech@Aerospace 2009 Conference*, AIAA, 2009.

[121] R. Savit, S. A. Brueckner, H. V. D. Parunak, and J. Sauter. Phase Structure of Resource Allocation Games. *Physics Letters A*, 311:359-364, 2002.

[122] I. Scholtes. *Harnessing Complex Structures and Collective Dynamics in Large Networked Computing Systems*. Thesis at University of Trier, 2010.

[123] I. Scholtes. Thoughts on Engineering Self-Organizing Systems. 2011.

[124] I. Scholtes, J. Botev, A. Höhfeld, H. Schloss, and M. Esch. Awareness-Driven Phase Transitions in Very Large Scale Distributed Systems. In *Proceedings of the Second IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO)*, IEEE, 2008.

[125] P. Sengers. Designing comprehensible agents. In *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1227–1232, Lawrence Erlbaum, 1999.

[126] H. A. Simon. *The Sciences of the Artificial*. Cambridge, MA, MIT Press, 1969.

[127] O. Simonin, F. Charpillet, O. Buffet, and A. Glad. Engineering Self-Organizing Systems. 2011.

[128] O. Simonin, F. Charpillet, and E. Thierry. Collective Construction of Numerical Potential Fields for the Foraging Problem. *ACM TAAS*, (forthcoming), 2011.

[129] A. Spicher and O. Michel. Declarative modeling of a neurulation-like process. *BioSystems*, 87:281–288, 2006.

[130] A. Tyrrell, G. Auer, and C. Bettstetter. Biologically Inspired Synchronization for Wireless Networks. In F. Dressler and I. Carreras, Editors, *Advances in Biologically Inspired Information Systems: Models, Methods, and Tools*, *Studies in Computational Intelligence*, pages 47-62. Springer, 2007.

[131] P. Valckenaers. Self-organizing systems with emergent behavior. 2011.

[132] P. Valckenaers, H. V. Brussel, K. Hadeli, O. Bochmann, B. S. Germain, and C. Zamfirescu. On the design of emergent systems: an investigation of integration and interoperability issues. *Engineering Applications of Artificial Intelligence*, 16:377-393, 2003.

[133] P. Valckenaers and H. Van Brussel. Holonic manufacturing execution systems *CIRP Annals of Manufacturing Technology*, 54(1):427-432, 2005.

[134] M. Viroli and M. Casadei. Biochemical Tuple Spaces for Self-Organising Coordination. In *Proceedings of Coordination Languages and Models*, pages 143-162, Springer, 2009.

[135] M. Viroli and A. Omicini. The "Self-organising Coordination" Paradigm in the Software Engineering of SOS. 2011.

[136] M. Viroli, A. Ricci, F. Zambonelli, T. Holvoet, and K. Schelfthout. Infrastructures for the environment of multiagent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 13, 2007.

[137] M. Viroli and F. Zambonelli. A Biochemical Approach to Adaptive Service Ecosystems. *Information Sciences*, 180(10):1876-1892, 2010.

[138] B. Walsham. *Simplified and Optimised Ant Sort for Complex Problems: Document Classification*. Thesis at Monash University, Department of School of Computer Science and Software Engineering, 2003.

[139] P. Wegner. Why Interaction is More Powerful than Algorithms. *Communications of the ACM*, 40(5 (May)):81-91, 1997.

[140] J. Werfel. *Anthills built to order: Automating construction with artificial swarms*. Thesis at MIT, Department of CSAIL, 2006.

[141] D. Weyns. *Architecture-based design of multi-agent systems*. Springer 2010.

[142] D. Weyns. Software Engineering for Self-Organizing Systems. 2011.

[143] D. Weyns, N. Boucke, and T. Holvoet. A Field-Based Versus a Protocol-Based Approach for Adaptive Task Assignment. *Journal on Autonomous Agents and Multi-Agent Systems*, 17(2):288-319, 2008.

[144] D. Weyns, S. Dustdar, H. Giese, K. Göschka, V. Grassi, J. Kramer, S. Malek, R. Mirandola, C. Prehofer, R. Schlichting, B. Schmerl, and J. Wuttke. Dagstuhl Group D: From Centralized to Decentralized Control in Self-Adaptation. Dagstuhl, 2010. Available at http://www.dagstuhl.de/Materials/Files/10/10431/10431.SWM9.Slides.ppt.

[145] D. Weyns, S. Malek, J. Andersson, and B. Schmerl. Call for Papers, Special Issue on "State of the Art in Self-Adaptive Software Systems," Journal of Systems and Software (JSS). 2011. Available at http://www.elsevierscitech.com/cfp/CFP-JSS-special-issue-2010.pdf.

[146] N. Wiener. *Cybernetics*. Cambridge, MA, MIT Press, 1948.

[147] T. D. Wolf and T. Holvoet. Towards a Methodology for Engineering Self-Organising Emergent Systems. In *Proceedings of the 2005 conference on Self-Organization and Autonomic Informatics*, pages 18-34, IOS Press, 2005.

[148] T. D. Wolf, T. Holvoet, and G. Samaey. Engineering Self-Organising Emergent Systems with Simulation-based Scientific Analysis. In *Proceedings of the Fourth International Workshop on Engineering Self-Organising Applications*, pages 146-160, 2005.