On Self-adaptation, Self-expression, and Self-awareness in Autonomic Service Component Ensembles

Franco Zambonelli[1], Nicola Bicocchi[2], Giacomo Cabri[2], Letizia Leonardi[2], Mariachiara Puviani[1]
1) Dipartimento di Scienze e Metodi dell'Ingegneria
2) Dipartimento di Ingegneria dell'Informazione
Universita' di Modena e Reggio Emilia, Italy
Email: name.surname@unimore.it

***Abstract*—Software systems operating in open-ended and unpredictable environments have to become autonomic, i.e., capable of dynamically adapting their behavior in response to changing situations. To this end, key research issues include: (i) framing the schemes that can facilitate components (or ensembles of) to exhibit self-adaptive behaviors; (ii) identifying mechanisms to enable components or ensembles to self-express the most suitable adaptation scheme; and (iii) acquiring the proper degree of self-awareness to enable putting in action self-adaptation and self-expression schemes. In this position paper, with the help of a representative case study, we frame and discuss the above issues, survey the state of the art in the area, and sketch the main research challenges that will be faced in the ASCENS project towards the definition of a fully-fledged framework for autonomic services.**

I. Introduction

Current and emerging ICT scenarios increasingly rely on complex distributed software systems for their proper functioning, and they are called to operate in open-ended and unpredictable environments. Accordingly, software systems have to become autonomic in their behavior [1], i.e., capable of dynamically adapt their behavior without human supervision, to respond to changing situations without suffering malfunctioning or degrading quality of service.

The overall ultimate goal of the EU-funded project ASCENS ("Autonomic Service Component Ensembles", *www.ascens-ist.eu*) is to define an innovative framework for the engineering of complex software-intensive service systems, i.e., of all those complex systems that serve some purpose and are operated by software. In particular, in ASCENS, the idea is to define formal models, languages, and software engineering tools, to support the modeling and development of software systems that are at the same time predictable capable of adaptation. This calls for investigating models and mechanisms by which both the individual components and ensembles of components can become *self-adaptive* – i.e., able to properly react on need by self-tuning their internal behavior and/or structure in an autonomic way – and *self-aware* – i.e., able to recognize the situations of their current operational context requiring self-adaptive actions.

Accordingly, one of the specific objectives to be faced by ASCENS, and the specific focus of this paper, is to study, analyze, and experiment with various models, schemes, and mechanisms enabling self-adaptation, at various levels, in service components and in ensembles. These should eventually turn into a sound and uniform set of conceptual and practical tools to guide developers and engineers in the exploitation of such mechanisms, at the level of abstract system modeling, verification, and implementation. In addition, during the analysis of such models and mechanisms, it will be possible to clearly identify the self-awareness requirements for these kinds of systems, i.e., which kind of knowledge is required by the identified adaptation mechanisms.

The results of the ASCENS activity in this area, beside becoming (hopefully) useful tools for designers, will also have scientific relevance *per se*. In fact, they can shed new light into the general issues involved in self-awareness, self-expression and self-adaptation, and in their impact in software development and execution. In particular, a key innovation that will be pursued over the state of the art is related to weakening the tension between individual (component-level) and collective (ensemble-level) self-adaptation, as well as between top-down (engineered by design and/or regulated) and bottom-up (based on emergence and self-organization) approaches to self-adaptation.

II. Framing Adaptation Dimensions

The understanding of how adaptation can be effectively enforced in modern service systems, along with the identification of the associated research challenges, requires framing the concepts associated.

With this regard, we have identified two primary orthogonal dimensions: *where* adaptation can take place and *what* adaptation mechanisms can take place (see Figure 1).

A. Where: Individual vs. Collective Adaptation

In modern software systems, adaptation can take place both via mechanisms integrated in individual components as well as in software component ensembles. In particular:

- *At the level of individual components* adaptation can take place by having components capable of perceiving

	SELF-ADAPTATION	SELF-EXPRESSION
Individual adaptation	Agent architectures (reactive vs. goal-oriented) Autonomic architectures (control loops)	Dynamic change of internal architectural patterns
Collective (Ensemble-level) adaptation	Organizational patterns Swarm intelligence Market-based mechanisms Normative systems	Dynamic change of interaction topology and of control regime

Figure 1. Adaptation Dimensions

the situations around them, and autonomously decide how to tune/modify their internal behavior (or the state of their operational environment) so as to ensure that the component will be able to achieve its objectives (let them be service-level objectives or long-terms goals);

- *At the level of ensembles* adaptation can take place by having components dynamically engage in interactions protocols, enabling the dynamic change of the control regime for the ensemble execution.

We are perfectly aware that the study of both individual and collective adaptation mechanisms has a long history, e.g., in the area of agent-based and multi-agent systems [2] and in the area of autonomic systems [1].

Individual adaptation is a very important thread of research since the early years of intelligent agents [2] and of reflective computing [3], and several architectures and mechanisms to enable adaptation have been proposed so far. Most recently, the autonomic computing approach has adopted a similar vision [1], which reflects in a number of proposal for self-adaptive (and self-aware) components capable of adapting their internal behavior, an approach which is to most extent also shared by autonomic communications researches [4]. For instance, the ACE component model proposed in the context of the CASCADAS project defines a very clean architecture enabling service components to dynamically adapt their behavior based on their context [5]. Yet, the large amount of different mechanisms that are being proposed is a clear proof that the research still needs further progresses, to come assessing a limited set of versatile and widely accepted models and mechanisms.

For collective adaptation, the idea of dynamically adapting the behavior of a set of interacting software components, in order to meet new or unexpected requirements or contingencies, is not new [6], [7], and has been a very hot research topics over the last ten years for several research communities. Yet, also in this case, it is widely recognized that the number of open issues in the area is still relevant [8].

On the one hand, adaptive negotiation mechanisms and market-oriented mechanisms have been extensively studied [2]. With this regard, we consider of particular relevance the lessons of normative multi-agent systems, since the proper functioning and adaptation of a complex system made up of autonomous components may require – in whatever area – the existence of "norms" that collectively regulate the activities of the components [6], [9], possibly enacted via specific components of the ensemble [10]. However, the definition of proper tools and mechanisms to engineer and control such kind of systems is still missing.

On the other hand, in the past few years, a large number of proposals for algorithms that can achieve adaptation in a distributed systems via decentralized self-organization have been proposed (see [11] for an in-depth survey). However, most of these solutions apply to specific problem instances, and do not propose themselves as general-purpose solutions. Despite the existence of some proposal aiming at more generality (e.g., the TOTA middleware [12]), the issue of systematically exploiting self-organization as a mechanism for achieving adaptation in large-scale heterogeneous software systems is still to be faced.

In summary, for both individual and collective adaptation, what we think is still missing are uniform models and tools supporting the design of complex adaptive systems. In particular, a key research challenge is to devise means by which to understand, for a given system and for its specific objectives/goals, how to properly balance individual and collective adaptation. That is, whether to encode goals and adaptation mechanisms at the level of individual components, at the level of collectives, or at both level. And, in the latter case, evaluating how to properly balance in a system the trade-off between individual and collective adaptation, and possibly to smooth the tension between the two.

B. What: Self-Adaptation vs. Self-Expression

Adaptation actions imply activating some mechanisms to modify "something" in the behavior of a component or ensemble. What mechanisms a component or ensemble can put in action is very important to understand the extent of their adaptation capabilities. With this regard, we distinguish between two main classes of mechanisms (that are sometime referred to in the literature as mechanisms of *weak* and *strong* adaptation, respectively [13]):

- *Self-adaptation mechanisms* concern the possibility of modifying parameters of components or ensembles, or at selecting, among the available ones, the most proper set of actions/behaviors (for components) or interaction protocols (for ensembles). In other words, components and ensembles react to those contingencies that require adaptation by exploiting their current abilities.
- *Self-expression mechanisms* concern the possibility of radically modifying at run-time the structure of components and ensembles. For components, this could imply internalizing capabilities previously unavailable to them (e.g., a new function or new sensors) as well as changing their internal architecture (e.g., switching from being reactive entities to goal-oriented ones). For ensembles, this could imply a re-structuring in terms

of topology (e.g., switching from a hierarchy to a collective of peers) or of control regime for interactions (e.g., switching from being a collective decision-making ensemble to a competitive market-based one).

While self-adaptation mechanisms have been the subject of a large body of research work, self-expression mechanisms are far from being fully understood, and so it is the issue of providing useful guidelines for developers to express (and let software self-express at run-time) among the most suitable mechanisms.

Self-expression has been a key issue in the area of programming languages, related to defining languages capable of self-expressing at the most appropriate paradigm (procedural vs. declarative) depending on the needs that have arisen, thus smoothing the tension between procedural and declarative programming paradigms.

In the context of individual components for self-adaptive systems, such tension translates in the dichotomy between architectures of simple reactive components (i.e., obeying in a reactive way with simple rule-action mechanisms, typically based on declarative knowledge, to external stimuli) and inherently self-adaptive components (i.e., goal-oriented components capable of procedurally digesting knowledge). Enabling a component to chose at run-time which structure to adopt is definitely challenging, and definitely very important for smoothing the tension between goal-oriented and reactive components. Again, the issue so far is mostly under-investigated.

Coming to the specific context of component ensembles, a very challenging issue that will have to be faced in enabling dynamic self-expression is that of promoting and supporting the possibility of dynamically switching between top-down and bottom-up regimes of control. As we have already stated, a part of the recent researches on self-adaptation in software systems has looked for inspiration in the area of natural and biological self-organizing systems, with the aim of reproducing the emergent adaptation capabilities of such systems [11]. At the same time, another research thread has focused on integrating adaptation in software systems according to the most assessed approaches of software engineering, that is, by explicitly encoding adaptation in system design [8].

It is very challenging to smooth the tension between the two approaches, and to devise means by which to make possible to dynamically self-express the most appropriate regime of control. Working towards such goal also requires understanding the implications at the global level for a system whose components and ensembles dynamically change their structure and behavior. In particular, it requires models and methods to deal with unpredictability and emergent behaviors, i.e., methods to engineer and control emergence.

The issue of engineering and control emergence is definitely under-investigated in the ICT arena. Researches on complex systems already recognize that, beside understanding complex behaviors, the next challenge is to find methods and tools to dominate and control them [14]. A few researches in software engineering and distributed systems explicitly address this issue, and mostly at the level of simple simulations for multi-agent systems [15] or cellular automata [16]. Yet, a general understanding of how to identify and control emergent behaviors in complex software systems is still to be reached, and we think that the work on regulated norm-based multi-agent systems [9] can be an effective starting point.

III. A Case Study

Let's sketch a simple example to clarify the concepts presented and to show the need of proper conceptual and practical tools to tackle the challenges we have identified.

Consider a group of autonomous robots – each to be considered as a service component – devoted to collaboratively explore and map an unknown environment. Here, adaptivity implies the capability of going on with the exploration and mapping regardless the contingencies occurring in the environment or in the robots themselves.

A fist key issue suddenly arises: exploring the environment has to be a shared goal for all individual robots and for the collective of robots. So, from the designer viewpoint, it is possible to chose among many apparently functionally-equivalent, solutions. Let us just detail three of them:

- *Leader*: Let elect one of the robot as leader of the group, and have it be a goal-oriented component embedding the global exploration goal, while keeping the other robots as simple reactive components. The leader, to accomplish its goal, will direct (via proper signals) the activities of the reactive robots to coordinate the overall activities of the group in a centralized way.
- *Peers*: As for the individual robots, let have each robot be a goal-oriented one, embedding, in its goal-oriented activities, the capability of perceiving the status of the environment and of adaptively selecting the best plan of actions/behavior to enable it reach its goal. As for the collective, let have robots directly exchange information about the explored environment and coordinate with each other, in order to avoiding exploring areas already visited by other robots.
- *Swarm*: Let have all the robots in the group (as sophisticated as they can be) acting as simple reactive components, e.g., as a swarm of simple bees that randomly explore the environment. The only coordination occurring between robots is in the form of simple exchange of local signals (e.g., virtual pheromones): robots, while exploring, deposit pheromone trails, and their random movements have a preferred direction that tends to move them away from existing trails. It has been shown that such mechanisms can lead to an effective exploration and mapping [11].

Beside mere considerations related to computational and communication efficiency, the designer has to chose one of the above three solutions taking into consideration the adaptation capabilities that the solution can express in response to expected contingencies.

The Leader solution appears very effective. In fact, the leader holds complete information about what the others are doing and have discovered so far, and so it can effectively plan alternate actions in the presence of unexpected situations occurring in the environment (e.g., a corridor becoming obstructed by a robot that has ran out of batteries). However, as any centralized solution, it is very fragile, and exploration suddenly stops if the leader for any reason gets broken. Also, when adopting the Leader solution, one should also evaluate whether, instead of having the non-leaders be simple reactive robots, it can be better to leave them with some limited goal-oriented capabilities, in order to discharge the leader from having to handle very specific individual adaptation needs (e.g., what sensors and engines should a robot activate to effectively pass across a specific portion of the environment?). That is (see Section II.A), there is the need of finding the proper trade-off between local vs. global goals and between individual vs. collective adaptation actions. Or, which is the same, there is need for models, tools, and guidelines, to support such decision.

The Peers solution is a bit more complex. It requires spreading knowledge about the global goal across all robots, and to enable local adaptation capabilities in each robot to achieve their goals in autonomy, other than adaptive coordination capabilities to coordinate their actions. Yet, it ensures that the exploration goal can be achieved even when a relevant number of robots die. However, if the environment is larger than expected, or in the presence of a large number of robots, it can become difficult (other than costly) for the group of robots to collectively converge to a shared plan of actions towards the global goal. Also, any time some contingencies occur (e.g., a corridor obstructed or some robot do no longer respond), there is need of re-planning in a collective way. Again, the possibility of finding a trade-off between local goal-oriented adaptivity in each robot and global distributed decision making arises.

The Swarm solution, to be effective, requires a large number of robots, but in this case it can represent a very simple solutions, and also a highly adaptable one. The Swarm approach, in fact, can effectively tolerate the death of several robots without the exploration capabilities being undermined. Also, the capabilities of effectively completing the exploration are not affected by changes and contingencies that can occur in the environment during the exploration itself.

So, let us now assume that the designer knows there will be the possibility of having a very large and dense group of robots. Then, it can decide to adopt the Swarm solution without doubts. However, because of very peculiar conditions that the robots find in the environment (e.g. holes in which they get trapped, or wild animals that destroy them), it can happen that the swarm becomes sparsely populated. At this point, while the few remaining robots in the swarm could in theory continue with the exploration, this becomes highly ineffective.

In such case, there are no simple adaptation mechanisms that the swarm can put at work to improve the situation. Rather, the remaining robots in the swarm have to dynamically self-express a totally different working regime. For instance, they could decide to switch to the Leader regime: one of the robots in the swarms takes the role of coordinator to orchestrate the activities of the group, while the other follows the directives of the leader. Beside devising mechanisms by which such switch can be made possible, we emphasize that such a change of collective behavior also implies individual robots to dynamically self-express a change: the leader, in particular, from a simple reactive components, should become a goal-oriented autonomous components, also should internalizing the necessary behaviors and knowledge to control its behavior, and a control loop in which to more properly analyze and digest the information on its surroundings. That is, as discussed in Section II.B, self-expression at the level of individual components and at the level of ensembles can hardly be tackled independently.

IV. Self-awareness Issues

So far, we have not discussed in detail the issue of self-awareness. Yet, for any adaptation action to take place, the component or ensemble must become aware of the existence of situations (inside them or in the environment around them) requiring it.

In our opinion, the key issues in awareness are not related to information availability: in whatever software system, you can easily get access to whatever sensor to get any information you need. Nor, from the perspective of our analysis, it is an issue that of properly inferring the necessary high-level knowledge from the available sensors and modeling it, since there are specific threads of research in ASCENS tacking this problem [17]. Rather, from our perspective on engineering self-adaptation and self-expression, and assuming that proper solutions for knowledge inference and modeling will be made available somehow, the key issues relate to:

- Understanding what minimal amount of information is to be made available, at the level of individual components, for enabling their adaptation capabilities;
- Understanding how knowledge must be shared at the level of ensemble for enabling proper self-adaptation and self-expression actions;
- Understanding the mechanisms by which, upon self-expressing specific adaptation patterns, components and ensembles can correspondingly modify the amount of information that they have to digest.

Let us exemplify why each of these issues is challenging with reference to the robotics case study.

- In the Leader solution, understanding what local goal-oriented capabilities to delegate to robots implies necessarily identifying what knowledge they should made available to reach the goal. Yet, depending on the goal locally delegated to each robot, the amount of knowledge to be digested by each of them can dramatically change: from short-term memory of the local shape of the environment to long-term memory of the global shape of the environment; from very simple information related to simple movements in the environment (according to the directives of the leader) to more sophisticated environmental knowledge to plan an itinerary of movements.
- In the Peers solution, the collective agreement that must be shared among robots raises a fundamental question. Since such agreement should of course rely on a shared knowledge of the overall situation of the exploration activity and of the overall status of the group of robots, where has such shared knowledge to reside? It is possible to think both at complex interaction protocols that let each and every robot in the group to eventually acquire such global knowledge. Or, alternatively, it is possible to assume that there are means (e.g., a shared blackboard) by which such global knowledge can be easily available to each robot.
- In the Swarm solution, components need nearly no local knowledge about the environment or about themselves to properly acting in an adaptive way, nor the robots need to share some global knowledge. However, when switching from the Swarm solution to the Leader one, knowledge about the environment and about the state and position of other robots should be materialized by the leader. Also, the simple action for a component or ensemble to self-express a different structure may require much more than being aware of what it is and what is around it: it requires knowing that it could purposefully become something different.

In summary, a key lesson that we have learnt from our early activity in ASCENS is that considering self-awareness in a general-way can be a rather useless exercise. In fact, self-awareness can be expressed at very different degrees, and different software solutions imply different expressions of self-awareness and very different awareness levels. Thus, self-awareness has to be contextualized to specific adaptation needs, and can require not simply the awareness of "what I am and what is happening in the world" but also the second-order awareness of "what I could become and how the world could change accordingly".

V. Conclusions and Next Steps

In this position paper, we have framed the key concepts and challenges related to defining models, tools, and guidelines, for the engineering of complex autonomic software systems, as they will be faced in the context of the ASCENS project. In particular, ASCENS will try to show that a system can dynamically express diverse classes of self-adaptation mechanisms without conflicts, and that it can be made able to self-evaluate whether to switch from top-down forms of self-adaptation to bottom-up ones.

To reach these goals, the ASCENS research activities on self-adaptation and self-expression will be organized as follows.

First, we are currently analyzing the key patterns of self-adaptation studied so far in the literature. At the level of *individual components*, the starting technical points have been recent research results in the area of autonomic self-adaptive components [18] and of adaptive goal-oriented agents [2]. Soft constraint programming [19] will be explored as a general-purpose tool to assert/retract relevant facts and rules that can drive individual self-adaptation. At the level of *ensembles*, the starting technical points of this task has been recent research results in the area of self-adaptive architectures [8] and of self-organizing agent systems [20]. In addition, recent research results in the area of adaptive and norm-based multi-agent systems [9], [6] will be of help as inspiring mechanisms. As for the case of individual self-adaptation, constraint and soft constraint logic programming will be most likely adopted as tools, as they can be well suited to express normative rules driving the behavior of ensembles and their adaptation constraints. The key result of this activity will be a catalogue of self-adaptive patterns, properly supported by the experimentation of such patterns in various scenarios, and eventually acting as of methodological guidelines [21].

Second, for each of the studied patterns, we intend to carefully analyze the associated requirements of self-awareness. Each of the patterns will be carefully investigated with this regard, also with the help of the above mentioned experimental activities and of several application scenarios. The starting point for this activity will be recent results in the area of context-modeling for complex adaptive distributed service systems [22], [23]. The key results of this activity will be in the identification of a set of guidelines to let designers understand what amount of information and knowledge (and, consequently, which sensors and knowledge inference tools, respectively) has to be made available depending on the characteristics of the application scenarios and on the adopted self-adaptation and self-expression mechanisms.

Third, we intend to study how to exploit self-expression, within a system and depending on needs, to enforce both bottom-up and top-down approaches to self-adaptation. In particular, a key goal of this task is to identify the potential problems of unpredictable behaviors that can emerge in large ensembles, due to the variously expressed self-adaptation patterns, classify and categorize them, and eventually identify methods and tools by which such emergent behav-

iors can be properly controlled and directed. The starting technical points of this task will be recent research results related to controlling emergent behavior in complex multi-agent and robotics systems [15] and in norm-based multi-agent systems [9]. The key results of this activity will be in the identification of mechanisms to overcome the dichotomy between bottom-up and top-down self-adaptation, in the identification of the potential problems of emergent behaviors in complex self-adaptive systems, and in the definition of guidelines and tools to control and direct such emergent behaviors.

The above ASCENS activities can have a disruptive character and can contribute collapsing into a single one different researches that, as of now, are mostly non-overlapping and non-communicating.

Acknowledgements: Work supported by the ASCENS project (EU FP7-FET, Contract No. 257414).

References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[2] N. R. Jennings, "An agent-based approach for building complex software systems," *Communications of the ACM*, vol. 44, no. 4, pp. 35–41, April 2001.

[3] T. Watanabe and A. Yonezawa, "Reflection in an object-oriented concurrent language," in *ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*. ACM Press, 1988, pp. 306–315.

[4] S. Dobson and al., "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 1, no. 2, pp. 223–259, 2006.

[5] A. Manzalini and F. Zambonelli, "Towards autonomic and situation-aware communication services: the cascadas vision," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, Prague (CZ), 2006, pp. 383–388.

[6] N. A. M. Tinnemeier, M. Dastani, and J.-J. C. Meyer, "Roles and norms for programming agent organizations," in *8th International Conference on Autonomous Agents and Multiagent Systems*, 2009, pp. 121–128.

[7] J. Andersson and al., "Reflecting on self-adaptive software systems," *Workshop on Software Engineering for Adaptive and Self-Managing Systems*, vol. 0, pp. 38–47, 2009.

[8] B. H. C. Cheng and al., "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*, 2009, pp. 1–26.

[9] A. García-Camino and al., "Constraint rule-based programming of norms for electronic institutions," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 1, pp. 186–217, 2009.

[10] P. Noriega, *Agent-mediated Auctions: The Fishmarket Metaphor*. Barcelona (E): Ph.D Thesis, Universitat Autonoma de Barcelona, 1997.

[11] M. Mamei and al., "Case studies for self-organization in computer science," *Journal of Systems Architecture*, vol. 52, no. 8-9, pp. 443–460, 2006.

[12] M. Mamei and F. Zambonelli, "Programming pervasive and mobile computing applications: The tota approach," *ACM Transactions on Software Engineering and Methodology*, vol. 18, no. 4, 2009.

[13] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 4, no. 2, 2009.

[14] J. Ottino, "Engineering complex systems," *Nature*, vol. 427, p. 399, 2004.

[15] A. Brintrup and al., "Distributed control of emergence: Lying agents in particles swarms and ant colonies," in *3rd International Conference on Self-adaptive and Self-organizing Systems*, San Francisco (CA), September 2009.

[16] M. Mamei, A. Roli, and 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, Part A*, vol. 35, no. 3, pp. 337–348, 2005.

[17] E. Vassev and al., "Requirements and initial model for knowlang: a language for knowledge representation in autonomic service-component ensembles," in *Fourth International C* Conference on Computer Science & Software Engineering*, Montreal (CA), 2011, pp. 35–42.

[18] L. Baresi, A. Di Ferdinando, A. Manzalini, and F. Zambonelli, "The cascadas framework for autonomic communications," in *Autonomic Communications*. Berlin, D: Springer, 2009.

[19] S. Bistarelli and F. Rossi, "Semiring-based contstraint logic programming: syntax and semantics," *ACM Transactions on Programming Languages and Systems*, vol. 23, no. 1, pp. 1–29, 2001.

[20] P-Snyder, R. Greenstadt, and G. Valetto, "Myconet: a fungi-inspired model for superpeer-based overlay topologies," in *3rd International Conference on Self-adaptive and Self-organizing Systems*, San Francisco (CA), 2009.

[21] G. Cabri, M. Puviani, and F. Zambonelli, "Towards ataxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles," in *2011 Annual Conference on Collaborative Technologies and Systems*, Philadelphia (USA), May 2011, pp. 306–315.

[22] N. Bicocchi and al., "Self-organized data ecologies for pervasive situation-aware services: The knowledge networks approach," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 40, no. 4, pp. 789–802, 2010.

[23] K. Rasch and al., "Context-driven personalized service discovery in pervasive environments," *World Wide Web*, vol. 14, no. 4, pp. 295–319, 2011.