Spray Computers: Explorations in Self-Organization

Franco Zambonelli, Marie-Pierre Gleizes, Marco Mamei, Robert Tolksdorf

Spray computers: Explorations in self-organization

Franco Zambonelli[a,*], Marie-Pierre Gleizes[b], Marco Mamei[a], Robert Tolksdorf[c]

[a] *DISMI, Università di Modena e Reggio Emilia, Reggio Emilia, Italy*
[b] *IRIT, Université Paul Sabatier, Toulouse, France*
[c] *Institut für Informatik, Freie Universität Berlin, Berlin, Germany*

Abstract

We envision a future in which clouds of microcomputers can be sprayed in an environment to provide, by spontaneously networking with each other, an endlessly range of futuristic applications. However, beside the vision, spraying may also act as a powerful metaphor for a range of other scenarios that are already under formation, from ad hoc networks of embedded and mobile devices to worldwide distributed computing. After having detailed the different spray computers scenarios and their applications, this paper discusses the issues related to the design and development of spray computer applications, issues which call for novel autonomic approaches exploiting self-organization as first-class tools. Finally, this paper presents the key research efforts being taken in the area and attempts at defining a rough research agenda.

Keywords: Spray computers; Bottom-up software engineering; Self-organization

1. Introduction

With the MEMS revolution in full swing, micro-sensors are following manufacturing curves that are at least related to Moore's Law [36]. This trend, when combined with both

the push for low power communication and computation devices and for the ubiquitous provisioning of data and services, pave the way for the spray computer revolution.

It is not hard to envision a future in which networks of microcomputers will be literally sold as spray cans, to be sprayed in an environment or on specific artifacts to enrich them with functionalities that, as of today, may appear futuristic and visionary [29,51, 37]. The number of potential applications of the scenario is endless, ranging from smart and invisible clothes, intelligent interactive environments, self-assembly materials and self-repairing artifacts.

However, the vision of spray computers may also act as a powerful metaphor for a range of other scenarios that are already under formation. These include distributed applications in embedded, possibly mobile, ad hoc networks [23,33,35], as well as distributed service and data-oriented activities on the Internet [42,39]. In fact, besides the different physical scale of the components involved and of their interactions (from microcomputers interacting within networks extending across a few meters, to Internet hosts interacting at a worldwide scale), all of these types of spray computer networks raise similar challenges as far as development and deployment of applications is involved, calling for radically novel – "autonomic" [48,18] – approaches to distributed systems development and management.

On the one hand, to avoid the unaffordable efforts related to the placement, configuration, and maintenance of such systems, there is the need for approaches that enable deploying components without any a priori layout effort, and let components self-organize their application activities and self-retune their overall behavior depending on specific contingencies (e.g., localized faults and environmental changes) [18]. On the other hand, the autonomous and decentralized nature of the activities in such scenarios, together with the possibly unpredictable dynamics of the operating environments, is likely to make those systems exhibit spontaneously "emergent" behaviors – as recent observations in several types of decentralized networks (i.e., the Internet, the Web, as well as Gnutella) suggest. Therefore, there is also a need for methodologies to predict and control the emergence of such behaviors and, when possible, offensively exploit them for the achievement of complex distributed tasks [32,20].

This paper aims at exploring the above issues and is organized as follows. Section 2 details our vision about spray computers, starting from the micro-scale (i.e., literally spray-able computers), to the medium scale (smart artifacts and MANETs), up to the macro-scale (wide-area networks). Section 3 presents and discuss the major issues arising when exploiting self-organization for the design and development of applications for spray computers. Section 4 presents a survey of some representative research efforts being taken in this area, discussing their advantages and drawbacks. Section 5 concludes the paper by attempting to define a roadmap of research activities in the area of spray computers.

2. Spray computers and applications

The concept of spray computers will soon pervade ICT scenarios at every scale and at every level. In the following we will briefly survey our idea of future computer-based

systems from the micro-scale (literally spray computers), to the medium-scale (handheld and wearable computers) to the global scale (Internet and Web computing).

2.1. The micro scale

As proved in the context of the Smart Dust project at Berkeley [3,36], it is already possible to produce fully-fledged computer-based systems of a few mm^3, and even much smaller ones will be produced in the next few years. Such computers, which can be enriched with communication capabilities (radio or optical), local sensing (e.g., optical, thermal, or inertial) and local effecting (e.g., optical and mechanical) capabilities, are the basic ingredients of our vision of spray computers.

Spray computers, as we imagine them, are clouds of sub-millimeter-scale microcomputers, to be deployed in an environment or onto specific artifacts via a spraying or a painting process. Once deployed, such components will spontaneously network with each other and will coordinate their actions (i.e., local sensing and effecting) to provide specific "smart" functionalities. We imagine it will be possible, say in 2020, to go to the local store and there buy, for a few euros, a "pipe repairing" spray, made up of a cloud of MEMS microcomputers capable of navigating in a pipeline, recognizing the presence of holes, and self-assembling with each other so as perfectly repair the pipe. Similarly, we could imagine a spray to transform our everyday desk into an active one, capable of recognizing the positions and characteristics of objects placed on it and letting them meaningfully interact.

Another peculiar application we envision is the "spray of invisibility" (described in [51]): a spray of micro devices capable of receiving and re-transmitting light emissions in a directional way, and capable of interacting with each other via short-range wireless communications. When an object is covered by a layer of such spray, the emissions of the devices make external observers perceive exactly the same light configurations that they would have perceived if there were nothing in between. In fact sensors on the rear side of the object can receive such configurations and, via distributed coordination, can communicate them to emitters on the observer's side to be retransmitted. Other types of application one could envision include any type of self-assembly artifact [29,46], there included things like Terminator T-1000, the nano-swarms of Michael Crichton's novel "Prey" [10], and MEMS-based artificial immune systems and drugs [37].

Whatever the applications one envisions, the key characteristics that will distinguish spray computer applications from traditional distributed computing systems are not – as one could at first think – the scale at which processes take place. After all the fact that a process executes on a micro device rather than on a high-end computer does not change its basic nature. Instead, what we think strongly distinguish spray computers are the facts that:

- computational activities take place in a network whose structure derives from an almost random deployment process (as a spraying process is), and that is likely to change over time with unpredictable dynamics (due to environmental contingencies, failure of components, or simply mobility);
- the number of (hardware and, consequently, software) components involved in a distributed application is dramatically high and hardly controllable. It is neither possible to enforce a strict configuration of software components nor to control their behavior during execution at a fine-grained level.

Both the above characteristics – which are somewhat exhibited by current sensor networks technologies [12] – compulsorily call for execution models in which applications are made capable of self-configuring and self-tuning their activities in a spontaneous and unsupervised way, adapting to whatever network structure and surviving network dynamics. In other words, the spray computer vision calls for the achievement of the autonomic computing vision [18].

2.2. The medium scale

Spray computing, other than being a likely incoming technology, can also act as a suitable metaphor for describing the key characteristics of the emerging scenarios of ubiquitous and pervasive computing, as enabled by handheld, wearable, and embedded networked computing systems.

We already typically carry two or three computers (i.e., a cell phone, a laptop, and possibly a PDA). Also, our houses are already populated by a variety of microprocessor-based furniture (e.g. TVs, phones, etc.). However, at the moment, the networking capabilities of these computer-based systems are under-exploited. In contrast, very soon, the world around us will be densely populated by personal-area networks (e.g., the ensemble of Bluetooth-enabled interacting computer-based devices we could carry or we could find in our cars), local ad hoc networks of handheld computers (e.g., networks of interacting PDAs carried by a team that have to directly interact and coordinate with each other in an open space), and furniture networks (e.g., Web-enabled fridges and ovens able to interact with each other and effectively support our cooking activities in a coordinated way).

What we want to emphasize here is that the above types of networks, although being formed by different types of computer-based devices (let us say, medium-end computers) and at different physical scales than literally spray computers, share with them the same issues as far as the development and management of distributed applications is concerned. In fact:

- most of these networks will be wireless, with structures dynamically varying depending on the relatives positions of devices, all of which intrinsically mobile (the persons in an ad hoc network can move around in an environment and the position of home furniture can changed on needs) and characterized by the dynamic arrival dismissing of nodes (a PDA running out of power or new home furniture being bought);
- even if technically possible, it is simply not commercially and economically viable to consider deploying applications that would require explicit configuration and explicit tuning to meet the amorphous and dynamic nature of the networks in which applications will be expected to operate.

Also in these cases, new approaches are needed to develop applications as if they were to execute on a network of spray computers.

2.3. The global scale

Similar considerations can apply to the case of macro-scale networks made up of high-end computer systems, i.e., the Internet and the Web. In fact, the dramatic growth of these

networks and of the information and traffic to be managed over them, together with the increasing request for ubiquitous connectivity and the peculiar structures exhibited by such networks [1,40] – making them sorts of "spray computer" networks – have raised researchers' attention to the need of novel approaches to distributed systems management.

Traditional approaches to management, requiring human configuration efforts and supervision, fall short when the number of nodes in the network (or the number of interrelated services and links in the Web) grows in a fully decentralized way, and when the presence of the nodes in a network is of an intrinsically ephemeral nature, as it is the case of laptops and, with regard to the Web, of several non-commercial data and services.

In a number of application scenarios, the need to access data and services according to a variety of patterns and independently of the availability/location of specific servers and of the dynamics of the network have suggested the adoption of P2P approaches [39, 40]. In P2P computing, instead of promoting a strict control over the execution of software components and of their interactions (an almost impossible task given the dynamics of the scenario), the idea is to promote and support adaptive self-organization and maintenance of a structured network of logical relationships among components (i.e., an overlay network), to abstract from the physical "sprayed" nature of the actual network and survive events such as the arrival of new nodes or the dismissing of some nodes.

Overlay networks are currently the most widely investigated approach to distributed application development and management in worldwide computing, and are leveraging a variety of useful applications facilitating access to (and coordination over) a variety of worldwide distributed data and services. Although self-organizing overlay networks – as they are studied today – are not necessarily the only and best approach, the attention towards them is the body of evidence of the need – also in this scenario – for new self-organizing approaches to distributed application development.

As a final note, we emphasize that, although the micro, medium, and global scale currently represent almost separated worlds, this will not be the case in the near future. All the above systems will probably be in the near future part of a mega decentralized network, including traditional Internet nodes, smart computer-enriched objects and furniture, networks of embedded and dispersed micro-sensors. For instance, the IPv6 addressing scheme will make it possible to assign an Internet address to every cubic millimeter in the earth's surface [15], thus opening the possibility for each and every computer-based component to become part of a single worldwide network.

3. Programming spray computers

Programming a spray computer (i.e. developing applications to be deployed on a network of spray computers) means engineering a pre-specified, coherent and useful behavior from the cooperation of an immense number of unreliable parts interconnected in unknown, irregular, and time-varying ways. This translates into devising algorithms and control methodologies to let the sprayed computing devices self-organize their interaction patterns and their activities: devices have to start working together without the presence of any a priori global supervisor or centralized controller.

The basic low-level mechanisms upon which to rely to enable self-organization appear to be quite well-understood and are basically the same whatever the scale, whether that of sensor networks or that of wide-area P2P computing. Among others: dynamic discovery of potential communication partners and of available services via broadcasting [14]; localization and navigation in some sorts of spatial environment, whether physical (as in sensor networks) or computational (as in P2P systems) [52].

What is still missing is an assessed understanding of how to design, develop, and manage, self-organizing applications for these kinds of systems, leading to some general purpose methodologies and programming environments, the main conceptual difficulty being that, while spray computers enable a direct-engineered control only on components' local activities, a variety of diverse application goals have to be achieved at a global scale.

Identifying some general and abstract solution to enable the design and development – via a proper programming of self-organizing activities – of specific global application goals, would have a dramatic impact in all sketched scenarios (micro, medium and global scale). In this paper, without having solutions at hand, we try to identify some key directions to investigate. In particular, the remainder of this section sketches three different approaches, and supports the presentation with the help of two simple, yet representative, case studies.

3.1. Direct engineering of self-organization

Overview

Direct engineering approaches to self-organization basically aim at defining distributed algorithms that, starting from a few basic mechanisms (e.g., broadcast and localization), and exploiting local interactions and local computations, can provably lead a system (or parts of it) to a final coherent global state.

Unlike traditional distributed algorithms, self-organizing algorithms disregard micro-level issues such as ordering of events, process synchronization, and structure of the underlying networks (issues for which no possibility of control is assumed). Rather, they focus on the fact that the algorithm will eventually converge despite micro-level contingencies and that it will keep the system in the stable state despite perturbations (e.g., changes in the network structure).

Examples

A typical example of a direct engineering approach to self-organization is distributed self-localization. There, a number of randomly distributed particles can determine their geographical position starting from a few "beacon" particles, possibly self-determined via leader election and acting as a reference frame. By recursively applying a local triangulation mechanism to have each particle determine its position w.r.t. close particles, a global coherent positioning of all particles in the reference frame is eventually reached [27], even in a large network, and whatever the specific pattern according to which the process propagates in the whole network.

Self-localization algorithms turn out to be very useful in a variety of scenarios, and specifically in sensor networks [12]. In general, the main goal of a sensor network is to spatially coordinate the activities of the sensors to cooperatively achieve specific sensing

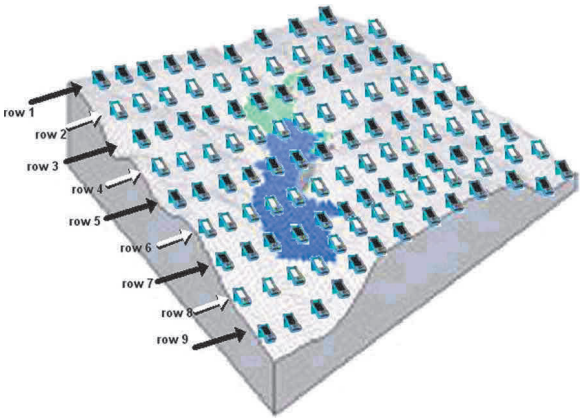

Fig. 1. Direct self-organization of activity patterns in a grid of sensors deployed on a landscape.

activities in an environment. Thus, in a sensor network, once each sensor has self-localized itself accordingly to some absolute or relative reference frame, sensors can easily self-organize their activities according to that. For example, they decide to alternatively get to a "sleepy" state according to some pre-configured geographical rule, i.e., in stripe-like patterns, so as to save battery power without undermining sensor coverage (Fig. 1). Being more visionary, a very similar approach can be exploited for the production of smart paints.

Another example that may be of use to sketch a different "direct" approach to self-organization relates to the problem of having a swarm of simple mobile robots be programmed to coordinate their respective movements and create a variety of global shapes. Apart from the vision of computational self-assembly [27,46], the problem also has more practical short-term applications: coordinating the movements of navigator-equipped cars, coordinating the movements of a rescue team provided with PDAs [23]; enforcing self-deployment of sensor networks and of complex robots in a landscape [25]. In particular, our goal is to devise distributed algorithms that, by locally driving the movements of the particles, eventually lead the system to self-organize in globally regular shapes.

Given a number of particles distributed in an environment, it is possible to devise distributed algorithms that, by locally driving the movements of the particles, eventually lead the system to self-organize in globally regular shapes [21]. For instance (see Fig. 2): a simple leader election algorithm can be exploited to have particles self-determine their center of gravity. Then, the center of gravity can propagate across the network a hop-increasing data structure (a sort of "gravitational field"), attracting all other particles toward the center, and until a specific distance from the center is reached. The result is an almost circular organization of particles.

Discussion

Direct engineering approaches to self-organization have the great advantage of enabling engineers to achieve "by design" a specific robust self-organized behavior. Engineers, following standard methodologies, can identify and design distributed algorithms to control distributed components and let them behave as specified by the application.

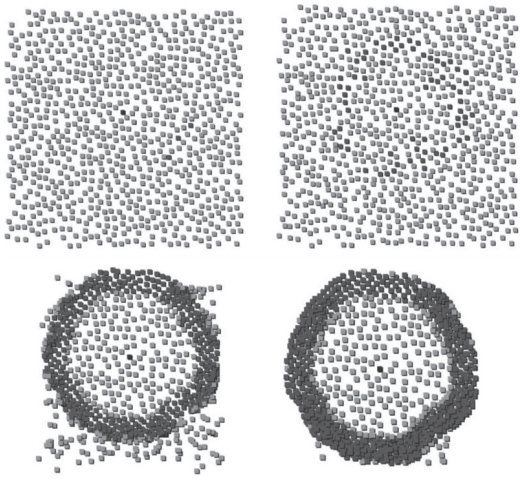

Fig. 2. Sequences of direct self-organization of a swarm of cooperative mobile robots into a circular shape.

Unfortunately, such approaches are effective only for a limited number of application needs. In fact, when the applications to be realized become more articulated and involve non-regular phenomena such as differentiation in activities, navigation and localization in complex manifolds, enforcement of complex coordination patterns, then the complexity of the distributed algorithms involved may become overwhelming.

Moreover, such approaches tend to require a priori assumptions on the configuration of the system that would limit its self-organizing nature and, consequently, its degree of adaptivity and its robustness.

With regard to the sensor network example, even the achievement of the simple stripe-like patterns in Fig. 1 requires the prior execution of the self-localization algorithm and some a priori agreement on geographical rules. With regard to the mobile particles example, making particles self-organize into complex non-symmetrical patterns by direct self-organization would require either a very complex algorithm to be executed by the particles, or having particles provided with some a priori information on where to go, that is, some a priori configuration efforts.

For all these cases, a reverse engineering approach may be preferred.

3.2. Reverse engineering of self-organization

Overview

Reverse engineering approaches to self-organization aims at achieving complex coordinated behaviors by recreating in spray computers (and by adapting to specific application needs) the conditions to make some emergent coordinated behaviors observed in other systems and in nature emerge in the computational spray computer system too. In these cases, due to the complexity (and often non-linearity) of the phenomena involved, engineers have no direct control on the evolution of the system, nor they can somehow prove that the system will behave as needed. Simply, they can be reasonably

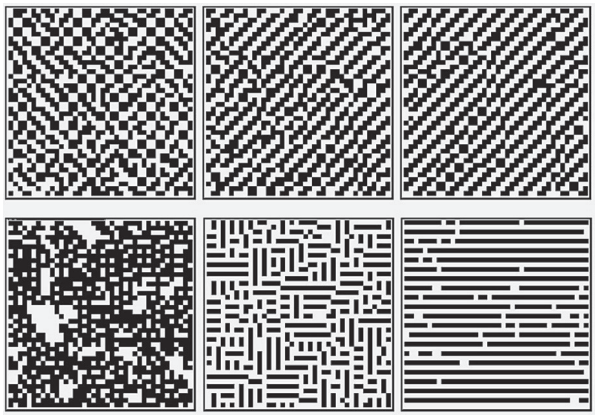

Fig. 3. From left to right, the evolution of two different types of stochastic cellular automata, evolving into macro-level spatial patterns of coordination.

(i.e., probabilistically) confident that the global evolution of the system will eventually lead to the desired globally coordinated behavior.

Clearly, simulations will be the workhorse of reverse engineering approaches [11]. Simulations of spray computer systems will not only provide a framework on which to test the functionalities of a systems once developed, but they will be an integral part of the design and development process. Since the behavior of the components and of their interactions that can lead to the desired global behavior can hardly be modeled and predicted "on paper", simulations appear to be the only tool with which to have feedback on how a system will actually work. In other words, in reverse engineering approaches, the modeling phase consists in verifying via simulations the correctness of an idealized model suitable for, but not necessarily close to, the target scenario (this model can be for example a biological or social model), then to refine the model and the simulations (that also realize a prototype implementation of the model) to render both sufficiently similar to the actual scenario to be taken into consideration as a candidate solution.

Examples

In the past few years, several approaches to self-organization relying on the reverse engineering of diverse natural phenomena have been proposed in different areas and have shown their effectiveness in achieving difficult global coordination tasks. For instance, the phenomena of ant foraging [2,26] and gossiping [6] turn out to be useful for discovering paths to information and diffuse information in networks of spray computers.

Getting back to our sensor network example, one can get inspiration to some behaviors exhibited by cellular automata. Cellular automata are a model for locally interacting cells, each with a local state (finite-state automata) that get updated on the basis of simple rules and accounting the state of neighbor particles in a grid [50]. Cellular automata, on the basis of simple rules and with a moderated injection of stochastic behavior, exhibit the emergence of macro-level spatial patterns of coordinated activity, e.g., among others, a variety of stripe-like patterns (Fig. 3). Reverse engineering and reproducing a similar

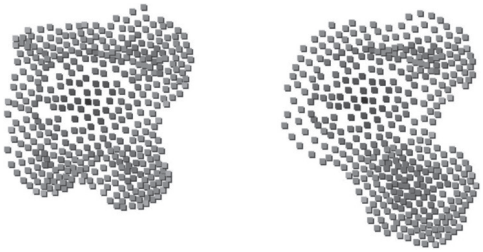

Fig. 4. Emergence of non-symmetrical patterns in swarms of simple cooperative mobile robots.

behavior in a grid of sensors (and, more generally, in any – even amorphous – network of sensor) is dramatically simple. Basically, this amounts to having the activity state of a sensor (i.e., sleepy versus active state) be decided on the basis of rules that account for the state of neighbor sensors, mimicking the behavior of cellular automata [20]. A variety of other types of distribution of activities can be analogously enforced, also by extending from cellular automata to more general phenomena of spatial differentiation of activities [44].

With regard to the example of the swarm of mobile robots, specific application needs may require a robot to assume a specific non-symmetric form. In particular, it is possible to enforce the emergence of non-symmetrical patterns by getting inspiration from the biological formation of morphogenesis. There, it can be observed that differentiation from regular symmetrical patterns occur due to the contrasting forces induced by cells increasing in number and still having to adhere to each other in a limited physical space. Similarly, in mobile computational particles, one can impose a constraint on the average density of cells, making cells repel each other if a too high density is reached. This constraint, contrasting with the gravitational force field that would attract them toward the center of gravity, forces them to organize in non-symmetrical patterns (see Fig. 4).

Discussion

Reverse engineering approaches to self-organization have several advantages. First, it is possible to rely on results from other disciplines to explore a variety of complex coordination phenomena to be exploited in spray computer systems. Second, once the basic mechanisms underlying a self-organized behavior are understood and properly reproduced via simulation, programming an actual system to exhibit such behavior is dramatically simple, and it reduces to programming typically simple local rules and local interactions. In addition, the resulting system is intrinsically robust and adaptive. The sensor network examples clearly show this: stripe-like patterns of activity can be obtained in a very simple way, without requiring the prior execution of a complex and costly self-localization mechanism, and without requiring any a priori agreement on geographical rules for selecting the sleepy/active state in sensors.

Unfortunately, reverse-engineering approaches may incur an important potential drawback. In fact, the evolution of a system relying on complex reverse engineered phenomena – typically involving non-linearity – may cause several potential final states to be reached by the system, each of which being potentially stable, and without the possibility of predicting which ones will be actually reached after the self-organization process.

In some cases, all of these states may be equivalent from the application viewpoint (e.g., in ant foraging, what matters is that a reasonably short path to food/information is reached, no matter what the path actually is). Also, in these cases, the presence of multi-stable states may be also advantageous, because this ensure that the system, even if strongly perturbed (e.g., due to network or environmental dynamics), will be able to soon re-organize its activity into another stable state. However, in several other cases, the designer may wish that its system self-organizes to a specific global state, not to any one.

For instance, in the case of the sensor network examples, when applying the cellular automata approach, one may wish that the network self-organize into "horizontal" rather than "vertical" stripes (cf. Fig. 3). In the case of cooperative mobile robots, specific application needs of self-assembly or of landscape may require the robots to assume a specific non-symmetric form, not any of the ones that could emerge from the self-organization process. For both cases, the same problem applies in the case of system perturbation: once an already self-organized system gets perturbed, the designer may not wish it to re-stabilize to a different state.

The above problems can be somewhat solved by trying to mix together direct and reverse engineering approaches.

3.3. Control of self-organization

Overview

When the evolution of a system can lead to several final global states, and only a limited set of these are useful to the specific application purposes, the problem arises of how to control/direct the spontaneous evolution of the system so as to ensure that it will self-organize as desired.

To enforce such control one can think of:

- using a *reverse engineering* approach to exploit a phenomenon – and the associated low-level mechanism – that can lead to a useful global behavior;
- coupling it with some *direct engineering* to somewhat control the evolution of the system global behavior.

In general, such a mixed approach presents a number of challenges. On the one hand, introducing some sorts of direct engineered control should be done without undermining the basic advantages of the reverse engineering approach, i.e., its capability to promote the spontaneous formation of complex and robust patterns of activity with little design and coding efforts. On the other hand, for such a mixed approach to be possible, it is necessary that both direct algorithms and reversed self-organizing mechanism are modeled and coded with the same set of basic abstractions.

Examples

We can still refer to the two already discussed examples to clarify what such a mixed approach for the control of emergent behaviors could be.

In the example of sensor networks, we have verified via cellular automata simulations [20] that it is indeed possible to control the evolution of the system so that its evolution will end in a specific global pattern – among several possible ones. This control can be achieved by having a limited percentage of the cells in the cellular automata (i.e., a limited

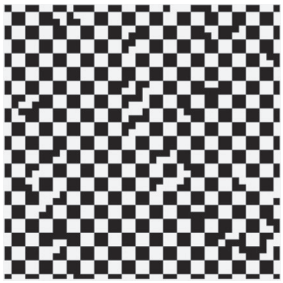

Fig. 5. Control of emergent behaviors in cellular automata.

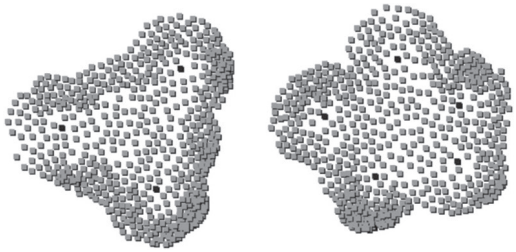

Fig. 6. Controlling the emergence of specific non-symmetrical patterns in swarms of mobile cooperative robots.

number of sensors in the network) modified so as to act on more constrained rules than those of the other cells. This limited percentage of cells is nevertheless able to globally influence the spontaneous evolution of the system so as to direct it towards a specific final configuration. With this methodology, we have also been able to have a system evolving towards configurations that would have been (probabilistically) very hard to achieve via spontaneous evolution only (Fig. 5).

In the example of the mobile robot swarm, we have successfully enforced some sort of distributed control of emergent behavior, by properly mixing the phenomena making non-symmetrical patterns emerge (described in the previous section) and sort of leader election algorithm. In particular, by using specific computational fields we have been able to let the system self-select a specific number of particles – equidistant from each other and from the center. These particles can then start to act as if they had a higher gravitational mass (i.e., diffusing a gravitational field of higher value), which makes it possible to control the emergence of polygon-like structures (see Fig. 6) with any required number of vertices. Overall, the solution preserves enough simplicity and robustness.

Discussion

Besides the above two examples, and a few additional ones described in the next section, the general problem of combining self-organizing emergent behaviors and direct engineering approaches to obtain a reliable, adaptive, robust self-organizing system is still open and widely uninvestigated. Nevertheless, in the authors' opinions, this will represent one of the key challenges for the whole research area of autonomic and self-organizing computing.

The urge for appropriate control models and for a uniform approach to direct and reverse engineering of self-organization appears even more compulsory when considering another factor: in several cases, even simple systems engineered with a direct approach may – due to simplifications in their modeling – exhibit unexpected self-organizing behaviors. Although sometimes such unexpected behaviors may be irrelevant, or it may even be useful for them to be offensively exploited (consider, for example, the emergence of scaling in complex networks, and the advantages it carries to the robustness of the network [1]), sometimes they may be damaging and may introduce the need to defend from them by proper forms of control.

4. Relevant research projects

Several projects around the world are starting to recognize the above needs and are facing issues related, to different extents, to the engineering and programming of spray computer systems and applications. Without the ambition to be exhaustive, we present here a few relevant threads of activities and discuss their shortcomings.

4.1. Micro scale

Most of the researches in the area of micro-scale spray computers (i.e., literally spray computers) are performed in the context of the "sensor networks" research community [12]. There, the key issues being investigated relate to the identification of effective algorithms and tools to perform distributed monitoring activities by a cloud of distributed sensors in a physical environment (tracing the position and movement of an object, determining the occurrence of specific conditions, reporting sensed data back in an efficient way). These researches are indeed providing good insights on the theme of self-organization and are leading to some very interesting results. Techniques for self-localization, self-synchronization of activities, adaptive data distribution, all of which of primary importance for any type of spray computers, have been widely investigated [12, 28]. Still, we feel these researches are somewhat limited by several main factors. First, most approaches rely on direct self-organization, while reverse engineering approaches are only occasionally exploited. Second, the accent on "sensing" tends to disregard the "actuating" factor – potential source of a wide range of interesting applications – and the algorithms and tools that could be of use to perform specific actuation works. Third, most research work is being devoted to the definition of "power-effective" algorithms, aimed at minimizing resource consumption. This is motivated by the current impossibility of providing such small computer systems with enough battery power to last for a long time. However, we think that short-life batteries and the consequent need of power-aware computing models are a current contingent problem, rather than basic research issues likely to have a long-term impact. Scavenging power from sunlight, vibration, thermal gradients, and background RF, the next generation of microcomputers will be fully autonomous in terms of power supply, and will be capable of long-lasting, if not ever-lasting, activity [37].

The Amorphous Computing project at MIT, more closely approaching our vision of spray computers, focuses on the problem of identifying suitable models for programming applications over amorphous networks of computational particles [29]. The particles

constituting an amorphous computer have the capabilities of locally propagating sorts of computational fields in the network, and to locally sense and react to such fields (the field abstraction has been reverse engineered from the biological concept of morphogen gradients). By having particles sense and re-propagate these fields, coordinated patterns of activities emerge in the system independently of the structure of the network. Moreover, the Amorphous Computing project has defined a simple yet effective language for programming particles on the basis of such computational fields. It has been shown how it is possible to exploit such a language to let the particles (directly) self-organize in a coordinate system and self-determine their position in it, and to have a variety of global patterns getting (directly) organized in a system from local interactions. Areas that the project has still not properly addressed concerns are the extensive reverse engineering exploitation of emergent phenomena and the study of networks with mobile and ephemeral particles.

Explicitly focused on mobility are those works in the area of cooperative mobile robotics [25,34,21,46], which also pay more attention to emergent phenomena. These works, as in the examples reported in the previous section, try to reverse engineer natural phenomena such as chemical diffusion and embryogenesis activities to drive the spatial assembling and the movements of a swarm of tiny mobile robots (particles). In these works mobile robots connect with each other in an ad hoc network to coordinate their movements. The main reversed abstractions being researched are those of chemical-gradient diffusion (that can be some somewhat assimilated to the computational fields exploited in the Amorphous Computing project) and local density estimation. These abstractions can be used by the robots to coordinate their movements. In particular, gradients can drive robots to reach specific locations [34,21]. Density can be used to create diffusion-like processes, where robots tend to stay as far as possible from each other [25]. Both these two abstractions can be reverse engineered in terms of distributed data structures (e.g., field-like hop-increasing structures and gradients of density pressures) over the ad hoc network defined by robots themselves. Despite some promising results, up to now, the cited works focused on very simple motion strategies and computational particles – not much different from cellular automata cells. Their effectiveness in coordinating systems of more complex particles and in more complex application scenarios is still to be verified.

Very similar considerations can apply to the area of motion gait enforcement in modular robots [43,38] and to the area of motion coordination for unmanned autonomous vehicles [31].

4.2. Medium scale

Coming to the medium scale, as far as we can see most of the researches are focusing either on routing algorithms for mobile ad hoc networks of handheld computers [7,19] or on the definition of effective user-level ubiquitous environments [41,49].

Researches on routing algorithms for mobile networks share several common issues with researches on algorithms for data distribution on sensor networks, and often are based on self-localization and spatial concepts [19], i.e., relying on direct engineering approaches. In our opinion, these works are, again, too often focused on power and resource

limitation problems and mostly disregard higher-level self-organization issues such as coordination of distributed behaviors.

Researches on ubiquitous computing environments mostly focus on achieving dynamic interoperability of existing application-level components and of smart-artifact and pervasive computing devices. For instance, the Gaia system developed at PARC [41] defines an architecture based on "active" interaction spaces, as a reification of a specific real-world environment (e.g., a meeting room), where pre-existing (and pre-programmed) devices and user-level software components can dynamically enter, leave and interoperate dynamically with each other according to specific patterns specified as part of the active environment. Although such an approach is very important for organizing user-level activities and their interactions with a smart environment, neither Gaia nor most of the other proposals in this direction has something to say on the issue of designing, developing, and controlling self-organizing coordinated distributed applications.

The specific current research area of the first and third authors goes exactly in the direction of extensively applying self-organization mechanisms and phenomena in the context of pervasive computing. The idea of computational fields (or morphogen gradients) has been applied in the implementation of the TOTA middleware [22]. TOTA relies on spatially distributed tuples, implementing the concept of computational field, for adaptive and context-aware interactions. Application components can inject these tuples in the network, to make available some kind of contextual information and to interact with other agents. Tuples are propagated by the middleware in the pervasive network, on the basis of application specific patterns, defining sorts of "computational fields", and their intended shape is maintained despite network dynamics, such as topological reconfigurations. Agents can locally "sense" these fields and can rely on them for both acquiring contextual information and carrying on distributed self-organizing coordination activities. The TOTA model is effective in realizing applications centered on "spatial" coordination, where the goal is to coordinate the activities of components in the space – either physical or network-based (e.g. motion coordination, self-assembly, network routing, etc.) – and can be effectively used in a general-purpose way to program and deploy applications exploiting both direct and reverse self-organization approaches. However, the generality of this approach in supporting very large-scale applications for highly dynamic networks, as well as in deploying mixed approaches for the control of self-organization, is still to be proved.

As an additional research direction, the idea of self-organizing pheromone-based coordination, as inspired by ant foraging [5,30], is being currently experienced by having the components of a pervasive environment deposit pheromones in the form of RF-ID tags [24].

4.3. Global scale

As far as the global scale is involved, we have already stated that most research on adaptive and unsupervised computing has focused, so far, on the key idea of defining structured self-organizing overlay networks for P2P computing. This can be considered as a typical example of a direct engineering approach to self-organization [39,42,8].

Most research work on structured P2P overlays is indeed very valuable and has provided several insights on the actual problems involved in self-organized computing

on a worldwide scale. However, this does not necessarily imply that structured overlay networks are the best approach to promote self-organizing global scale computing. In fact, building and maintaining globally coherent overlay networks on a worldwide scale may be very costly. Thus, despite the simulation on small-scale systems showing the feasibility of the approach, it is not very clear how this could scale to millions of nodes. In addition, although most of the proposals for overlay networks prove their effectiveness in re-organizing a coherent structure upon dynamic changes in the structure, such studies are typically performed by testing the sequential arrival/dismissing of single nodes, and it is not clear if a higher degree of networks dynamics (with concurrent arrivals/dismissing of nodes) would be sustained equally well. In our opinions, next generation P2P should better rely on more flexible and lightweight approaches, possibly exploiting reverse engineering approaches, or a sort of mixed approach.

In this context, some recent works in the area of Internet routing and distributed interactions root on reverse engineering ideas coming from biology – i.e. ant foraging [2,5, 6], or physics – i.e. potential fields [23]. As an example of this class of approaches (based on artificial ants), Anthill [2] supports the design and development of adaptive peer-to-peer applications by relying on distributed mobile components ("ants") that can travel and can indirectly interact and cooperate with each other by leaving and retrieving bunches of information (to act as synthetic pheromones) in the visited hosts. The key objective of Anthill is to build robust and adaptive "semi-structured" networks of peer-to-peer services by exploiting the capabilities of ants to re-organize their activity patterns according to the changes in the network structure. As another example, SwarmLinda [26] (developed by the fourth author) is an ant-inspired system enabling globally distributed application components to adaptively coordinate with each other. Application components on the Internet can access a global distributed tuple space that is realized by set of independent local tuple spaces to retrieve and deposit information. Swarms of ant-agents that represent tuples or templates roam across the network of spaces performing a kind of foraging activity that creates routes to guide application components in accessing the proper tuple space location. Although we consider both SwarmLinda and Anthill as very promising approaches, they still lack the general methodologies to engineer and configure the systems to work as desired.

Another promising thread of researches strictly related to self-organization for global scale (and not only) computing is in the multi-agent systems community. Multi-agent systems researches aim at identifying suitable models and tools to enable the definition of adaptive applications based on autonomous software components, capable of dynamically interacting with each other and with the (computational) environment in which they are situated, so to achieve global application goals. For instance, the second author has defined a specific methodology, ADELFE [4], supporting the design and development of adaptive multi-agent systems. Adaptive software is used in situations in which the environment is unpredictable or the system is open; in these cases designers cannot implement a global control on the system and cannot list all situations that the system has to be faced with. To solve this problem ADELFE guarantees that the software is developed according to the AMAS (*Adaptive Multi-Agent System*) theory [13]. This theory focuses on the design of cooperative interior medium systems in which agents are in cooperative interactions. Each agent possesses the ability of self-organization, i.e. the capacity to locally rearrange

its interactions with others depending on the individual task it has to solve. Changing the interactions between agents can indeed lead to a change at the global level and this induces the modification of the global function. This capacity of self-organization at the lowest level enables one to change the global function without coding this modification at the upper level of the system. Self-organization is founded on the capacity an agent possesses to be locally "cooperative"; this does not mean that it is always helping the other ones or that it is altruistic but only that it is able to recognize cooperation failures called "Non Cooperative Situations" (NCS, which could be related to exceptions in classical programs) and to treat them. The local processing of NCS is a means to build a system that does the best it can when a difficulty is encountered.

5. Research agenda

To conclude, we sketch a rough research agenda for what we believe are the key challenges to be faced in the area of self-organization for the design, development, and control, of spray computer applications.

First, we think that researches should rely on a deeper understanding of the global behavior of spatially distributed systems of autonomous and interacting components, in any area. This could be used to exploit principles of self-organization offensively, as discussed in Section 3.2, i.e., to achieve via reverse engineering adaptive and robust globally coordinated behaviors in a very simple and effective way. While several approaches already take inspiration from a number of natural phenomena (e.g., ant-inspired algorithms and coordination based on computational fields), a number of currently underestimated phenomena occurring in other types of spatially distributed systems of autonomous components (e.g., macro-ecology patterns of population distribution and biodiversity, physics of granular media, emergence of synchronization, morphogenesis) [44,47] are still to be deeply investigated. Such investigations could not only lead to new interesting solutions, but could also illuminate the potential emergence of possibly dangerous self-organizing behaviors in computational systems prior to their deployment. Clearly, all of this research work should be coupled with extensive simulation practice [11].

Together with the study of self-organization phenomena, it will be necessary to study which mechanisms and tools can be adopted to somehow direct the behavior such self-organized systems in a decentralized way. This will be necessary to enforce some sort of control over these systems, to ensure that they will behave as desired, despite the impossibility of controlling them in full, and keeping in to account that a full "direct" control would undermine the basic advantages of self-organization. Besides the two simple examples reported in this paper, this is a nearly virgin area of research.

An additional general issue that is to be outlined is that – in several cases – the reverse engineering of some natural phenomena may lead to nothing but *solutions looking for problems*. In these years where biologically and socially inspired solutions appear to "sell well" in the scientific community, the risk is that of forgetting real-world problems to focus only on problems for which nice biologically inspired solutions can be found, developed by reverse engineering of some natural phenomenon. Obviously, good scientific and engineering practice remind us that problems come first, and that in most of the cases

it is difficult to find a ready-to-use solution for a new problem. To solve the dilemma, one should recall that computational systems, unlike natural systems, do not obey physical laws. Rather, they live in an environment for which programmers can be the creator of any desired virtual physical law. Thus, researchers are by no means limited to find out useful applications for existing self-organization phenomena but could, in principle, invent new mechanisms and laws, leading to new self-organization phenomena specifically suited to specific application problems. While only a few visionary people will be able to "see" what these new phenomena could be and how they could be made to emerge, most of us will probably have to exploit evolutionary approaches (e.g., genetic algorithms or learning agents) to generate them in an empirical way, and then reverse engineer them as we are already used to doing with natural phenomena [16,17]. In this context, it is worth outlining that some approaches exploiting learning theories and evolutionary approaches to have a system autonomously learn how to self-organize itself have already been proposed, e.g., in the area of sensor networks [9,45] and of multi-agent systems [4].

Once the above understanding has been quite assessed, we think that there will be need to define a general purpose programming model for designing and deploying applications in such dynamic networks of spray computers, together with the development of associated middleware infrastructure and tools. One very ambitious objective could be for such a model to enable people to program, deploy, and control self-organizing and adaptive distributed applications (exploiting both direct and reverse engineering approaches) with a minimal background knowledge – in the same way as undergraduate students can currently develop excellent distributed Web-based Java applications – and independently of the specific application scenario, sensor networks rather than wide-area distributed applications – in the same way as an undergraduate student can easily and with minimal efforts adapt its applications for execution on both a Linux workstation and a cellular phone. The definition of such a model will clearly require the identification of a minimal set of abstractions enabling the modeling of salient characteristics of spray computers and their operational environments. In our opinion, approaches based on computational fields [21, 22], which have been mentioned several times in previous sections, are very promising for this purpose, by enabling one to uniformly model a wide variety of distributed self-organizing behaviors (both with direct and reverse engineering) and also to effectively model ant-inspired approaches [26]. However, this opinion is still to be verified.

Eventually, all the above researches will definitely increase our understanding of the potentials of spray computers at any scale, and will likely cause a range of new application areas to come to the fore. For instance, systems such as worldwide file sharing and artifacts like the cloak of invisibility could simply never have been conceived a few years ago. The new software and hardware technology will also call for visionary application-oriented thinkers, to unfold in full the newly achieved application potentials.

As a final note, we also want to emphasize that the widespread and pervasive diffusion of self-organizing distributed computing systems which we will assist in the next few years will not come without dangers. Even without referring to the (scientifically improbable) catastrophic scenarios depicted by Michael Crichton, more pragmatic problems will have to be faced such as pollution due to (literally) spray computers being dispersed in the environment and garbage collection of obsolete spray computer software.

References

[1] R. Albert, H. Jeong, A. Barabasi, Error and attack tolerance of complex networks, Nature 406 (2000) 378–382.

[2] O. Babaoglu, H. Meling, A. Montresor, Anthill: a framework for the development of agent-based peer-to-peer systems, in: 22nd International Conference on Distributed Computing Systems, June 2002, IEEE CS Press, Vienna, A, 2002.

[3] A.A. Berlin, K.J. Gabriel, Distributed MEMS: New challenges for computation, IEEE Computing in Science and Engineering 4 (1) (1997) 12–16.

[4] C. Bernon, M.P. Gleizes, S. Peyruqueou, G. Picard, ADELFE: A methodology for adaptive multi-agent systems engineering, in: 3rd International Workshop Engineering Societies in the Agents World, September 2002, LNAI, vol. 2577, 2002, pp. 156–169.

[5] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence, Oxford University Press, 1999.

[6] D. Braginsky, D. Estrin, Rumor routing algorithm for sensor networks, in: 1st Workshop on Sensor Networks and Applications, WSNA, September 2002.

[7] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: ACM/IEEE Conference on Mobile Computing and Networking, October 1998, ACM Press, Dallas, TX, 1998.

[8] M. Castro, M. Costa, A. Rowstron, Performance and dependability of structured peer-to-peer overlays, in: International Conference on Dependable Systems and Networks, July 2004, IEEE CS Press, Firenze, I, 2004.

[9] E. Catterall, K. Van Laerhoven, M. Strohbach, Self-organization in ad hoc sensor networks: An empirical study, in: Artificial Life VIII, MIT Press, Sydney, AU, 2002.

[10] M. Crichton, Prey: A Novel, Harper & Collins, 2002.

[11] B. Edmonds, J. Bryson, The insufficiency of formal design methods — the necessity of an experimental approach for the understanding and control of complex MAS, in: 3rd International Conference on Autonomous Agents and Multiagent Systems, New York, NY, July 2004.

[12] D. Estrin, D. Culler, K. Pister, G. Sukjatme, Connecting the physical world with pervasive networks, IEEE Pervasive Computing 1 (1) (2002) 59–69.

[13] J.P. George, B. Edmonds, P. Glize, Making self-organizing adaptive multi-agent systems work, in: F. Bergenti, M.-P. Gleizes, F. Zambonelli (Eds.), Methodologies and Software Engineering for Agent Systems, Kluwer, 2004.

[14] J. Hightower, G. Borriello, Location systems for ubiquitous computing, IEEE Computer 34 (8) (2001) 57–66.

[15] T. Imielinski, S. Goel, Dataspace — querying and monitoring deeply networked collections in physical space, IEEE Personal Communications Magazine (2000) 4–9.

[16] C. Jacob, Illustrating Evolutionary Computation with Mathematica, Morgan Kauffman Publisher, San Francisco, 2001.

[17] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kauffman Publisher, San Francisco, 2001.

[18] J. Kephart, D.M. Chess, The vision of autonomic computing, IEEE Computer 36 (1) (2003) 41–50.

[19] F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger, Geometric ad hoc routing: of theory and practice, in: 22nd ACM Symposium on Principles of Distributed Computing, July 2003, ACM Press, Boston, MA, 2003.

[20] M. Mamei, A. Roli, F. Zambonelli, Emergence and control of macro spatial structures in perturbed cellular automata, and implications for pervasive computing systems, IEEE Transactions on Systems, Man, and Cybernetics 36 (5) (2005) (in press).

[21] M. Mamei, M. Vasirani, F. Zambonelli, Experiments of morphogenesis in swarms of simple mobile robots, Journal of Applied Artificial Intelligence 8 (9–10) (2004) 903–919.

[22] M. Mamei, F. Zambonelli, Programming pervasive and mobile computing applications with the TOTA middleware, in: 2nd IEEE Conference on Pervasive Computing and Communications, March 2004, IEEE CS Press, Orlando, FL, 2004.

[23] M. Mamei, F. Zambonelli, L. Leonardi, Co-Fields: A physically inspired approach to distributed motion coordination, IEEE Pervasive Computing 3 (2) (2004) 52–60.

[24] M. Mamei, F. Zambonelli, Physical deployment of digital pheromones through RFID technology, Technical Report No. DISMI-UNIMORE-001, Università di Modena e Reggio Emilia, January 2005.

[25] J. McLurkin, J. Smith, Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots, in: 7th International Symposium on Distributed Autonomous Robotic Systems, Toulouse, F, 2004.
[26] R. Menezes, R. Tolksdorf, SwarmLinda: A new approach to scalable Linda systems based on swarms, in: 16th ACM Symposium on Applied Computing, Melbourne, FL, March 2003.
[27] R. Nagpal, Programmable self-assembly using biologically-inspired multirobot control, in: ACM Joint Conference on Autonomous Agents and Multiagent Systems, July 2002, ACM Press, Bologna, I, 2002.
[28] R. Nagpal, H. Shrobe, J. Bachrach, Organizing a global coordinate system from local information on an ad hoc sensor network, in: Information Processing in Sensor Networks, LNCS, vol. 2643, Springer Verlag, 2003.
[29] R. Nagpal, A. Kondacs, C. Chang, Programming methodology for biologically-inspired self-assembling systems, in: AAAI Spring Symposium on Computational Synthesis, March 2003.
[30] V. Parunak, Go to the ant: Engineering principles from natural agent systems, Annals of Operations Research 75 (1997) 69–101.
[31] V. Parunak, S. Brueckner, J. Sauter, Digital pheromones for coordination of unmanned vehicles, in: 1st AAMAS Workshop on Environments for Multi-agent Systems, LNAI, vol. 3374, Springer Verlag, 2004.
[32] V. Parunak, S. Brueckner, J. Sauter, ERIM's approach to fine-grained agents, in: NASA/JPL Workshop on Radical Agent Concepts, January 2002, Greenbelt, MD, 2002.
[33] D. Patterson, L. Liao, D. Fox, H. Kautz, Inferring high-level behavior from low-level sensors, in: UBICOMP, Seattle, WA, October 2003.
[34] D. Payton, M. Daily, R. Estowski, M. Howard, C. Lee, Pheromone robotics, Autonomous Robots 11 (3) (2001) 319–324.
[35] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, D. Hahnel, Inferring activities from interactions with objects, IEEE Pervasive Computing 3 (4) (2004) 50–57.
[36] K. Pister, On the limits and applicability of MEMS technology, in: Defense Science Study Group Report, Institute for Defense Analysis, Alexandria, VA, 2000.
[37] K. Pister, Invited plenary talk, in: 23rd International Conference on Distributed Computing Systems, Providence, RI, May 2003.
[38] Modular Reconfigurable Robotics at PARC, http://www2.parc.com/spl/projects/modrobots.
[39] S. Ratsanamy, P. Francis, M. Handley, R. Karp, A scalable content-addressable network, in: ACM SIGCOMM Conference 2001, August 2001.
[40] M. Ripeani, A. Iamnitchi, I. Foster, Mapping the Gnutella network, IEEE Internet Computing 6 (1) (2002) 50–57.
[41] M. Roman et al., Gaia: A middleware infrastructure for active spaces, IEEE Pervasive Computing 1 (4) (2002) 74–83.
[42] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, in: 18th ACM Conference on Middleware, Heidelberg, D, November 2001.
[43] W. Shen, B. Salemi, P. Will, Hormone-inspired adaptive communication for self-reconfigurable robots, IEEE Transactions on Robotics and Automation 18 (5) (2002) 1–12.
[44] T. Shinbrot, F.J. Muzzio, From noise to order, Nature 410 (2001) 251–258.
[45] S. Simic, A learning-theory approach to sensor network, IEEE Pervasive Computing 2 (4) (2003) 44–49.
[46] K. Stoy, R. Nagpal, Self-reconfiguration using directed growth, in: 7th International Symposium on Distributed Autonomous Robotic Systems, Toulouse, F, 2004.
[47] S. Strogatz, Synchrony, Theia Publishing, 2003.
[48] D. Tennenhouse, Proactive computing, Communications of the ACM 43 (5) (2000) 43–50.
[49] X. Wang, J. Song Dong, C. Chin, S. Hettiarachchi, D. Zhang, Semantic space: An infrastructure for smart spaces, IEEE Pervasive Computing 3 (3) (2004) 32–39.
[50] S. Wolfram, A New Kind of Science, Wolfram Media, Inc., 2002.
[51] F. Zambonelli, M. Mamei, The cloak of invisibility: Challenges and applications, IEEE Pervasive Computing 1 (4) (2002) 62–70.
[52] F. Zambonelli, M. Mamei, Spatial computing: An emerging paradigm for autonomic computing and communication, in: 1st International Workshop on Autonomic Communication, LNCS (3457) 2005 (in press).