Auton Agent Multi-Agent Syst (2008) 17:457–475
DOI 10.1007/s10458-008-9054-9

Autonomic communication services: a new challenge for software agents

Raffaele Quitadamo · Franco Zambonelli

Published online: 14 May 2008

Abstract The continuous growth in ubiquitous and mobile network connectivity, together with the increasing number of networked devices populating our everyday environments, call for a deep rethinking of traditional communication and service architectures. The emerging area of autonomic communication addresses such challenging issues by trying to identify novel flexible network architectures, and by conceiving novel conceptual and practical tools for the design, development, and execution of "autonomic" (i.e., self-organizing, self-adaptive and context-aware) communication services. In this paper, after having introduced the general concepts behind autonomic communication and autonomic communication services, we analyze the key issue of defining suitable "component" models for autonomic communication services, and discuss the strict relation between such models and agent models. On this basis, we survey and compare different approaches, and eventually try to synthesize the key desirable characteristics that one should expect from a general-purpose component model for autonomic communication services. The key message we will try to deliver is that current research in software agents and multi-agent systems have the potential for playing a major role in inspiring and driving the identification of such a model, and more in general for influencing and advancing the whole area of autonomic communication.

Keywords Autonomic communication services · Self-organization · Self-adaptation · Multi-Agent Systems

R. Quitadamo (✉)
Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia, Modena, Reggio Emilia, Italy
e-mail: raffaele.quitadamo@unimore.it

F. Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, Modena, Reggio Emilia, Italy
e-mail: franco.zambonelli@unimore.it

1 Introduction

Our everyday world is increasingly being populated with a wide variety of new communication technologies and computing devices. On the one hand, several wireless and ad-hoc communication solutions are providing people with continuous and ubiquitous connectivity [22]. On the other hand, devices such as sensor networks [17], RFID tags [53], GPS and other location systems [22], are making it easier to dynamically acquire information about the physical world and to interact with it.

These technological enhancements open up a wide range of new applications that are demanding more powerful and dynamic communication infrastructures. On-line monitoring of the world [10,17] and enhanced social experiences [40] (through context-awareness and dynamic personalization [15]) are just two examples of such applications. It can be easily pointed out that current communication infrastructures and their underlying paradigms are revealing all their weaknesses in handling the degree of *complexity* posed by those scenarios. The complexity we are talking about arises from several factors:

- *Heterogeneity of involved entities* (hardware and software). The range of new network and computing technologies is already wide and it is expected to grow consistently. Future network scenarios will see the seamless coexistence of devices as diverse as micro computer-based sensors and high-end computers, and communication technologies as Wi-Max, Bluetooth, and ZigBee. This introduces the need for network and computing models flexible enough to integrate all these technologies, along with a major effort to standardize interfaces and protocols and enable full interoperability.
- *Dynamics of the Network.* As connectivity is becoming ubiquitous and computers are getting embedded in our everyday objects, communication networks turn out to be highly dynamic in terms of *topology* and *usage patterns*. From the standpoint of network topology, wireless-based technologies, such as sensor networks, are dramatically challenging classical IP-based communication protocols and their associated routing and discovery services. From the point of view of the end-user, access to wireless ubiquitous devices may enable personalization of services, yet such transient communication resources (e.g. a localization service provided by a geographical sensor network) is characterized by the inevitable presence of fluctuations, e.g. caused by peers/devices continuously joining and leaving the network or simply by the physical mobility of nodes.
- *Decentralization and control*. The highly decentralized and embedded nature of the involved devices makes it hard to enforce some forms of direct control over such entities. This makes traditional centralized approaches to configuration and security management inadequate and economically unbearable, and calls for novel decentralized solutions, which can limit the need for direct human intervention.

All the above implies identifying suitable models and tools by which communication services (at both the infrastructural and the user level) can be made more *flexible* and *dynamically adaptable*. Communication services should be able to properly react to the dynamism and unreliability of the network without suffering relevant malfunctioning; they should be able to increase user satisfaction by adapting their behavior to the current context (physical and/or social) of users and to their own individual needs. All those characteristics cannot always be accommodated in current communication models and architectures.

The interdisciplinary research area of *autonomic communications* [16,46] attempts at overcoming the limitations of current communication models and architectures. Autonomic communications generally refers to all those research thrusts [33,49] involved in a foundational rethinking of communications, networking, and distributed computing paradigms,

to face the increasing complexity, decentralization, and dynamics of modern network scenarios. The ultimate vision is that of a networked world, in which networks and associated devices and services can work in a totally unsupervised—i.e., *autonomic*—way, being able to self-configure, self-monitor, self-adapt, and self-heal. Such ambitious research goals are somewhat shared by the close research area of *autonomic computing* [28]. Although overlaps exists between the two areas, autonomic computing typically focuses on application-level software and services and on the dynamic management of computing resources, and assumes the capability of acting in closed systems where full control over software and resources can be exerted. Autonomic communications, instead, considers open and decentralized systems, and focuses on the management of network resources and services at both the infrastructure and the user level [16]. With this regard, among the topics of interest to autonomic communications we can include: semantic communication models [15], dynamic and non statically-layered protocol stacks [45], adaptive and evolvable architectures for network components [46]; and, last but not least, the search for suitable paradigms and tools for the design, development, and execution of *autonomic communication services* (at both the infrastructural and the user level) to flexibly and autonomically act in open and decentralized scenarios [33].

In this paper, we specifically focus on autonomic communication services and on their relations with multi-agent systems, and will try to achieve several goals. First, in Sect. 2, we introduce a simple communication scenario and analyse some tangible reasons that motivate our search for a novel approach to the design of autonomic communication services. Specifically we identify the need for a general-purpose "component model", to act as the basic building block for autonomic communication services, and that will have to exhibit the typical properties of agents. Following (Sect. 3), we survey and critically analyse current research approaches in the area of autonomic communications and (being somewhat related) of autonomic computing. The aim is to outline the strict relations of these research with those in the area of multi-agent systems and of agent-oriented software engineering. In particular, we show that agent-based approaches can effectively go beyond the dichotomy between self-adaptive and self-organising approaches, and take advantage of both. Eventually, in Sect. 4, we try to synthesise the key desirable—yet challenging—features that we expect from a general-purpose component model for autonomic communication services.

The key message we hope to deliver is that current (and past) research in software agents and multi-agent systems have the potential for inspiring and driving the identification of such a model, and more in general for influencing and advancing the whole area of autonomic communications.

2 Autonomic communication services

The vision of autonomic communications is that of a networked world, in which networks, devices and services will be able to work in an autonomic way [16,28,46]. The idea is to consider networks as sorts of immense organisms. Components (both hardware and software) are conceived as parts of these organisms, and are expected to be able to prosper and survive contingencies autonomously [33]. To motivate such a vision, we sketch in the following subsection a pervasive communication scenario, where several and diverse devices are involved. On this basis, we stress the compelling need for a new software component model for *autonomic communication services* (hereafter abbreviated as *ACSs*), capable of meeting some key requirements. A presentation of some selected "software agent" ideas that will significantly influence our ACS model throughout the rest of the paper is also given.

2.1 Scenarios of autonomic communications

Traditional communication and distributed computing paradigms were conceived to target a now obsolete perspective of computer networks: wired networks of (rather homogeneous) medium/high-end computers and routers; network disconnections and failure of components were considered exceptions, and network and system managers were able to act on the system for reconfiguration and fault-recovery.

However, as stated in the introduction, modern network scenarios increasingly include very heterogeneous entities, interacting over a variety of wireless channels, and in the presence of mobility (of both devices and users exploiting them). There, failures and network disconnections are the norm rather than the exception, and the possibility for network and system managers to intervene in the system is challenged by its intrinsic decentralization and complexity.

Just to reach a better understanding of what such scenarios could look like, consider how our cities are evolving (see Fig. 1). First, a variety of computer-based sensors are increasingly spread around in every street, crossing, square, and within buildings. We already find a variety of simple sensors around in our cities (e.g., to measure traffic intensity and pollution), but the future will see these sensors become wireless-enabled, and dramatically increase in density and diversity. It will be possible, for instance, to determine in real time how many free benches are there in a specific park or how long is the queue at the nearest post office. Second,

Fig. 1 An urban scenario of autonomic communication

wireless-enabled computing devices are increasingly being worn by people and embedded in cars. Such devices, beside the capability to access the Internet, will also be able (via ad-hoc wireless communications) to directly interact with each other, and localize themselves via GPS or other means. Third, all of these devices will be able to mobilize data from and to the Internet, based on a variety of communication channels, from WiFi, to UMTS, or satellite communications.

The heterogeneity of entities and network technologies involved in the above scenario is evident, as it is the fact that the resulting network is highly dynamic (due to the ephemeral and mobile nature of wearable and embedded devices) and highly decentralized (no system manager can enforce a strict control over dispersed sensors and over personal devices). These factors clearly justify the efforts of autonomic communication research towards the identification of more advanced ways of modeling communications by means of *ACSs*, as explained in the following subsections.

2.2 Towards autonomic communication services

The term *communication service* refers to a functionality available within a network to access and exploit communication resources and devices. We emphasize that the distinction between "communication service" and "computing service" is not and has never been clear-cut, being mostly dependent on the observation point. IP datagram routing, DNS, socket-based point-to-point communication, cryptographic tools and P2P data delivery can all be considered communication services. The definition applies independently of the fact that such services act either as "user-level" services or as "infrastructural" ones. Services used to access sensing devices are better considered as communication services. They are in fact exploited to integrate such devices into the networked scenario, to improve the behavior of the network, and/or to add functionalities and properties to other communication services.

In the sketched scenario, we can consider "communication services" all those services that should be put in place to properly access and exploit the available networked resources at the best:

- At the more *infrastructural level*, one can think at localization services that, by exploiting GPS, WiFi signal strength, or other localization tools, are able to provide the location of users, cars, or other devices. In addition, a variety of routing services can be made available to deliver data and messages across the network. From traditional routing services, offering delivery to a specific network/host ID, to more advanced routing services, capable of delivering messages to and across mobile devices, at specific locations in the physical environment, or at specific groups of nodes or users.
- At the *user level*, one can think of enhancing "traditional" communication service with ubiquitous/mobile access and contextual (i.e., location-based) adaptation. In addition, one can think of making available to users various services to query the physical world and obtain several information about the surrounding situation (there included about other users). Those pieces of information can possibly be integrated with information dynamically downloaded from the Web [10]. Another relevant example is that of services to alleviate traffic congestion in the cities. This implies devices in cars (for computing, sensing and visualization) to interact with devices in streets and crossings (for sensing the current traffic situation and communicating it to vehicles). Cars can also interact with each other and form wireless ad-hoc network that can be used to properly forward information across the town. A global traffic service can then exploit all these available data to map in real-time the status of streets in the city, and calculate on-the-fly faster routes for users that avoid congested or likely to become congested areas.

Many of the mentioned communication services are already a reality or are being actively experimented [1]. Yet, following the current traditional networking approach, network services are rigidly organized in layered communication stacks, with somewhat rigid hard-coded logics, predefined interaction patterns, and strict limitations in accessing information across layers (research work on cross-layering is being carried out, e.g. in sensor networks [34]). Unfortunately, supporting autonomic features in services with such a rigidity can either end up in a worthless effort or lead to tricky and inefficient solutions.

Thus, future autonomic communication scenarios require the adoption of an innovative *software component model*, to design and implement flexible communication services according to dynamic patterns. Such components should live and interact in the network without being tied to rigid or layered structures and without being constrained in information access. They should self-configure and self-adapt as needed to provide the needed functionalities. Each component should act as an access point to the actual service, regardless of layering constraints. Moreover, depending on the service, a component should be able to implement the service either in autonomy or by interacting with other components. For instance, a component implementing an autonomic localization service should always be able to provide information in a best-effort fashion; it should not rigidly require the availability of specific localization devices (like an adjacent service layer in the stack), but it should rather opportunistically exploit a variety of available services. As another example, a routing service should guarantee message delivery in very dynamic and mobile networks, without requiring manual reconfigurations, and should possibly tune quality-of-service depending on the specific needs of the user/application exploiting it.

2.3 Requirements for autonomic communication components

Provided that ACSs are designed as software components, our next step is to identify a number of essential requirements that such a component model should satisfy. In particular:

- *Self-management.* A component model for ACSs implies the capability of components (both at the individual level and at an aggregate social level) to continuously manage their internal functional and/or non-functional properties independently of contingencies, just like a living organism is able to maintain its internal balances.
- *Self-reconfiguration*. ACSs execute in highly dynamic and unpredictable network scenarios, due to the presence of wireless channels, mobile and ephemeral nodes. For a component model, this means the capability of supporting the dynamic self-reconfiguration of components and of their composition/interaction patterns, without requiring a predefined schema and/or explicit human intervention.
- *Situation-awareness*. To achieve self-management and to reconfigure at need, the component model should be aware of the surrounding context. The technologies needed to capture contextual data are becoming increasingly available (e.g., sensorial data, location systems, network monitoring tools, user profiling tools). However, there is a need to evolve from simple models of "context-aware" components (expressing the simple capability of accessing contextual information) to models of "situation-aware" components, in which components are given access to properly elaborated and organized information concerning complex "situations".
- *Abstraction and technology-independence*. As the sketched scenario identifies, it is expected that ACSs will run on a variety of heterogeneous devices and interact through a variety of communication technologies. Therefore, for a component model to be technology-independent, it must tolerate a design that can abstract from the actual characteristics of the underlying hardware. Deployment and execution of ACSs should be possible on

diverse devices, such as embedded sensors and high-end computers, without forcing design and developers in adopting radically different conceptual and practical tools in dependence of the target.

- *Scalability.* Given the possibly very large scale of the target scenarios, a component model for ACSs must promote scalability of both design (i.e., software engineering scalability) and execution complexity (i.e., performance scalability). In other words, the component model should be based on design principles that can be practically applicable to small systems as well as very large systems, and should promote organizing services according to patterns that exhibit scalable performances (or quality of service).

2.4 Agents as autonomic communication components

Based on the above discussion, it emerges that ACSs can hardly be modeled and implemented as "passive software components", like in standard layered protocols stacks and service-oriented architectures [24]. Rather, they should be exposed by "active" components endowed with inherent autonomic features of self-management, adaptivity, and situation-awareness, and capable of dynamically interacting with each other.

These considerations directly bring us to the key point of the paper: a component model for ACS should exhibit those characteristics that are widespread recognized as key distinguishing characteristics of agents and multiagent systems: autonomy, situatedness, and sociality [26]. Thus, independently of whether the "agent" keyword will ever make it through the autonomic communication community, many ideas and approaches developed within the agent community can indeed be of great inspiration to our ACS component model, and the design and development of ACSs can indeed be considered as a process of building distributed multiagent systems.

For instance, agent-based software engineering have always highlighted the importance of modeling systems drawing inspiration from human and ecological societies: agents live and interact (being *proactive* and *autonomous* entities) in complex social organizations and *react* to events and situations generated into the environment [54,58]. Indeed, this is what we also search for autonomic communication services immersed in dynamic network environments. As another example, in multiagent systems economies individual agents with rational behaviour interact with other agents accordingly to specific mechanisms to ensure that a stable economic market emerges [21]. Again, this is something that relates to the search of self-regulatory mechanisms to ensure network stability. Similar examples could be found for the areas of cooperative agents and robots [29], for trust and reputation-based systems [56], and for swarms and emergent agent systems [9].

To push the discussion forward, the following section surveys and analyses the key research directions that have been pursued so far in the areas of autonomic communication (and autonomic computing) to realize the vision, and relate these with research proposals in the area of multiagent systems.

3 Approaches to autonomic communications

Several strategies have been studied to integrate autonomic features in software and communication systems. As depicted in Fig. 2, two opposite viewpoints have emerged, if we take into account the degree of emphasis either on individuals or on groups. At one extreme, the idea of *self-adaptation* puts the emphasis on individuals: the basic idea is of inherently autonomic components [28], which continuously monitor the environment and spontaneously trigger

Fig. 2 Group behavior versus individual self-adaptation

corrective actions whenever needed. At the other extreme, *self-organization* puts the emphasis on groups and considers instead that adaptive behaviors can emerge from the interactions among simple components, without any central control (i.e. the so called swarm intelligence [30,39]).

These two "extreme" proposals, being the foundation stones for autonomic communications, are surveyed in the next two subsections. They are also compared to similar research in agent and multi-agent systems, again distinguishing between proposals with the emphasis on individuals (i.e., on proactive and intelligent agents) and on collectives (i.e., on systems of simple reactive agents). Yet, these two approaches are far from representing the whole spectrum of possible solutions. Multi-agent systems research has studied, identified and proposed models that can hardly fit into any of these. Such models tend to properly mix and integrate features of both self-adaptation and self-organization.

3.1 Self-adaption and autonomic component models

IBM, in its "manifesto" for the autonomic computing initiative [25], first raised the issue of defining proper component models for innovative "autonomic" software systems. The *autonomic communication* research area inherits from autonomic computing the keyword but it differs from it in its being more oriented to open and decentralized systems and to management of network resources at both the infrastructure and the user level [46]. Nevertheless, both research areas recognize that traditional software systems get less and less advantages by continuous technology advances (e.g. powerful CPUs, large memory availability and so forth), because the complexity of development and management are indeed becoming overwhelming.

Autonomic computing starts from the assumption that, to enforce an autonomic behavior, a complex software system should exhibit "self-*" properties at both the level of individual components and of the whole system.

At the *individual component level*, such an assumption leads to an "autonomic component model" in which every component of the system must be inherently self-managing. This implies that each component is responsible for: (i) configuring itself internally; (ii) healing over internal failure where possible; (iii) optimizing its own behavior; and (iv) protecting itself from external probing and attack. To this end, a generic architectural model for such autonomic component model considers two logical parts [28]: the *managed resource* and the *autonomic manager* (see Fig. 3). The managed resource can represent the internal data and functional logic of the software component, or the services provided by it. More in general, the component model considers the possibility of wrapping any resource into a component (e.g. a CPU, a database, etc.). The autonomic manager acts as a "supervisor", in charge of understanding if everything is working fine, and in charge of actuating configuration actions whenever needed, to ensure the safety of the managed resource. The most assessed model describes the autonomic manager as the driver of a so-called "MAPE-K" (Monitor, Analyze, Plan, and Execute, based on Knowledge) feedback loop on the managed resource.

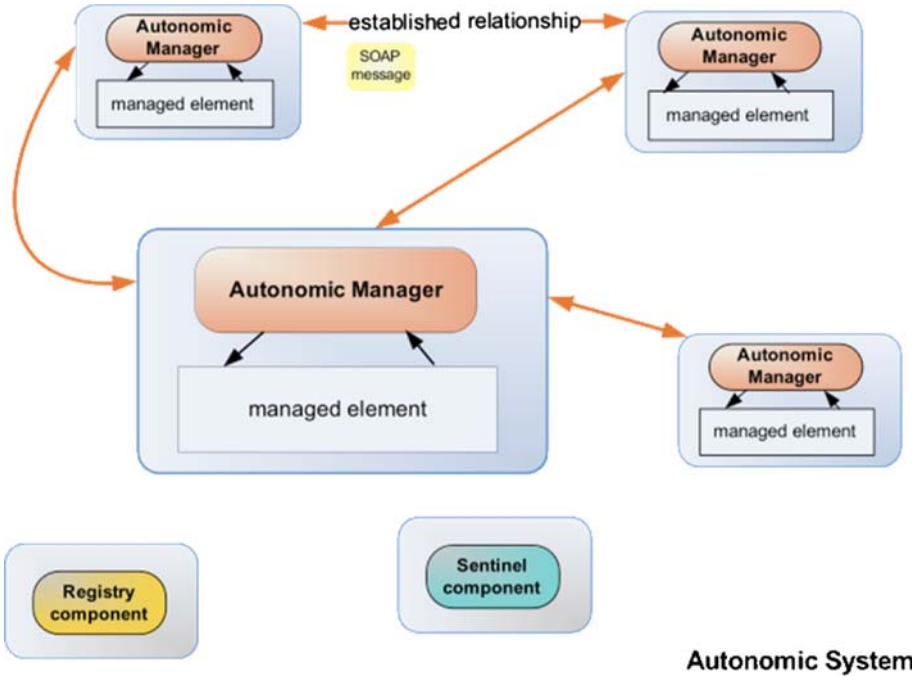

Fig. 3 Autonomic components in the IBM perspective

At the *level of software system*, the general idea is to enforce coordination between components according to a service-oriented model [24]: components are assumed to be characterized by well-defined interfaces, described sometimes as web services [8] (e.g. exchanging SOAP messages between WSDL service interfaces) or with standard OGSA interfaces [36]; no other means of communication between the elements is permitted. The latter principle allows to completely specify the interactions between the elements in terms of the interfaces that they support and the behaviors that they exhibit through these interfaces.

Although the above component models have been studied mostly in the area of autonomic computing (i.e., with a focus on closed systems and on the application level only), many of such considerations can apply to autonomic communication systems as well.

The similarities of the above "autonomic component models" with assessed multi-agent systems models are sharp, at both the individual and the system level [26,55]. Autonomic components are goal-driven autonomous entities, able to sense and react to their surrounding environment (the "K" of knowledge in the "MAPE-K" feedback cycle). "MAPE-K"-based autonomic components can thus be considered as a sort of reference agent architecture, explicitly conceived to enforce internal autonomic behavior in components. However, the question arises of whether—among the large number of agent architectures conceptualized and studied in the past few years—more suitable architectures can be defined. For instance, we feel that the well-assessed BDI architecture [43] can subsume in many aspects the "autonomic component model". The BDI architecture considers agents as *intentional entities*, whose behavior can be modeled and explained in terms of beliefs, desires, and intentions. Functional and non-functional desiderata of an ACS can be modeled in terms of "desires"; the knowledge acquired from internal self-monitoring and the external context can be considered as

"beliefs". Consequently, the expected autonomic and situated behavior of the ACS is achieved by simply defining those plans of actions (i.e., "intentions") that enable self-configuration, self-adaptation, self-healing and self-protection.

Shifting to the system level, many mentioned ideas have much in common with several (e.g., FIPA-based [6]) agent infrastructures. However, autonomic computing research also recognizes the need for special supervisor components (e.g., the sentinel in Fig. 3), devoted to control and rule interactions at the system level. Only recently [4] multi-agent systems research has started to recognize the need for infrastructure-level tools to monitor and drive agent interactions [37,44]. Others have also defined methodologies to engineer systems based on such infrastructures [44,58]. However, so far, most of these tools and methodologies are oriented to ensure organizational coherence and respect of social rules [20], while the issue of exploiting them to enforce autonomic and adaptive behavior at the system level is mostly unexplored.

3.2 The self-organization approach

At the opposite side of the spectrum of Fig. 2, an increasing number of proposals suggest the possibility of enforcing some forms of self-management through nature-inspired phenomena of *self-organization*.

Many natural and biological systems are able to spontaneously produce forms of structural and/or functional self-organization; those behaviors emerge in the system without any explicit control, but simply descending from local interactions among a large number of very simple individuals (e.g., ants, bees, termites) [7,30]. Many of these systems are also *adaptive*, in that they spontaneously react to changes by dynamically reshaping their internal structure without having the structure lose its specific properties. In the past few years, several potential applications of nature-inspired form of self-organization have been documented, most of which of direct interest to the area of autonomic communications [3]. Examples include adaptive routing algorithms inspired by the phenomena of ant-foraging [12], security services inspired by the human immune system [35], self-aggregation and self-differentiation algorithms for load balancing of distributed resource usage [3].

At the level of *individual components*, the enforcement of forms of self-organization typically requires the presence of active components; they need to be just capable of executing simple algorithms in reaction to environmental events and of locally interacting with each other. At the level of *software system*, some sort of infrastructure must be provided supporting local interactions between components. In many cases, though, naturally inspired forms of self-organization take place with indirect interactions through a shared environment [27,52]. This is the case of interactions mediated by pheromones, as in ant colonies, or by spreading of chemicals, as in multi-cellular organisms (a mechanism known as *stigmergy* [7,23]. It is worth outlining that the environment plays an active role in stigmergic interactions, in that it is doing duty for supporting the diffusion of chemicals and their evaporation. Accordingly, a computational infrastructure to support nature-inspired forms of self-organization has to embed processes devoted to support similar forms of diffusion/evaporation.

In the area of multi-agent systems, such phenomena of self-organization have been extensively studied and pioneered [41]. The simple components of a self-organizing system can be indeed considered as simple *autonomous reactive agents* [39]. However, despite the large number of theoretical and simulation studies, a few practical tools exist to support the development of software systems with reactive agents and (stigmergic) self-organization. The DIET platform, conceived within the EU DIET project and now donated to the open-source community [13], defines a simple model of reactive agents and the corresponding infrastructure,

as a general framework to build self-organizing agent systems. The TOTA (Tuple On The Air) middleware [32], not interested in defining a new agent model, focuses instead on creating a flexible and usable infrastructure for supporting a wide variety of stigmergic interactions.

3.3 Discussion

The above survey strongly suggests that both inherently self-adaptive components and swarms of simple self-organizing agents can be fruitfully exploited to implement future ACSs. Moreover, it is likely that hybrid forms of the two approaches will emerge, trying to balance benefits and drawbacks of each one. The heterogeneity of resources and devices in autonomic communication scenarios requires a critical comparison of the pros and cons in these approaches. This subsection analyzes self-adaptation and self-organization in the light of the requirements identified in Sect. 2.3, with a twofold purpose: (i) giving the reader some elements/recommendations to choose the best approach for his particular case; (ii) motivating the need for a unifying agent-based model that allows the designer to seemingly use both self-adaptive and self-organizing ACSs (and any solution in-between).

Self-management. Both component-level self-adaptive approaches and system-level self-organizing ones exhibit forms of self-management. On the one hand, autonomic behavior in self-adaptive approaches is enforced "by design", and is made explicit through specific behavioral rules embedded in components. On the other hand, self-management in self-organizing approaches is only implicitly encoded into the local interaction rules of components, which can ensure the achievement and the preservation of some global emergent behavior in the system. The enforcement of autonomic behaviors at the component level is much easier in self-adaptive approaches, because designers are given direct control over them. In self-organizing approaches the designer has instead nearly no control over individual components, but he can only influence the convergence of the system to a certain status.

Self-reconfiguration. Services can reconfigure themselves to adapt to changed conditions (e.g., different user needs or changed network conditions). This can occur by proper coding of specific "adaptation rules" in self-adaptive approaches, while it occurs via automatic reconfiguration of the structure in self-organizing systems. Also in these cases, there is not a definite answer to what approach is to be preferred. Self-adaptive approaches ensure a higher-degree of programmability, since it is possible to code how individual components and the whole system should reconfigure in response to what conditions changed. However, taking into account all possible reconfiguration patterns, notably for very large systems, turns out to be an overwhelming effort. In the latter case, a self-organizing system can simplify things because it can inherently reshape its internal structure, anyhow perturbed from the outside. This capability is nonetheless given at one precise cost: there is usually little possibility for designers to drive *in a timely fashion* (e.g. real-time) the natural adaptation of such systems, e.g. to control how and when they should change their behavior and structure.

Situation-awareness. Both self-adaptive and self-organizing behaviors are often enforced by making components and systems aware of their surrounding context. As already stated, this implies not only having access to contextual information, but also understanding and elaborating complex knowledge about the surrounding status of the system. Self-adaptive components are easier to design in such a way that their internal behavior (both their functional and self-adaptive behavior) is continuously fed by contextual information. Self-organizing components, on the other side, rely on very limited contextual information. For instance, in stigmergic models of interactions, the only contextual information available to components is the local gradients of diffused pheromones/chemicals. Of course, this prevents components from performing complex elaborated reasoning about situations, which may be instead

desirable in several cases. Anyway, the limited information available to self-organizing agents can be more than enough in several applications. For instance, a local gradient of pheromones condenses in itself information about what other ants have passed by recently carrying food, and what directions they have taken to reach the nest.

Abstraction and technology-independence. This requirement has been defined in Sect. 2.3 as the capability of the component model to suit the development and deployment of service components on a wide range of computing devices and with a variety of communication technologies. Clearly, self-organizing approaches, being based on simple components with mostly simple local interaction patterns, are the ideal candidates for this requirement. On the contrary, it is less natural to deploy complex self-adaptive components on a network of small sensors, due to their internal knowledge reasoning capabilities and their need for heavyweight middleware infrastructures.

Scalability. Considering execution scalability first, self-organizing approaches offer more advantages. Self-organizing systems are inherently scalable to very large networks, because components act independently of each other and make decisions based on locally available information only. At the opposite, self-adaptive approaches, in which the global adaptive behavior of a system often takes place through "behavioral contracts" between components, can difficultly scale to very large systems. However, when shifting to software engineering scalability (i.e., modularity), it appears that self-organizing approaches have currently very little to contribute. In fact, in self-organizing approaches, issues such as encapsulation of behavior, composability, confinement of effects, have never been properly addressed. This makes it difficult to (i) conceive and design services in terms of self-organizing components, and (ii) to develop complex networks of services, composing different classes of self-organizing systems. On the opposite, self-adaptive approaches, grounded on more traditional software engineering lessons, explicitly tolerate encapsulation and can achieve software engineering scalability, at least in the presence of careful design choices (e.g. the usage of proper design patterns).

To summarize the discussion, our model for ACS should give the designer abstractions to implement his services with a behavior that can range from self-adaptive to self-organizing according to the problem at hand. The features of self-adaptive approaches are needed to enable the development of components and systems, which can provide high-level knowledge-based services at the best of their possibilities and can enforce flexible and controllable adaptive behaviors. The features of self-organizing systems are needed to enable the development and deployment of lightweight and fine-grained services, suitable for low-end devices and for large-scale networks, as well as to enable the achievement of adaptive behaviors in more limited and simple ways.

4 Towards a unifying component model: challenges and future work

The outlined need for an agent model able to exploit and integrate both self-adaptation and self-organization for ACSs is quite challenging. The identification of an ultimate and agreed-upon model is still part of an active debate in the autonomic communications community [48]. Therefore, we have tried so far to shed some light on this new research field, first "isolating" a minimum set of requirements for ACSs (Sect. 2) and then describing the state of the art (Sect. 3) in the area. This last section aims at being at the same time an "analysis of important experiences from the past" and "a dive in the future challenges of ACSs". Our intent is to further support our statement in favor of an agent-based model for ACSs and provide the reader with new fertile ground for future agent research. More specifically, we

try to synthesize all the above discussions and analysis by sketching the key capabilities that we *envision* for such a unifying component model:

- The capability to spontaneously *aggregate* (hereafter called *self-aggregation*) ACSs in different forms, to enable multiple distributed agents to collectively and adaptively provide a distributed communication service on demand;
- The capability to *enforce control* in the ecology of ACSs;
- The capability to exhibit *self-similar* forms of aggregation, i.e. reproducing nearly identical structures over multiple scales, and achieving software engineering scalability [14];
- The capability to *organize situational information*, promoting more informed adaptation choices by agents and advanced forms of stigmergic interactions.

4.1 Self-aggregation as an adaptation mechanism

In an "ecology" of adaptive ACSs, *self-aggregation* (or coalition formation) is a key mechanism to build and exploit complex distributed communication services. Self-aggregation is clearly an autonomic adaptation process, in that it must occur on need and without direct human intervention: whenever some changes happen in the surrounding environment, an ensemble of simpler ACSs can decide to form a coalition that can better handle the new unforeseen situation or provide an improved service.

Self-aggregation helps achieve an improved service adaptation. Whenever the execution of a service requires more resources or a certain QoS has to be guaranteed, one or more ACSs may decide to join and work in cooperation to provide such an improved service. Adaptation means also offering a better degree of fault-tolerance and service availability: self-aggregation clearly helps ACSs spontaneously reorganize themselves and, for instance, replace lost resources or unavailable services with other ones reachable in the surroundings. As it will be stressed in Sect. 4.4, this feature demands a powerful knowledge infrastructure to enable situation-awareness of ACSs and make them carry out informed aggregation choices, whenever environmental conditions change.

Enabling self-aggregation in our agent model implies rethinking traditional integration architectures both from an *architectural* and a *behavioural point of view*.

Let's consider, first of all, the *architectural* viewpoint, i.e., how our ACSs should be designed to support aggregations formation. The flexible and non-strictly-layered service model we envision implies that ACSs should be allowed to participate in more than one aggregation, e.g., because their service can be shared among different clients. This requires that our ACS should be more than simply a "service provider" with a fixed published interface. We envision a new concept of *interface* that is far more flexible than classical software component interfaces. When a service agent participates in an aggregation, its interface should be updated or, better, it should expose a new interface: the "aggregate service interface". Such an interface is expected to be the same in every aggregate ACS and its provided operations are negotiated and constructed according to the aggregation strategy and the requirements of the new complex service. Moreover, ACSs are very often located on different network nodes and the aspect of their physical distribution should be also taken into account. This leads as a consequence that our agent model should transparently support "at least" both centralised and distributed service aggregations (see Fig. 4) and we said "at least", because there might be intermediate or hybrid solutions between these two extremes. *Centralized aggregations* are those where a brand new "aggregate ACS" is the new centralized access point to the service and the service is not physically distributed. *Distributed aggregation* are those where several ACSs decide to join an aggregate service, but they still preserve their physical distribution in the network. In other words, they all agree on a common service interface, but

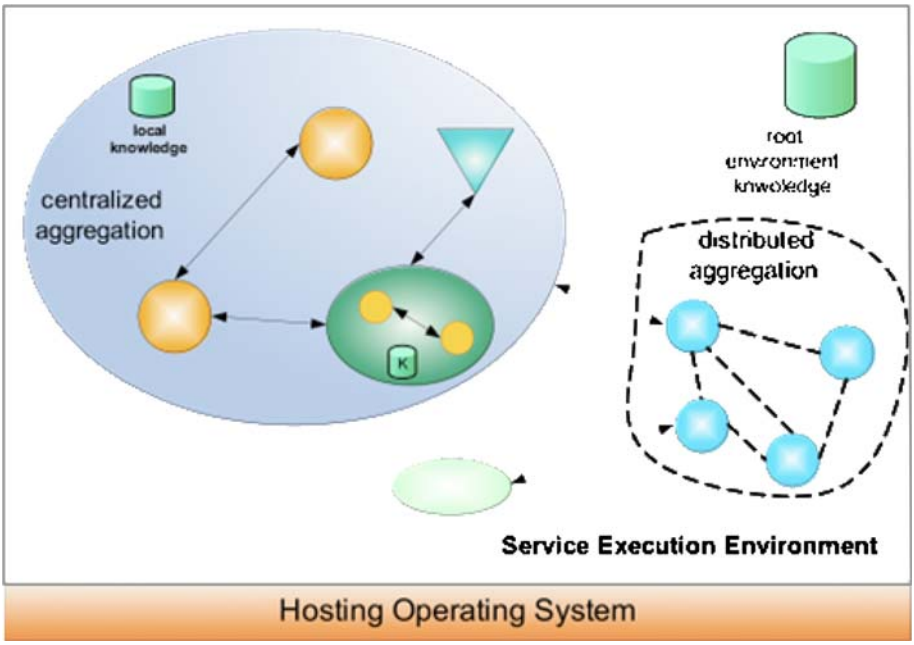

Fig. 4 Self-aggregation of ACSs

there is not a new "aggregate ACS" exposing it; every single participant instead is considered "access point" to the aggregate service and is equipollent to all others (i.e. it exposes the same interface and compound behavior). Distributed aggregation is essential in our ACS model to properly leverage self-organizing emergent services, which are intrinsically distributed. In the latter case, this kind of aggregation has the additional advantage of giving a more *resilient* behavior to the aggregate ACS, because in a set of very small and equivalent ACSs there is no single point of failure (unlike the centralized aggregation case). Furthermore, its communication interface is replicated in several agents and this helps achieving improved service accessibility.

Besides architectural design choices, self-aggregation needs effective algorithms and tools to work in dynamic and open environments, without human intervention. From a more *behavioural* standpoint, ACSs are expected to support different aggregation techniques, which are an active research area of AI. Innovative algorithms will have to be explored to face the challenges of distribution and heterogeneity in ACSs. Examples of important contributions to this purpose have been proposed for task allocation problems and coalition formation algorithms [11,47]. All this MAS research are more than a valuable starting point for the challenging task of enabling autonomic self-aggregation in our ACS model.

4.2 Enforcing control over service aggregations

As already highlighted, one of the driving principles of the autonomic communications vision is that services should be *self-managing*. The fundamental problem when trying to enable self-management consists in establishing some kind of control over ACSs, in order to constantly guarantee an optimal overall functionality, protect against malfunctioning parts and so forth. As already explained in Sect. 3.1, the IBM proposal for building autonomic components [28] is based upon the introduction of the so-called "autonomic manager", which is

an intelligent software entity that monitors the activity of its managed resource, and can take corrective actions in a sort of continuous control loop.

The agent model we are sketching in this paper should allow both traditional autonomic managers and self-organising approaches, but here we deem crucial to foster also an innovative vision of self-management [57] tailored to the features of autonomic communications. Many communication scenarios are dramatically distributed, often without any clearly identifiable stakeholders. Thus, a possible solution to enforce some forms of control over a swarm of self-organising service components can be that of populating the ecosystem with additional "manager components", capable of somehow affecting the self-organising behaviour of the system. That is, without completely taking control over the self-organising behaviour of the system, being able to drive it towards specific emergent structures of organisation. In an environment where every service, even the most basic one, is provided by an ACS, it is reasonable to assume that such manager components could be ACSs themselves, injected on demand into the system. They will have to live inside the system and interact with other ACSs. This brings as a consequence that the knowledge management and planning capability, previously placed as a possibly heavyweight burden on every component, is now "externalized" and made distributed across the various deployed manager agents. The big difference with sentinels (see Sect. 3.1) is that those managers are not only in charge of monitoring/supervising the system. They drive the behaviour of an ensemble of agents, taking functional decisions thanks to their enhanced reasoning capabilities.

To some extent, one can affirm that the proposed means of control situates in between the two extremes of self-adaptation and self-organization previously discussed with reference to Fig. 2. It must be pointed out that some of these ideas have already been experimented and formalized in multi-agent systems research. In a previous work, we have studied how to control the emergence of spatial structures in cellular agent system by injecting a few percentage of "manager" agents in the lattice [31]. The ideas of Electronic Institutions and norm-aware agent societies have been proposed as a model to rule interactions among software agents using norms (e.g. obligations, permissions, etc.) [51]. For instance, in [2] and subsequent work, norms are explicitly represented and managed by way of rules and a team of "administrative (institutional) agents" is deployed in the distributed architecture, to ensure normative positions are complied with and updated by individual agents.

Clearly, experience from the above and other research in the area of self-organized control [57] will be of paramount importance to formalise an "ecology-inspired" autonomic service model [42]. Still, as of now, the extent of control that can be exerted by distributed "manager components" in a self-organising system is still to be fully investigated and understood, as it is the identification of suitable infrastructure to support such control features.

4.3 Achieving robustness of aggregations via self-similarity

In our service agent model, applying the *self-similarity* principle means that "individual components self-organize and self-aggregate to reproduce nearly identical structures over multiple scales" [14].

From a software engineering point of view, having the same structural and organizational principles in force at different scales facilitates the management of services. From a more architectural standpoint, self-similar structures are known to be intrinsically robust. Many biological systems exhibit such properties, thanks to their being organized in hierarchical and self-similar structures at different scales.

A successful agent model for ACSs should therefore support self-similar aggregation, and multi-agent systems research has already explored some important applications in this

direction, introducing the idea of *holonic agents* [18,19], so far mainly applied to manufacturing scenarios. The term *holon* was originally introduced by the philosopher Arthur Koestler to name recursive and self-similar structures in biological and sociological entities. Building holonic service compositions enables the construction of complex systems that are efficient in the use of resources, highly resilient to disturbance (both internal and external) and adaptable to changes in their surrounding environment. Holarchies (i.e., service aggregations) are *recursive* in the sense that a holon (i.e., an agent component of such an aggregation) may itself be an entire holarchy that acts as an autonomous and cooperative unit in the first holarchy. The *stability* of holonic service aggregations stems from holons being independent units, which handle circumstances and problems on their particular level of existence (i.e., the local execution environment of the aggregator agent), without asking higher level holons for assistance.

Learning from such experiences with holons and self-similarity, self-similar aggregate ACS will be able to participate in further aggregations/holarchies or simply exist as new available services, but always as independent units: hiding their internal complexity under a self-similar interface, they can react to changes in the environment and adapt to different situations, transparently re-organising their internal structure.

4.4 Organizing situational data into knowledge networks

Another essential requirement for ACSs is their capability to perceive their surrounding context and consequently adapt and improve their behavior. As already stressed, information about the context are expected to be increasingly important to enable *situation-awareness* in next generation communication services.

Nowadays, several mechanisms exist to produce situational data from the environment (e.g., intelligent sensors or monitoring mechanisms) and such data is expected to increase to dramatic amounts soon. Such a huge amount of information cannot be fully managed or internalized by every ACS, because it would require a significant knowledge management capability that we consider an "avoidable burden" in our agent model. Our basic idea is that situational data should be scattered part in the environment (e.g., in a shared tuple space [38]) and part across the different ACSs. In further details, we envision that when ACSs decide to form aggregations, they share their pieces of contextual knowledge with the other participants, forming a sort of "aggregated situational knowledge". The global knowledge is dynamically built by the various ACSs that join and leave the system during the execution.

The final conceptual outcome of the above knowledge organisation and analysis phases is the formation of so-called *knowledge networks* [5], in which all information about individual contexts are properly represented, organized and correlated. Distributing this knowledge in the environment, ACSs can self-organize their activities using "cognitive stigmergy" approaches [50]. As already anticipated, a distributed knowledge network is expected to play the part of a high-level intelligent and dynamic environment, useful in particular for those self-organizing services that use the environment as a mediator for their local stigmergic interactions [54]. Self-adaptation and self-organization would thus be driven by more sophisticated application-level knowledge data, other than simple pheromones value to react, and this will enable more robust and adaptive configuration patterns (e.g., the knowledge network can be used to enforce a more intelligent control over a set of swarm agents, as envisioned in Sect. 4.2). Actual benefits and challenges from knowledge networks will come to the surface as our experience with them advances.

5 Conclusions

The continuous growth in ubiquitous and mobile network connectivity, together with the increasing number of networked computational devices populating our everyday environments, call for a deep rethinking of traditional communication and service architectures. In this paper, we have focused on communication services, and have analyzed the key characteristics and features that a proper innovative component model for the effective development and deployment of autonomic (i.e., self-organizing, self-adaptive, self-healing) communication services will likely have to exhibit.

The results of our analysis can be simply summarized as follows:

- Such new component model—and the associated supporting infrastructures and tools—should be general-purpose, able to enforce autonomic behavior in both the forms of self-adaptation and self-organization, able to handle "situatedness" in complex knowledge environments, and should tolerate scalable forms of dynamic self-aggregation.
- To realize this ambitious vision, we claim that autonomic communications will have to go further with respect to a mere service-oriented architecture for network services. The very nature of ACSs strongly suggests that they will likely inherit many of the features that contributed to the popularity of software agents. From sociality to reactivity, from cooperation based on semantic agent languages to coalition formation algorithms, from knowledge-based reasoning to goal-based planning, future networks of ACSs will likely be a new important branch of multi-agent systems in telecommunications.

It is our hope that the discussion and analysis in this paper can be of some inspiration to find in the fascinating scenarios of autonomic communication and autonomic communication services new impetus for future software agent research.

Acknowledgments Work supported by the European Commission within the Integrated Project CASCADAS (IST-027807), funded under the FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications.

References

1. Ando, Y., Fukazawa, Y., Masutani, O., Iwasaki, H., & Honiden, S. (2005). Performance of pheromone model for predicting traffic congestion. In *5th International Joint Conference on Autonomous Agents and Multi-Agent Systems* (pp. 73–80). Japan, 2005.
2. Arcos, J. L., Esteva, M., Noriega, P., Rodriguez, J. A., & Sierra, C. (2005). Engineering open environments with electronic institutions. *Journal on Engineering Applications of Artificial Intelligence, 18*(2), 191–204.
3. Babaoglu, O., Canright, G., Deutsch, A., Di Caro, G., Ducatelle, F., Gambardella, L., Ganguly, N., Jelasity, M., Montemanni, R., Montresor, A., & Urnes, T. (2006). Design patterns from biology for distributed computing. *ACM Transactions on Autonomous and Adaptive Systems, 1*(1), 26–66.
4. Baresi, L., Baumgarten, M., Mulvenna, M., Nugent, C., Curran, K., & Deussen, P. H. (2006). Towards pervasive supervision for autonomic systems. In *1st IEEE Workshop on Distributed Intelligence Systems* (pp. 365–370). Czech Republic, 2006.
5. Baumgarten, M., Bicocchi, N., Curran, K., Mamei, M., Zambonelli, F., & Mulvenna, M. (2006). Towards self-organizing knowledge networks for smart world infrastructures. In *2nd International Conference on Self-organization in Multi-Agent and Grid Systems*. Germany, 2006.
6. Bellifemine, F., Poggi, A., & Rimassa, G. (2001). JADE—A FIPA2000 compliant agent development environment. In *5th International Conference on Autonomous Agents* (pp. 216–217). Canada, 2001.
7. Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). Swarm intelligence: From natural to artificial systems. USA: Oxford University Press.
8. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M. Ferris, C., & Orchard, D. (2004). Web services architecture. World Wide Web Consortium.

9. Brueckner, S., Parunak, V., & Savit, T. (2004). Universality in multi-agent systems. In *3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems* (pp. 930–937). USA, 2004.
10. Castelli, G., Rosi, A., Mamei, M., Zambonelli, F. (2007). A simple model and infrastructure for context-aware browsing of the world. In *5th IEEE International Conference on Pervasive Computing and Communications* (pp. 229–238). USA, 2007.
11. Contizer, M., Sandholm, V., Ohta, T., & Iwasaki, N. (2005). Coalitional games in open anonymous environments. *International Joint Conference on Artificial Intelligence, 19*, 1668–1669.
12. Di Caro, G., & Dorigo, M. (1998). Ant colonies for adaptive routing in packet switched communication networks. In *5th International Conference on Parallel Problem Solving from Nature, LNCS* (Vol. 1498, pp. 673–682). Springer Verlag.
13. DIET Agents project: http://diet-agents.sourceforge.net/index.html, Accessed March 18, 2008.
14. Dill, S., Kumar, R., Mccurley, K., Rajagopalan, S., Sivakumar, D., & Tomkins, A. (2003). Self-Similarity in the Web. *ACM Transactions on Internet Technology, 2*(3), 205–223.
15. Dobson, S. (2004). Putting meaning into the network: some semantic issues for the design of autonomic communications systems. In *1st IFIP Workshop on Autonomic Communications, LNCS* (Vol. 3457, pp. 207–216). Springer Verlag.
16. Dobson, S., Denaziz, S., Fernandez, A., Gaiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., & Zambonelli, F. (2007). A survey of autonomic communication. *ACM Transactions on Autonomous and Adaptive Systems, 1*(3), 223–259.
17. Estrin, D., Culler, D., Pister, K., & Sukjatme, G. (2002). Connecting the physical world with pervasive networks. *IEEE Pervasive Computing, 1*(1), 59–69.
18. Fisher, K. (1999). Holonic multi-agent systems—theory and applications. In *9th Portuguese Conference on Progress in Artificial Intelligence, LNAI* (Vol. 1695, pp. 34–48). Springer.
19. Giret, A., & Botti, V. (2004). Holons and agents. *Journal of Intelligent Manufacturing, 15*, 645–659, Kluwer Academic Publishers.
20. Haegg, S. (1996). A sentinel approach to fault handling in multi-agent systems. In *2nd Australian Workshop on Distributed Artificial Intelligence* (pp. 181–195). Australia, 1996.
21. He, M., Jennings, N. R., & Leung, H. (2003). On agent-mediated electronic commerce. *IEEE Transactions on Knowledge and Data Engineering, 15*(4), 985–1003.
22. Hightower, J., & Borriello, G. (2001). Location systems for ubiquitous computing. *IEEE Computer, 34*(8), 57–66.
23. Holland, O., & Melhuis, C. (1999). Stigmergy, self-organization and sorting in collective robotics. *Artificial Life, 5*(2), 173–202.
24. Huhns, M., & Singh, M. P. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet Computing, 9*(1), 75–81.
25. IBM (2001). Autonomic computing: IBM's perspective on the state of information technology.
26. Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM, 44*(4), 35–41.
27. Keil, D., & Goldin, D. (2005). Indirect interaction in environments for multi-agent systems. In *2nd International Workshop on Environments for Multi-Agent Systems* (pp. 68–87). The Netherlands.
28. Kephart, J., & Chess, D. M. (2003). The vision of autonomic computing. *IEEE Computer, 36*(1), 41–50.
29. Kraus, S., Shehory, O., & Taase, G. (2003). Coalition Formation with Uncertain Heterogeneous Information. In *2nd International Conference on Autonomous Agents and Multi-Agent Systems* (pp. 1–8). Australia.
30. Mamei, M., Menezes, R., Tolksdorf, R., & Zambonelli, F. (2006). Case Studies for Self-organization in Computer Science. *Journal of Systems Architecture, 52*(8–9), 443–460.
31. Mamei, M., Roli, A., & Zambonelli, F. (2005). Emergence and control of macro spatial structures in perturbed cellular automata, and its implications for pervasive computing systems. *IEEE Transactions on Systems, Man, and Cybernetics, 35*(5), 337–348.
32. Mamei, M., & Zambonelli, F. (2004). Programming pervasive and mobile computing applications with the TOTA middleware. In *2nd IEEE International Conference on Pervasive Computing and Communications* (p. 263). Florida, USA, 2004.
33. Manzalini, A., & Zambonelli, F. (2006). Towards autonomic and situation-aware communication services: The CASCADAS vision. In *1st IEEE Workshop on Distributed Intelligent Systems* (pp. 383–388). Czech Republic, 2006.
34. Marrón, P. J., Minder, D., Lachenmann, A., & Rothermel, K. (2005). TinyCubus: An adaptive cross-layer framework for sensor networks. *Information Technology, 47*(2), 87–97.
35. Martino, S. (1999). A mobile agent approach to intrusion detection. Technical Report, Joint Research Center - Institute for Systems, Informatics and Safety.

36. Nick, J., Foster, I., Kesselmann, C., & Tuecke, S. (2002). The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum.
37. Noriega, P., & Sierra, C. (2002). Electronic institutions: Future trends and challenges. In *Cooperative Information Agents VI, LNCS* (Vol. 2446, pp. 14–17). Springer Verlag.
38. Omicini, A., & Zambonelli, F. (1999). Coordination for internet application development. *Journal of Autonomous Agents and Multi-Agent Systems, 2*(3), 251–269.
39. Parunak, H. V. D. (1997). Go to the ant: Engineering principles from natural multi-agent systems. *Annals of Operations Research, 75*, 69–101.
40. Pentland, A. (2005). Socially aware computation and communication. *IEEE Computer, 38*(3), 33–40.
41. Platon, E., Mamei, M., Sabouret, N., Honiden, S., & Parunak, H. V. D. (2007). Mechanisms for environments in multi-agent systems: Survey and opportunities. *Journal of Autonomous Agents and Multi-Agent Systems, 14*(1), 31–47.
42. Quitadamo, R., Zambonelli, F., & Cabri, G. (2007). The service ecosystem: Dynamic self-aggregation of pervasive communication services. In *1st Workshop on Software Engineering of Pervasive Computing Applications, Systems and Environments (SEPCASE) at ICSE 2007*. Minneapolis, MN, USA, May 2007.
43. Rao, A., & Georgeff, M. (1995). BDI agents: From theory to practice. In *1st International Conference on Multi-Agent Systems* (pp. 312–319). California.
44. Ricci, A., Omicini, A., & Denti, E. (2003). Activity theory as a framework for MAS coordination. *Engineering Societies in the Agents' World III, LNCS* (Vol. 2577, pp. 96–110). Springer Verlag.
45. Scott, J., Hui, P., Crowcroft, J., & Diot, C. (2006). Haggle: A networking architecture designed around mobile users. In *3rd Annual IFIP Conference on Wireless On-demand Network Systems and Services*, Les Menuires (F), January 2006.
46. Sestini, F. (2006). Situated and autonomic communications: An EC FET European initiative. *ACM Computer Communications Review, 36*(2), 14–17.
47. Shehory, O., & Aknine, S. (2006). A feasible and practical coalition formation mechanism leveraging compromise and task relationships. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (pp. 436–439).
48. The Autonomic Communication Forum. http://www.autonomic-communication-forum.org/. Accessed March 18 2008.
49. The CASCADAS Project Web Site. http://www.cascadas-project.org. Accessed March 18, 2008.
50. Tummolini, L., Castelfranchi, C., Ricci, A., Viroli, M., & Omicini, A. (2005). Exhibitionists and voyeurs do it better: A shared environment approach for flexible coordination with tacit messages. *Environments for Multi-Agent Systems, LNAI* (Vol. 3374, pp. 215–231). Springer Verlag.
51. Valckenaers, P., Sauter, J., Sierra, C., & Rodriguez-Aguilar, J. A. (2007) Applications and environments for multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems, 14*(1), 61–85.
52. Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K., & Zambonelli, F. (2007). Infrastructures for the environment of multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems, 14*(1), 49–60.
53. Want, R. (2006). An introduction to RFID technology. *IEEE Pervasive Computing, 5*(1), 25–33.
54. Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first class abstraction in multiagent systems. *Autonomous Agents and Multi-Agent Systems, 14*(1), 5–30.
55. Wooldridge, M. (1997). Agent-based software engineering. *IEEE Proceedings Software Engineering, 144*(1), 26–37.
56. Yolum, P., & Singh, M. (2005). Engineering self-organizing referral networks for trustworthy service selection. *IEEE Transactions on Systems, Man, and Cybernetics, 36*(5), 396–407.
57. Zambonelli, F. (2006). Self-management and the many facets of nonself. *IEEE Intelligent Systems, 21*(2), 50–58.
58. Zambonelli, F., Jennings, N., Wooldridge, M. (2003). Developing multi-agent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology, 12*(3), 317–370.