The Knowledge Engineering Review, Vol. 30:4, 419–434.
doi:10.1017/S0269888915000089

Software engineering for self-organizing systems

H. VAN DYKE PARUNAK[1] and SVEN A. BRUECKNER[2]

[1]*AxonConnected, 2322 Blue Stone Hills Drive, Suite 20, Harrisonburg, VA 22801;*
e-mail: van.parunak@axonconnected.com;
[2]*AxonAI, 2322 Blue Stone Hills Drive, Suite 20, Harrisonburg, VA 22801;*
e-mail: sven.brueckner@axonai.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.

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 (Scholtes, 2011).

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 find it useful to distinguish three kinds of systems: autonomous, self-adaptive, and self-organizing. These terms are used with a wide array of meanings. We offer the following definitions, not to challenge the usage of other researchers, nor to claim that these are generally accepted, but to clarify and bound our own discussion.

An *autonomous* system is one that senses and responds to its environment. In software, the concept can be traced back to Wiener's work on feedback control systems starting in the 1940s (Watson & Scheidt, 2005). When the agent paradigm began to crystallize in the 1980s and 1990s, the term became common in the title 'autonomous agent'. One widely quoted definition of 'autonomous agent' (Wooldridge & Jennings, 1995) explains: 'Agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state'.

Recently, the term *self-adaptive* has come into vogue. The name suggests a system that responds to change without intervention by its creator (thus the 'self'), but this characteristic is also true of the classic autonomous system. Definitions of 'self-adaptive' often distinguish it from autonomy by emphasizing the system's ability to respond to something besides the environment. Here are two examples.

One team defines self-adaptive this way: 'Self-adaptive systems are capable of dealing with a continuously changing environment **and emerging requirements** that may be unknown at design-time' (emphasis ours; Brun *et al.*, 2009). By this criterion, a system with a single hard-wired goal may be autonomous (adapting its behavior in the light of environmental changes in order to achieve or maintain the goal), but to be self-adaptive requires the ability to develop new goals over time, without intervention of the designer. Architectures that meet this criterion usually have some hierarchical structure of goals, and

the adaptivity happens at lower levels. For example, most biological systems have a fairly immutable goal of procreation, and their self-adaptivity consists in refining their immediate plans in an effort to achieve this higher-level goal. In engineered systems, perhaps the most common form of this structure is the widely-used BDI (Belief-Desire-Intention) agent architecture (Rao & Georgeff, 1991; Haddadi & Sundermeyer, 1996; Kinny *et al.*, 1996), in which agents select their intentions (changeable, immediate objectives, on which they act) from among their desires (persistent objectives). In the nature of the case, the system cannot change its top-level goal by itself. If it could, it would need some criterion to guide that adaptation, and that criterion would then become the higher-level goal. Any system's ability to handle 'emerging requirements that may be unknown at design-time' is limited by the immutable top-level goal, which is known at design-time. Ultimately, the system is constrained by its designer, after all.

Another team applies the term 'self-adaptive' to 'systems that are able to adjust their behavior in response to their perception of the environment **and the system itself**' (emphasis ours; Cheng *et al.*, 2009). This definition requires that the system be self-aware. The definition of 'self-awareness' is itself slippery, and the topic is the focus of a major EU research program (Egan, 2011). Human self-awareness is recursive: we not only attend to our own state, but we are aware that we attend to our own state, and we are aware that we attend to our own state, and so forth. We know of no present-day engineered system that aspires to this kind of recursive awareness. Much work on self-adaptive software requires the system to have an internal representation of its goals (Heaven *et al.*, 2009), or a model of its own architecture (Nierstraz *et al.*, 2009), or a set of explicit policies (in case of X, do Y; Georgas & Taylor, 2009; Di Marzo Serugendo *et al.*, 2010), to guide its adaptation. The system's awareness is limited to an explicit representation that its designer has given it, and thus the 'self' nature of its awareness is compromised.

In sum, an autonomous system senses and responds to its environment without human intervention. Such systems are ubiquitous. Systems that claim to be self-adaptive sense and respond to something in addition to their environment, such as changing requirements or their own state, but in the examples that currently exist, this response is built in by the designer, qualifying the emphasis on the term 'self'. Practically speaking, when we call a system *self-adaptive*, we imply that the changes with which the system must cope are unusually large and potentially disruptive, the kind of change that one might expect would ordinarily require redesign by a human. If we build such a system, we are claiming that it can, by itself, perform adaptations that in a previous technology generation required human intervention.

The distinction between autonomous and self-adaptive is thus a moving target. Consider, by way of illustration, the evolution of the automobile. The earliest automobiles required the human to manage all adaptation with the environment, including setting the spark advance, controlling the mixture of fuel and air, adjusting the throttle to maintain speed as the slope of the terrain changes, and adjusting speed to accommodate other vehicles. One by one, these variables have become autonomous, in the order listed (with the successive invention of the vacuum spark advance, the automatic choke, speed control, and radar sensing and braking systems). A driver accustomed (say) only to automatic spark advance would consider a car that could manage its own fuel–air mixture and control its own velocity to be self-adaptive. Modern drivers think of such a vehicle simply as autonomous. In the domain of vehicle technology, 'self-adaptive' now implies capabilities that are on the cutting edge of research, such as platooning, obstacle recognition and avoidance, and self-routing. A decade from now, these will be considered simply autonomous.

The *self-organizing* system is a special kind of autonomous system. We emphasize three 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 autonomous or 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 (Parunak & Brueckner, 2001; Gershenson & Heylighen, 2003).

Second, we are particularly interested in systems whose response to change does not require centralized reflection. As we have discussed, the awareness definition of a self-adaptive system is usually realized by explicit internal representations of aspects of the system, such as goals (Heaven *et al.*, 2009), architecture (Nierstraz *et al.*, 2009), or policies (Georgas & Taylor, 2009; Di Marzo Serugendo *et al.*, 2010). 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 autonomous rather than self-organizing.

Third, to be self-organizing, this decentralized collection of entities must increase or maintain its degree of organization in the face of change. With other researchers, we measure organization in terms of the entropy of the population of elements (Parunak & Brueckner, 2001; Gershenson & Heylighen, 2003). A self-organizing system is one that counters the natural increase of entropy among its members over time.

Our focus in this paper is on self-organizing systems, not just autonomous or self-adaptive ones. While this distinction is important and useful (Merkle *et al.*, 2007), 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 (Di Marzo Serugendo *et al.*, 2010; Weyns *et al.*, 2011).

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 19th century (Maxwell, 1867), and it was natural for the idea to be applied to computer programs, largely under the inspiration of Norbert Wiener (1948). The flavor of adaptive control in robotic and manufacturing systems was captured in NIST's Real-time Control System reference architecture (Albus (1992, 1997)). Later, IBM's Autonomic Computing Initiative (Kephart & Chase, 2003; IBM, 2006) 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 (Huang *et al.*, 2005) and develop methods to test a system's autonomy (ITEA, 2010).

Self-adaptive software has been the object of two recent seminars at Schloss Dagstuhl (Cheng *et al.*, 2009; de Lemos *et al.*, 2012), and a special issue of the Journal of Systems and Software is in preparation on this topic (Weyns, 2011). The topic is the object of a careful review article (Salehie & Tahvildari, 2009), 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 (Di Marzo Serugendo *et al.*, 2004; Brueckner *et al.*, 2005, 2006, 2007). It is treated in Gershenson's (2007) recent dissertation, 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 (SASO)together are the focus of the ongoing IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO, 2011)[1].

[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.

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 (Bonabeau *et al.*, 1998; Heusse *et al.*, 1998; Holzer *et al.*, 2008; Scholtes, 2011) and water distribution (Dötsch *et al.*, 2010), and highly desirable in managing large numbers of robots (Sauter *et al.*, 2005, 2009; Glad *et al.*, 2008, 2009, 2010) and in agile manufacturing settings (Brueckner, 2000; Peeters *et al.*, 2001; Valckenaers *et al.*, 2003), where it competes with hierarchical control systems, including holonic schemes (Van Brussel *et al.*, 1998; Valckenaers & Van Brussel, 2005) that we would consider self-adaptive but not self-organizing. In purely informational settings, self-organization has been used to coordinate multiple theorem provers (Denzinger & Fuchs, 1999), to enable documents to organize themselves (Parunak *et al.*, 2006) and find likely users (Brueckner *et al.*, 2008), and to reassign tasks among agents (Odell *et al.*, 2003; Cicirello & Smith, 2004). Mechanisms inspired by wasps and termites have been demonstrated for self-organized construction of physical systems (Werfel, 2006).

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 (Parunak, 1997; Bonabeau *et al.*, 1999; Camazine *et al.*, 2001). These derive mostly from social animals: pheromone systems (Payton *et al.*, 2001; Peeters *et al.*, 2001; Sauter *et al.*, 2007; Kasinger *et al.*, 2009; Viroli & Casadei, 2009; Viroli & Zambonelli, 2010), stimulus-based load balancing (Werfel, 2006), insect clustering (Kuntz & Layzell, 1997; Monmarché, 1999; Handl *et al.*, 2003; Walsham, 2003; Hamdi *et al.*, 2010), firefly synchronization (Tyrrell *et al.*, 2007). But markets (Clearwater, 1996; Parunak *et al.*, 1999) and physical systems such as potential fields (Masoud & Masoud, 2000; Flacher & Sigaud 2002; Mamei & Zambonelli, 2005; Weyns *et al.*, 2008; Simonin *et al.*, 2011) 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 (Weyns, 2011). 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. These topics are empirical, not analytic: they represent our subjective impression of dominant themes, and we do not claim that they constitute a formal basis that spans the space of self-organizing systems. These characteristics reflect our definition of 'self-organization', and 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 unexpected, but we can gain understanding by analyzing it more closely.

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 (Albus, 1992). Holonic systems were originally motivated by emergent dynamics over a hierarchy that defines scale rather than control (Koestler, 1967), but engineered holonic applications often look very much like hierarchical control (Ricketts, 1996; Bongaerts, 1998; Van Brussel *et al.*, 1998). 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. (In terms of the BDI agent architecture, these kinds of information are the agent's beliefs and intentions, respectively.)

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, or beliefs), and it can only act within that narrow purview (imperative information, or intentions).

We have defined the endpoints in terms of local vs. global interaction, involving two kinds of information (declarative and imperative). Two other, intermediate combinations of these categories are possible.

It is quite common to collect the state of the system (declarative information) centrally, then make it available to all components, which take action locally. This approach, exemplified by the blackboard architecture (Nii, 1986a, 1986b) and its derivative mediator architecture (Shen & Norrie, 1997), is efficient for small systems, but encounters scaling problems as the size of the system increases.

In principle, one could imagine an inverse structure, choosing actions centrally and forcing agents to execute them, without the benefit of central knowledge of the system state. Because of the scaling problems of maintaining a complete, timely, centralized snapshot of the full system state, attempts to centralize both state and action often approximate this scheme. Because the centralized action decisions cannot take into account the local realities of the environment, the outcome of those actions is often disappointing.

In some cases, one can detect movement along this cline. For example, Denzinger *et al*. (1997) began working with multiple interacting theorem provers in a centralized setting, but then revised the system to use global information but only local decisions (Denzinger & Fuchs, 1999). In recent work in other domains, their design has moved to a distribution of both declarative and imperative information (Dötsch *et al.*, 2010). A motivation for this movement is the increasing need for real-time response (Denzinger *et al.*, 2011), which can be hindered if multiple layers of hierarchy need to be queried to make a decision (Weyns *et al.*, 2012).

In some domains, yet another approach to the question of centralization and decentralization is possible[2]. Sometimes there is a (domain-specific) relationship between (declarative) beliefs and (imperative) decisions. That is, certain information may be relevant only to certain decisions, and vice-versa. If the partitioning of information lends itself to a topological structure within which neighbor relations may be defined, one can localize both kinds of information, so that each process maintains state and makes decisions within its specialty, while interacting with neighbors when necessary. (If interaction with other belief-decision clusters is never necessary, attributing such a system to a single domain is of questionable value).

Complete localization of interaction does have a weakness: it limits look-ahead. 'It only functions acceptably when the (recent) past is representative for the (near) future' (Valckenaers, 2011). Predictive mechanisms have been proposed to address this problem (Cheng *et al.*, 2009). A particularly interesting approach 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 (Parunak *et al.*, 2007; Holvoet *et al.*, 2010). Mathematically, this approach is captured by the Monte Carlo tree search approach to learning Markov transition probabilities (Kearns *et al.*, 1999; Kocsis & Szepesvári, 2006).

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 (Clearwater, 1996), thus embodying our midpoint of global declarative information and local imperative information.

[2] We are grateful to an anonymous reviewer for this insight.

However, an alternative form of market, Edgeworth barter (Axtell & Epstein, 1997), 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 (Parunak *et al.*, 1999).

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. These candidates for interaction now include not only other elements that are like itself, but also technical, geographic, political, social and economic realities (Di Marzo Serugendo, 2009; Scholtes, 2011). 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' (Brooks, 1991). In the natural world, animals sometimes use the physical world to hold arbitrary markers (for instance, insect pheromones), which is one form of stigmergy (Grassé, 1959)[3]. Disembodied agents require a computational framework to hold such markers, such as a smart sensor network or RFID chips, and a major line of research (Mamei & Zambonelli, 2005; aliCE, 2008; Viroli & Omicini, 2011) is focused on designing such frameworks and their component mechanisms (Omicini & Zambonelli, 1999; Omicini, 2002; Omicini *et al.*, 2004; Viroli *et al.*, 2007).

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 (Valckenaers, 2011). 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 (Lejter & Dean, 1996; Casadei *et al.*, 2009) and highly structured symbolic 'pheromones' (Peeters *et al.*, 2001).

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. Thus, engineering of self-organizing systems needs to draw extensively on work on cyber-security and trust (T3 Group, 2012).

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 set of rich linguistic mechanisms that humans use to coordinate with one another. Computational mechanisms modeled on human consciousness and linguistic interaction are the

[3] In the other major form, sematectonic stigmergy, agents coordinate, not through arbitrary markers, but through functional changes to the structures that are the object of their coordination.

holy grail of AI research, but still beyond our grasp. Artificial versions of cognition have been described as autistic (Valckenaers, 2011) and schizophrenic (Sengers, 1999; Höning, 2011), 'idiot savants' with focused capability but lacking adaptability. This realization may lie behind the preference for simpler insect models in self-organizing software (Valckenaers, 2011), though in fact humans often use the same kind of simple mechanisms that insects do (Parunak, 2006).

3.4 Simulation

Simulation, rather than formal analysis, plays a prominent role in the engineering of most current self-organizing systems (De Wolf *et al.*, 2005). The complexity of these systems makes the development of formal models difficult (Höning, 2011). In fact, a set of even very simple agents interacting with one another has the computational power of a Turing machine (Edmonds & Bryson, 2004), or perhaps even more (Wegner, 1997), and by Rice's theorem (Rice, 1953), 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 (Epstein, 2006) such as that in the previous paragraph is invalid. The distinction we are making is not one of formal structure, but of insight. A computer program such as a simulation of a self-organizing system, 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[4]. 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. In general, such characterizations are unattainable (Edmonds & Bryson, 2004). Thus, it is likely that the engineering of self-organizing software will continue to rely heavily on simulation. However, 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 (2010) 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' (Beal, 2011), 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 (Puviani *et al.*, 2011) is to decompose an approach into fragments using SPEM (Software & Systems Process Engineering Metamodel; OMG, 2008) 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 (De Wolf & Holvoet, 2007; Gardelli *et al.*, 2007; Kasinger *et al.*, 2009; Holvoet *et al.*, 2010) 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 (aliCE, 2008) and architectures (Weyns, 2010) are seeking to address this problem, but the need for 'predictable results' awaits advances in formal analysis (Section 4.3).

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

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 characteristic, identified by Simon as critical to artificial systems (Simon, 1969), also requires a depth of formal insight that is not yet at hand.

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 (Kephart *et al.*, 1989; Brueckner, 2000; Janssen, 2002; Viroli & Zambonelli, 2010; Beal, 2011; SAPERE, 2011). In natural ecosystems, individual organisms interact in different ways, including amensalism, competition, antagonism, commensalism, and competition. For example, in a mutualistic relation such as pollination, insects and plants benefit one another by exchanging products or services. Tools such as service-oriented architectures (Brazier *et al.*, 2009) can enable software entities to discover one another and dynamically compose themselves into larger systems on the basis of mutually beneficial interactions.

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 (Denzinger *et al.*, 2011) 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 (Di Marzo Serugendo, 2009).

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 (Hudson *et al.*, 2010). 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 (Brueckner & Parunak, 2003) or over signals passing between agents (Holzer *et al.*, 2008; Höning & La Poutre, 2010) 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 (Weyns, 2011). A system of local constraints with attributes defined over component interfaces (Georgiadis *et al.*, 2002) is one promising way forward. Another is to deploy a control swarm in parallel with the functioning swarm (Merkle *et al.*, 2007). 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 (Di Marzo Serugendo *et al.*, 2010).

An important current area of research that promises to contribute to system characterization and control is work on normative MAS (Boella *et al.*, 2007), particularly mechanisms for the emergence and maintenance of norms (Villatoro, 2011). Results in this area offer the prospect of a system that develops its own characterization that it can then report to its stakeholders.

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 to one another within a single level, and a temporal dimension (dynamics) that explores how the system develops through time. To see the distinction among these dimensions, consider a stream of automobiles traveling down a highway. The vertical dimension describes how the behaviors of the components in an individual automobile join to produce the behavior of the overall automobile, or (at the next level) how the interactions of individual automobiles yield system-level features such as net throughput of the traffic system. The horizontal dimension includes the rules of

the road that regulate the interactions of automobiles with each other. The temporal dimension describes changes in the system over time, including the transition from fluid flow to a traffic jam, or the periodic variations in traffic density depending on time of day.

4.3.1 Emergence

Perhaps the most widely recognized phenomenon in dealing with self-organizing systems is emergence (De Wolf & Holvoet, 2005), which we define (Parunak & Brueckner, 2004) as system-level behavior that is not explicitly specified in the individual components. Emergence is a feature of the vertical dimension, describing the relation between the behavior of components and that of the entire system. Abstracted from software, the problem has a long history, forming the central focus 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 (Parunak & Brueckner, 2001; Guerin & Kunkle, 2004; Holzer *et al.*, 2008; Scholtes, 2010), phase shifts (Savit *et al.*, 2002; Brueckner & Parunak, 2005; Scholtes *et al.*, 2008), master equations (Bonabeau, 2002; Lerman *et al.*, 2005), and universality (Parunak *et al.*, 2004) to multi-agent systems. There are further insights to be gained from this approach. For example, the renormalization group (Binney *et al.*, 1992) has the potential to illuminate discontinuities in the behavior of a self-organizing system. By considering the system as it approaches certain limits (e.g. low agent density and high number of agents, allowing the use of a gas model, as in Bachrach *et al.* (2010)), we can place bounds on system characteristics of interest, offering 'thermodynamic guarantees' (Scholtes, 2011) 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 (Sauter *et al.*, 2002; Booker, 2003). This approach requires a system architecture whose representation lends itself to such evolutionary search, one whose structure remains valid under genetic operators such as mutation and crossover.

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' (Valckenaers, 2011), 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 Dawkins (1976) can be viewed as a process for favoring a well-defined group of agents (those agents possessing the gene) as opposed to individuals.

4.3.2 Organization

The horizontal dimension explores how constraints on which agents can interact affect the behavior of the whole system. Patterns of interactions among agents at the same level are naturally represented as a graph, a class of mathematical object that is amenable to a variety of formal tools (Newman, 2010). For example, useful definitions of autonomy and emergence can be formalized in terms of entropy on signals over the edges in the interaction graph (Holzer *et al.*, 2008), and usability can be defined in terms of similar measures on edges connecting the system to users (Höning & La Poutre, 2010). It has been suggested (Scholtes, 2010) 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 (Scholtes, 2011).

4.3.3 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 (Parunak *et al.*, 2007; Cheng *et al.*, 2009; Holvoet *et al.*, 2010; Valckenaers, 2011), but the nonlinear nature of component interactions means that trajectories diverge over time, leading to a prediction horizon (Parunak *et al.*, 2008) 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 (Scholtes, 2011).

One approach to formalizing the temporal dimension is to define formal languages to specify system development (Beal, 2010a). At this point, one may legitimately invoke Epstein's (2006) insistence on the formal nature of any computer program, 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 (Doursat, 2006), term-rewriting systems modeling plant growth (Prusinkiewicz & Lindenmayer, 1990) and their generalization in MGS (Spicher & Michel, 2006), Coore's (1999) Growing Point Language for interconnect topologies, Nagpal's (2001) Origami Shape Language, Werfel's (2006) system for distributed adaptive structure generation, and Beal's (2010b) Proto system for spatial computing.

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 (Beal & Knight Jr, 2008; Beal, 2011). Another example is the adaptation of methods for safety engineering to increasing the robustness and dependability of self-organizing software (Di Marzo Serugendo, 2009).

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 (Weyns, 2010, 2011). Research on frameworks to provide interaction environments for components (aliCE, 2008; Viroli & Omicini, 2011) 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 (Valckenaers, 2011).

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 (Simonin *et al.*, 2011). The approach has been applied to a number of mechanisms, including market-based control (De Wolf & Holvoet, 2007), gradient fields (De Wolf & Holvoet 2007; Kasinger *et al.* 2009), predictive swarms (Parunak *et al.*, 2007; Weyns *et al.,* 2012), replication, collective sort, evaporation, aggregation, and diffusion (Gardelli *et al.*, 2007). 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 (OMG, 2008) to facilitate the isolation and integration of method fragments (Puviani *et al.*, 2011), 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 (Larman & Basili, 2003; Beal, 2011). 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 (Baresi *et al.*, 2010). The system must be specified in

terms of desired performance and means of incrementally correcting deficiencies (Beal, 2010b), leading to systems that grow and react rather than being constructed and controlled (Scholtes, 2010). 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' (Becker *et al.*, 2010) 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.

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 (alphabetically by last name: Bernhard Bauer, Jake Beal, Olivier Buffet, François Charpillet, Jörg Denzinger, Giovanna Di Marzo Serugendo, Regina Frei, Kurt Geihs, Arnaud Glad, Nicolas Höning, Holger Kasinger, Andrea Omicini, Ingo Scholtes, Olivier Simonin, Paul Valckenaers, Mirko Viroli, and Danny Weyns). The authors particularly appreciate the detailed reviews of the field that several respondents contributed (Beal, 2011; Denzinger *et al.*, 2011; Höning, 2011; Scholtes, 2011; Simonin *et al.*, 2011; Valckenaers, 2011; Viroli & Omicini, 2011; Weyns *et al.*, 2011). 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.

References

Albus, J. S. 1992. RCS: a reference model architecture for intelligent control. *IEEE Computer* **25**(5), 56–59.

Albus, J. S. 1997. The NIST real-time control system (RCS): an approach to intelligent systems research. *Journal of Experimental and Theoretical Artificial Intelligence* **9**(2–3), 157–174.

aliCE 2008. *aliCE (agents, languages and infrastructures for complexity engineering) Home*. `http://alice.unibo.it/xwiki/bin/view/aliCE/`.

Axtell, R. & Epstein, J. 1997. *Distributed Computation of Economic Equilibria via Bilateral Exchange*. Technical report, Brookings Institution.

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

Baresi, L., Bencomo, N., Cukic, B., Gorla, A., Inverardi, P., Nier-strasz, O., Park, S., Smith, D., Vogel, T., de Lemos, R. & Andersson, J. 2010. *Dagstuhl Group c: Process*. `http://www.dagstuhl.de/Materials/Files/10/10431/10431.SWM12.Slides.ppt`.

Beal, J. 2010a. Functional blueprints: an approach to modularity in grown systems. In *Proceedings of the Seventh International Conference on Swarm Intelligence (ANTS 2010)*.

Beal, J. 2010b. *Mit Proto*. `http://stpg.csail.mit.edu/proto.html`.

Beal, J. 2011. Software engineering for self-organizing systems. Personal Communication.

Beal, J. & Knight, T. F. Jr 2008. 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)*.

Becker, B., Karsai, G., Mankovskii, S., Muoller, H., Pezze, M., Schaofer, W., Sousa, J. P., Tahvildari, L., Tamura, G., Villegas, N. M. & Wong, K. 2010. *Dagstuhl Group a: Towards Practical Run-Time V&V (for self-adaptive systems)*. `http://www.dagstuhl.de/Materials/Files/10/10431/10431.SWM10.Slides.ppt`.

Binney, J. J., Dowrick, N. J., Fisher, A. J. & Newman, M. E. J. 1992. *The Theory of Critical Phenomena—An Introduction to the Renormalization Group*. Clarendon Press.

Boella, G., Torre, L. v. d. & Verhagen, H. 2007. *Dagstuhl Seminar Proceedings 07122: Normative Multi-agent Systems*. LZI Host.

Bonabeau, E. 2002. Agent-based modeling: methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences* **99**(Suppl 3), 7280–7287.

Bonabeau, E., Dorigo, M. & Theraulaz, G. 1999. *Swarm Intelligence: From Natural to Artificial Systems (SFI Studies in the Sciences of Complexity)*. Oxford University Press.

Bonabeau, E., Henaux, F., Guerin, S., Snyers, D., Kuntz, P. & Theraulaz, G. 1998. Routing in telecommunications networks with "smart" ant-like agents. In *Proceedings of the Second International Workshop on Intelligent Agents for Telecommunications Applications (IATA98)*, Lecture Notes in AI, **1437**, 60–71. Springer.

Bongaerts, L. 1998. *Integration of Scheduling and Control in Holonic Manufacturing Systems.* PhD thesis, PMA.

Booker, L. 2003. Learning tactics for swarming entities. In *Swarming: Network Enabled C4ISR*, Inbody, D., Chartier, C., DiPippa, D. & McDonald, B. (eds), 40–48. ASD C3I.

Brazier, F. M. T., Kephart, J. O., Parunak, H.V.D. & Huhns, M. N. 2009. Agents and service-oriented computing for autonomic computing: a research agenda. *IEEE Internet Computing* **13**, 82–87.

Brooks, R. A. 1991. Intelligence without representation. *Artificial Intelligence* **47**, 139–159.

Brueckner, S. 2000. *Return from the Ant: Synthetic Ecosystems for Manufacturing Control.* Dr.rer.nat.thesis.

Brueckner, S., Downs, E., Hilscher, R. & Yinger, A. 2008. Self-organizing integration of competing reasoners for information matching. In *ECOSOA Workshop at SASO 2008*.

Brueckner, S., Hassas, S., Jelasity, M. & Yamins, D. 2007. *Engineering Self-Organising Systems*. Lecture Notes in AI 3910. Springer.

Brueckner, S. & Parunak, H. V. D. 2003. Resource-aware exploration of emergent dynamics of simulated systems. In *Autonomous Agents and Multi-Agent Systems (AAMAS 2003)*, Rosenschein, J. S., Wooldridge, M., Sandholm, T. & Yokoo, M. (eds), ACM, 781–788.

Brueckner, S. & Parunak, H. V. D. 2005. Information-driven phase changes in multi-agent coordination. In *Workshop on Engineering Self-Organizing Systems (ESOA, at AAMAS 2005)*, Lecture Notes in AI 3464, Brueckner, S. A., Di Marzo Serugendo, G., Karageorgos, A. & Nagpal, R. (eds). Springer.

Brueckner, S. A., Di Marzo Serugendo, G. & Hales, D. 2006. *Engineering Self-Organising Systems.* Lecture Notes in AI 3910. Springer.

Brueckner, S. A., Di Marzo Serugendo, G., Karageorgos, A. & Nagpal, R. 2005. *Engineering Self-Organising Systems. Lecture Notes in Computer Science. Springer*.

Brun, Y., Di Marzo Serugendo, G., Gqacek, C., Giese, H., Kienle, H., Litoiu, M., Mller, H., Pezz, M. & Shaw, M. 2009. Engineering self-adaptive systems through feedback loops. In *Software Engineering for Self-Adaptive Systems*, Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P. & Magee, J. (eds), **5525** Springer, 128–145.

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

Casadei, M., Viroli, M. & Gardelli, L. 2009. On the collective sort problem for distributed tuple spaces. *Science of Computer Programming* **74**(9), 702–722.

Cheng, S.-W., Poladian, V. V., Garlan, D. & Schmerl, B. 2009. Improving architecture-based self-adaptation through resource prediction. In *Software Engineering for Self-Adaptive Systems*, Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P. & Magee, J. (eds), **5525**, Springer, 128–145.

Cicirello, V. A. & Smith, S. F. 2004. Wasp-like agents for distributed factory coordination. *Journal of Autonomous Agents and Multi-Agent Systems* **8**, 237–266.

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

Coore, D. 1999. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer.* PhD thesis.

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

de Lemos, R., Giese, H., Muoller, H. & Shaw, M. 2012. *Software Engineering for Self-Adaptive Systems II*, Lecture Notes in Computer Science **7475**. Springer.

De Wolf, T. & Holvoet, T. 2005. Towards a methodology for engineering self-organising emergent systems. In *The 2005 Conference on Self-Organization and Autonomic Informatics*, Czap, H., Unland, R., Branki, C. & Tianfield, H. (eds), 18–34. IOS Press.

De Wolf, T. & Holvoet, T. 2007. Design patterns for decentralised coordination in self-organising emergent systems. In *The Fourth International Workshop on Engineering Self-Organising Applications (ESOA) at AAMAS 2006,* Brueckner, S., Hassas, S., Jelasity, M. and Yamins, D. (eds), LNAI **4335**, 28–49. Springer.

De Wolf, T., Holvoet, T. & Samaey, G. 2005. Engineering self-organising emergent systems with simulation-based scientific analysis. In *The Fourth International Workshop on Engineering Self-Organising Applications,* 146–160.

Denzinger, J. & Fuchs, D. 1999. Cooperation of heterogeneous provers. In *The 16th International Joint Conference on Artificial Intelligence (IJCAI 1999),* Dean, T. (ed.), **1**, 10–15. Morgan Kaufmann.

Denzinger, J., Fuchs, M. & Fuchs, M. 1997. High performance ATP systems by combining several AI methods. In *IJCAI-97*, Pollack, M.E. (ed.), 102–107. Morgan Kaufmann.

Denzinger, J., Kasinger, H. & Bauer, B. 2011. Software engineering for self-organizing systems. Personal Communication.

Di Marzo Serugendo, G. 2009. Robustness and dependability of self-organising systems—a safety engineering perspective. In *The 11th International Symposium on Stabilization, Safety and Security of Distributed Systems (SSS 2009),* Guerraoui, R. and Petit, F. (eds), LNCS **5873**, 254–268. Springer.

Di Marzo Serugendo, G., Fitzgerald, J. & Romanovsky, A. 2010. Metaself—an architecture and development method for dependable self-* systems. In *The 25th Symposium on Applied Computing (SAC 2010).*

Di Marzo Serugendo, G., Karageorgos, A., Rana, O. F. & Zambonelli, F. 2004. *Engineering Self-Organising Systems,* Lecture Notes in AI 2977. Springer.

Dötsch, F., Denzinger, J., Kasinger, H. & Bauer, B. 2010. Decentralized real-time control of water distribution networks using self-organizing multi-agent systems. In *The 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010),* Gupta, I., Hassas, S. & Rolia, J. (eds), 223–232. IEEE.

Doursat, R. 2006. The growing canvas of biological development: multiscale pattern generation on an expanding lattice of gene regulatory networks. *InterJournal: Complex Systems*, `http://www.interjournal.org`.

Edmonds, B. & Bryson, J. J. 2004. The insufficiency of formal design methods: the necessity of an experimental approach for the understanding and control of complex MAS. In *The 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004),* Jennings, N. R., Tambe, M., Sierra, C. & Sonenberg, L. (eds), 938–945. IEEE.

Egan, C. 2011. Awareness: self-awareness in autonomic systems. `http://www.aware-project.eu/`.

Epstein, J. M. 2006. *Generative Social Science*, Princeton Studies in Complexity. Princeton University Press.

Flacher, F. & Sigaud, O. 2002. Spatial coordination through social potential fields and genetic algorithms. In *The Seventh International Conference on Simulation of Adaptive Behavior (From Animals to Animats)*, Hallam, B., Floreano, D., Hallam, J., Meyer, J.-A. & Hayes, G. (eds), MIT Press.

Gardelli, L., Viroli, M. & Omicini, A. 2007. Design patterns for self-organizing multiagent systems. In *The 5th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2007),* Hans-Dieter, B., Gabriela, L., Rineke, V., & Lszl Zsolt, V. (eds), LNCS **4696**, 123–132. Springer.

Georgas, J. C. & Taylor, R. N. 2009. Policy-based architectural adaptation management: robotics domain case studies. In *Software Engineering for Self-Adaptive Systems*, Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P. & Magee, J. (eds), **5525** Springer, 89–108.

Georgiadis, I., Magee, J. & Kramer, J. 2002. Self-organising software architectures for distributed systems. In *The First Workshop on Self-healing Systems (WOSS '02)*, Garlan, D., Kramer, J. & Wolf, A. (eds). ACM.

Gershenson, C. 2007. *Design and Control of Self-organizing Systems.* PhD thesis.

Gershenson, C. & Heylighen, F. 2003. When Can We Call a System Self-Organizing? `http://arxiv.org/pdf/nlin.AO/0303020`.

Glad, A., Buffet, O., Simonin, O. & Charpillet, F. 2009. Self-organization of patrolling-ant algorithms. In *The International Conference on Self-Adaptive and Self-Organizing Systems (SASO09),* 61–70.

Glad, A., Buffet, O., Simonin, O. & Charpillet, F. 2010. Influence of different execution models on patrolling ant behaviors: from agents to robots. In *The Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10),* 1173–1180.

Glad, A., Simonin, O., Buffet, O. & Charpillet, F. 2008. Theoretical study of ant-based algorithms for multi-agent patrolling. In *The Eighteenth European Conference on Artificial Intelligence (ECAI'08),* 626–630.

Grassé, P.-P. 1959. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux* **6**, 41–84.

Guerin, S. & Kunkle, D. 2004. Emergence of constraint in self-organizing systems. *Nonlinear Dynamics, Psychology, and Life Sciences* **8**(2), 131–146.

Haddadi, A. & Sundermeyer, K. 1996. Belief-desire-intention agent architectures. In *Foundations of Distributed Artificial Intelligence*, O'Hare, G. M. P. & Jennings, N. R. (eds). John Wiley, 169–185.

Hamdi, A., Antoine, V., Monmarché, N., Alimi, A. & Slimane, M. 2010. Artificial ants for automatic classification. In *Artificial Ants: From Collective Intelligence to Real-life Optimization and Beyond*, Monmarch, N., Guinand, F. & Siarry, P. (eds). John Wiley and Sons, 265–290.

Handl, J., Knowles, J. & Dorigo, M. 2003. Ant-Based Clustering: A Comparative Study of its Relative Performance with Respect to k-means, Average Link and 1d-som. Technical Report TR-IRIDIA-2003-24, IRIDIA.

Heaven, W., Sykes, D., Magee, J. & Kramer, J. 2009. A case study in goal-driven architectural adaptation. In *Software Engineering for Self-Adaptive Systems*, Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P. & Magee, J. (eds), **5525** Springer, 109–127.

Heusse, M., Guerin, S., Snyers, D. & Kuntz, P. 1998. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems* **1**, 234–257.

Holvoet, T., Weyns, D. & Valckenaers, P. 2010. Delegate MAS patterns for large-scale distributed coordination and control applications. In *EuroPlop*.

Holzer, R., de Meer, H. & Bettstetter, C. 2008. On autonomy and emergence in self-organizing systems. In *Intern. Workshop on Self-Organizing Systems (IWSOS)*, LNCS **5343**. Springer.

Höning, N. 2011. *Comments on Software Engineering for Self-Organizing Systems*. Personal Communication.

Höning, N. & La Poutre, H. 2010. Designing comprehensible self-organising systems. In *The Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2010)*, 233–242. IEEE Computer Society.

Huang, H.-M., Pavek, K., Novak, B., Albus, J. & Messina, E. 2005. A framework for autonomy levels for unmanned systems (ALFUS). In *AUVSI Unmanned Systems 2005*.

Hudson, J., Denzinger, J., Kasinger, H. & Bauer, B. 2010. Efficiency testing of self-adapting systems by learning of event sequences. In *The 2nd International Conference on Adaptive and Self-adaptive Systems and Applications (ADAPTIVE 2010),* 200–205. IARIA.

IBM 2006. An architectural blueprint for autonomic computing. Technical report, IBM.

ITEA 2010. Agenda. In *The Developing And Testing Autonomy (DATA) Workshop*. International Test and Evaluation Association (ITEA).

Janssen, M. A. 2002. *Complexity and Ecosystem Management: The Theory and Practice of Multi-Agent Systems*. Edward Elgar.

Kasinger, H., Bauer, B. & Denzinger, J. 2009. Design pattern for self-organizing emergent systems based on digital infochemicals. In *EASe 2009*, 45–55.

Kearns, M., Mansour, Y. & Ng, A. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *The Sixteenth International Joint Conference on Artificial Intelligence*, 1324–1331. Morgan Kaufmann.

Kephart, J. O. & Chase, D. M. 2003. The vision of autonomic computing. *Computer* **36**(1), 41–50.

Kephart, J. O., Hogg, T. & Huberman, B. A. 1989. Dynamics of computational ecosystems. *Physics Review* **40A**, 404–421.

Kinny, D., Georgeff, M. & Rao, A. 1996. A methodology and modelling technique for systems of BDI agents. In *Agents Breaking Away. 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, Walter Vande, V. & John, W. P. (eds), Lecture Notes in Artificial Intelligence **1038**, 56–71. Springer.

Kocsis, L. & Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *The EMCL 2006*, Fiirnkranz, J., Scheffer, T. & Spiliopoulou, M. (eds), LNCS **4212**. Springer, 282–293.

Koestler, A. 1967. *The Ghost in the Machine*. Penguin Group.

Kuntz, P. & Layzell, P. 1997. An ant clustering algorithm applied to partitioning in VLSI technology. In *Fourth European Conference on Artificial Life*, Husbands, P. and Harvey, I. (eds), 417–424. MIT Press.

Larman, C. & Basili, V. 2003. Iterative and incremental development: a brief history. *IEEE Computer* **36**(36), 2–11.

Lejter, M. & Dean, T. 1996. A framework for the development of multiagent architectures. *IEEE Expert* **11**, 47–59.

Lerman, K., Martinoli, A. & Galstyan, A. 2005. A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm Robotics Workshop: State-of-the-art Survey*, Sahin, E. & Spears, W. (eds). Springer-Verlag, 143–152.

Mamei, M. & Zambonelli, F. 2005. *Field-Based Coordination for Pervasive Multiagent Systems*. Springer.

Masoud, S. A. & Masoud, A. A. 2000. Constrained motion control using vector potential fields. *IEEE Trans. on Systems, Man, and Cybernetics* **30**(3), 251–272.

Maxwell, J. C. 1867. On governors. *Proceedings of the Royal Society of London* **16**, 270–283.

Merkle, D., Middendorf, M. & Scheidler, A. 2007. Swarm controlled emergence—designing an anti-clustering ant system. In *IEEE Swarm Intelligence Symposium*, 242–249.

Monmarché, N. 1999. On data clustering with artificial ants. In *AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, Freitas, A. A. (eds), 23–26. AAAI.

Nagpal, R. 2001. *Programmable Self-Assembly: Constructing Global Shape using Biologically-inspired Local Interactions and Origami Mathematics.* PhD thesis.

Newman, M. E. J. 2010. *Networks: An Introduction*. Oxford University Press.

Nierstraz, O., Denker, M. & Renggli, L. 2009. Model-centric, context-aware software adaptation. In *Software Engineering for Self-Adaptive Systems*, Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P. & Magee, J. (eds), **5525**, Springer, 128–145.

Nii, H. P. 1986a. Blackboard systems. *AI Magazine* **7**(3), 40–53.

Nii, H. P. 1986b. Blackboard systems. *AI Magazine* **7**(4), 82–107.

Odell, J. J., Van Dyke Parunak, H., Brueckner, S. & Sauter, J. 2003. Temporal aspects of dynamic role assignment. In *Workshop on Agent-Oriented Software Engineering (AOSE03) at AAMAS03*, LNAI **2935**, 201–213. Springer.

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

Omicini, A. 2002. Towards a notion of agent coordination context. In *Process Coordination and Ubiquitous Computing*, Marinescu, D. & Lee, C. (eds). CRC Press, 187–200.

Omicini, A., Ricci, A., Viroli, M., Castelfranchi, C. & Tummolini, L. 2004. Coordination artifacts: environment-based coordination for intelligent agents. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, Jennings, N. R., Tambe, M., Sierra, C. & Sonenberg, L. (eds), **1**, 286–293. ACM.

Omicini, A. & Zambonelli, F. 1999. Coordination for internet application development. *Autonomous Agents and Multi-Agent Systems* **23**(3), 251–269.

Parunak, H. V. D. 1997. 'go to the ant': engineering principles from natural agent systems. *Annals of Operations Research* **75**, 69–101.

Parunak, H. V. D. 2006. A survey of environments and mechanisms for human-human stigmergy. In *Proceedings of E4MAS 2005,* Weyns, D., Michel, F. & Van Dyke Parunak, H. (eds), LNAI **3830**, Lecture Notes on AI, 163–186. Springer.

Parunak, H. V. D., Belding, T. C. & Brueckner, S. A. 2008. Prediction horizons in agent models. In *Engineering Environment-Mediated Multiagent Systems (Satellite Conference at ECCS 2007)*, Weyns, D., Brueckner, S. & Demazeau, Y. (eds), LNCS **5049**, 88–102. Springer.

Parunak, H. V. D. & Brueckner, S. 2001. Entropy and self-organization in multi-agent systems. In *The Fifth International Conference on Autonomous Agents (Agents 2001),* André, E., Sen, S., Frasson, C. & Müller, J. P. (eds), 124–130. ACM.

Parunak, H. V. D., Brueckner, S. & Savit, R. 2004. Universality in multi-agent systems. In *Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, 930–937. ACM.

Parunak, H. V. D., Brueckner, S., Weyns, D., Holvoet, T. & Valckenaers, P. 2007. E pluribus unum: Polyagent and delegate mas architectures. In *Eighth International Workshop on Multi-Agent-Based Simulation (MABS07),* Lecture Notes in AI **5003**, 36–51. Springer.

Parunak, H. V. D. & Brueckner, S. A. 2004. Engineering swarming systems. In *Methodologies and Software Engineering for Agent Systems*, Bergenti, F., Gleizes, M.-P. & Zambonelli, F. (eds). Kluwer, 341–376.

Parunak, H. V. D., Rohwer, R., Belding, T. C. & Brueckner, S. 2006. Dynamic decentralized any-time hierarchical clustering. In *Proceedings of the Fourth International Workshop on Engineering Self-Organizing Systems (ESOA'06)*, LNAI **4335**. Springer.

Parunak, H. V. D., Ward, A. C., Fleischer, M. & Sauter, J. A. 1999. The RAPPID project: symbiosis between industrial requirements and mas research. *Journal ofAutonomous Agents and Multi-Agent Systems* **2**(2), 111–140.

Payton, D., Daily, M., Estowski, R., Howard, M. & Lee, C. 2001. Pheromone robotics. *Journal Autonomous Robots* **11**(3), 319–324.

Peeters, P., Van Brussel, H., Valckenaers, P., Wyns, J., Bongaerts, L., Kollingbaum, M. & Heikkila, T. 2001. Pheromone based emergent shop floor control system for flexible flow shops. *Artificial Intelligence in Engineering* **15**, 343–352.

Prusinkiewicz, P. & Lindenmayer, A. 1990. *The Algorithmic Beauty of Plants*. Springer.

Puviani, M., Di Marzo Serugendo, G., Frei, R. & Cabri, G. 2011. A method fragments approach to methodologies for engineering self-organising systems. *ACM Transactions on Autonomous and Adaptive Systems* **7**(3), 1–25.

Rao, A. S. & Georgeff, M. P. 1991. Modeling rational agents within a BDI architecture. In *International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, Allen, J., Fikes, F., & Sandwall, E. (eds), 473–484. Morgan Kaufman.

Rice, H. G. 1953. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society* **74**, 358–366.

Ricketts, S. 1996. Holonic manufacturing systems.

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

SAPERE 2011. Eu SAPERE Project (Self-Aware Pervasive Service Ecosystems). `http://www.sapere-project.eu/`.

SASO 2011. Self-adaptive and Self-Organizing Systems. `http://www.saso-conference.org/`.

Sauter, J. A., Matthews, R., Parunak, H. V. D. & Brueckner, S. 2002. Evolving adaptive pheromone path planning mechanisms. In *Autonomous Agents and Multi-Agent Systems (AAMAS02)*. ACM, 434–440.

Sauter, J. A., Matthews, R., Parunak, H. V. D. & Brueckner, S. A. 2005. Performance of digital pheromones for swarming vehicle control. In *Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems,* Pechoucek, M., Steiner, D. & Thompson, S. (eds), 903–910. ACM.

Sauter, J. A., Matthews, R., Parunak, H. V. D. & Brueckner, S. A. 2007. Effectiveness of digital pheromones controlling swarming vehicles in military scenarios. *Journal of Aerospace Computing, Information, and Communication* **4**(5), 753–769.

Sauter, J. A., Matthews, R. S., Robinson, J. S., Moody, J. & Riddle, S. P. 2009. Swarming unmanned air and ground systems for surveillance and base protection. In *AIAA Infotech@Aerospace 2009 Conference*. AIAA.

Savit, R., Brueckner, S. A., Parunak, H. V. D. & Sauter, J. 2002. Phase structure of resource allocation games. *Physics Letters A* **311**, 359–364.

Scholtes, I. 2010. *Harnessing Complex Structures and Collective Dynamics in Large Networked Computing Systems.* PhD thesis.

Scholtes, I. 2011. Thoughts on Engineering Self-Organizing Systems. Personal Communication.

Scholtes, I., Botev, J., Hohfeld, A., Schloss, H. & Esch, M. 2008. Awareness-driven phase transitions in very large scale distributed systems. In *The Second IEEE International Conferences on Self-Adaptive and Self-Organizing Systems (SASO)*, Brueckner, S. A., Robertson, P. & Bellur, U. (eds). IEEE.

Sengers, P. 1999. Designing comprehensible agents. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Dean, T. (ed.), 1227–1232. Lawrence Erlbaum.

Shen, W. & Norrie, D. 1997. Facilitators, mediators or autonomous agents. In *Second International Workshop on CSCW in Design*, Siriruchatapong, P., Zongkai, L. and Barthes, J. P. (eds), 119–124. International Academic Pubilshers, Beijing.

Simon, H. A. 1969. *The Sciences of the Artificial*. MIT Press.

Simonin, O., Charpillet, F., Buffet, O. & Glad, A. 2011. Engineering Self-Organizing Systems. Personal Communication.

Spicher, A. & Michel, O. 2006. Declarative modeling of a neurulation-like process. *BioSystems* **87**, 281–288.

T3 Group 2012. T3 Group: Trust Theory Technology. `http://t3.istc.cnr.it/trustwiki/index.php/Main_Page`.

Tyrrell, A., Auer, G. & Bettstetter, C. 2007. Biologically inspired synchronization for wireless networks. In *Advances in Biologically Inspired Information Systems: Models, Methods, and Tools*, Dressler, F. & Carreras, I. (eds), Studies in Computational Intelligence. Springer, 47–62.

Valckenaers, P. 2011. Self-Organizing Systems with Emergent Behavior. Personal Communication.

Valckenaers, P. & Van Brussel, H. 2005. Holonic manufacturing execution systems. *CIRP Annals of Manufacturing Technology* **54**(1), 427–432.

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

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. & Peeters, P. 1998. Reference architecture for holonic manufacturing systems: prosa. *Computers In Industry* **37**(3), 255–276.

Villatoro, D. 2011. *Social Norms for Self-policing Multi-agent Systems and Virtual Societies.* PhD thesis.

Viroli, M. & Casadei, M. 2009. Biochemical tuple spaces for self-organising coordination. In *Coordination Languages and Models*, Field, J. & Vasconcelos, V. T. (ed.), **5521**, Springer, 143–162.

Viroli, M. & Omicini, A. 2011. The "self-organising coordination" Paradigm in the Software Engineering of SOS. Technical report, Universita di Bologna.

Viroli, M., Ricci, A., Zambonelli, F., Holvoet, T. & Schelfthout, K. 2007. Infrastructures for the environment of multiagent systems. *Journal of Autonomous Agents and Multi-Agent Systems* **14**(1), 49–60.

Viroli, M. & Zambonelli, F. 2010. A biochemical approach to adaptive service ecosystems. *Information Sciences* **180**(10), 1876–1892.

Walsham, B. 2003. *Simplified and Optimised Ant Sort for Complex Problems: Document Classification.* Bachelor of Software Engineering thesis.

Watson, D. P. & Scheidt, D. H. 2005. Autonomous systems. *Johns Hopkins APL Technical Digest* **26**(4), 368–375.

Wegner, P. 1997. Why interaction is more powerful than algorithms. *Communications ofthe ACM* **40**, 81–91.

Werfel, J. 2006. *Anthills Built to Order: Automating Construction with Artificial Swarms.* PhD thesis.

Weyns, D 2010. *Architecture-Based Design of Multi-Agent Systems*. Springer.

Weyns, D. 2011. Software engineering for self-organizing systems. Personal Communication.

Weyns, D., Boucke, N. & Holvoet, T. 2008. A field-based versus a protocol-based approach for adaptive task assignment. *Journal on Autonomous Agents and Multi-Agent Systems* **17**(2), 288–319.

Weyns, D., Malek, S., Andersson, J. & Schmerl, B. 2011. Call for Papers, Special Issue on "state of the art in self-adaptive software systems", Journal of Systems and Software (jss). `http://www.elsevierscitech.com/cfp/CFP-JSS-special-issue-2010.pdf`.

Weyns, D., Schmerl, B., Grassi, V., Malek, S., Mirandola, R., Prehofer, C., Wuttke, J., Andersson, J., Giese, H. & Goschka, K. 2012. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*, de Lemos, R., Giese, H., Müller, H. & Shaw, M. (eds), LNCS **7475**, pp. 76–107. Springer.

Wiener, N. 1948. *Cybernetics*. MIT Press.

Wooldridge, M. & Jennings, N. R. 1995. Agent theories, architectures, and languages: a survey. In *Intelligent Agents*, Wooldridge, M. & Jennings, N. R. (eds). Springer, 1–22.