Towards a Taxonomy of Adaptive Agent-based Collaboration Patterns for Autonomic Service Ensembles

Giacomo Cabri
DII, University of Modena and Reggio Emilia, Italy
giacomo.cabri@unimore.it

Mariachiara Puviani
DISMI, University of Modena and Reggio Emilia, Italy
mariachaira.puviani@unimore.it

Franco Zambonelli
DISMI, University of Modena and Reggio Emilia, Italy
franco.zambonelli@unimore.it

ABSTRACT

Services are increasingly becoming the building block of today's distributed systems. However, to support the development of robust complex applications made up of ensembles of cooperating service components and to promote autonomic features, adaptive collaboration patterns among components have to be enforced. In this paper, we introduce a taxonomy of adaptive agent-based collaboration patterns, for their analysis and exploitation in the area of autonomic service ensembles. The preliminary proposed taxonomy has two main advantages: (i) it enables the reuse of existing experience from the agents' world in the area of autonomic service systems, and (ii) it can provide useful suggestions to designer for the choice of the most suitable patterns.

KEYWORDS: Self-adaptation, collaboration, patterns, agent.

1. INTRODUCTION

Designers and developers consider *service components* as the building blocks that can be put together in order to achieve the system-level objectives of distributed systems [19]. Service components, as components in general, have to be well-defined and to expose a clear interface, in order to be reusable, testable and, more importantly, easily composable into ensembles. In this context, two aspects must be taken into consideration: *(i)* the characterization of the service components themselves, in terms of internal organization, and *(ii)* the characterization of their means of composition into ensembles. To describe service components and service components ensembles, and beside the (not less important) adoption of formal modeling approaches, the software engineering practice suggests the adoption of descriptive *design patterns* [17]. In general terms, a design pattern is a general reusable solution to a problem in software design. That is, a semi-formal description or template about how to solve a problem, which can be used in many different situations. Patterns can be characterized in terms of simple attributes like name, context, problem, solution, example and design rationale [4]. But how representing the patterns is not the key point treated here.

The key questions that motivate this paper and our current research, in general relate to the issues of *(i)* effectively enforcing *autonomic* features (i.e., self-management, self-configuration, and self-optimization) in modern service systems, which requires for the system itself to be *adaptive*, and *(ii)* identifying suitable patterns, both at the level of individual service components and at the level of service components ensembles, supporting adaptability. With regard to the first issue, it is a matter of fact that modern software systems are in most of the cases conceived to operated in highly-dynamic and open-ended operational environments [32]. Indeed, the search for novel models and tools to facilitate the design and development of autonomic, self-adaptive, complex software system, is the key object of the ASCENS project [1], in which our current research situates. Unfortunately, the typically static and predefined orchestration patterns of service-oriented architectures can hardly apply, and adaptive collaboration patterns among service component have to be exploited [26], so as to make it possible for service ensembles to exhibit autonomic features [22]. As a consequence, and here we come to the second issue, novel dynamic interaction patterns of collaboration supporting autonomic behavior have to be promoted and supported. This also necessarily calls for the individual components to adopt adaptive architectural patterns so as to support such adaptive collaboration patterns. Despite the large amount of research performed in the area of adaptive systems and of adaptive collaboration patterns, only a few and incomplete tools or conceptual frameworks are available

[1] Autonomic Service-Components ENSembles, *http://www.ascensist.eu*

to facilitate software designers in understanding how and when to exploit specific adaptation patterns to achieve specific goals [4, 10], and these definitely miss in connecting with the area of agent-based computing and multi-agent systems [20, 44], which could represent instead a fertile source of inspiration for the engineering of modern autonomic service systems. More in particular, the contributions of this paper are:

- To show that the founding concepts and lessons of agent-based computing are of primary importance for the study of adaptive collaboration patterns in autonomic service component ensembles.
- To sketch a (preliminary and not ultimate) taxonomy of collaboration patterns, to be used in the context of the design and development of autonomic service systems. A key point for our taxonomy, missing in related work, is relating the architectural patterns of individual components with the adaptive collaboration patterns that they can (or not) support.
- To present, without the ambition of being exhaustive, some exemplary agent-based collaboration patterns, properly framed within our taxonomy.
- To discuss the potential advantages of our taxonomy and identify avenues for further research.

In the rest of the paper we first introduce the agent paradigm and its relevance for modern service systems (Section 2). Second, we present the proposed taxonomy (Section 3) along with some examples of classified patterns and a discussion of the potential advantages. Then, we overview related work in the area (Section 4) and conclude the paper (Section 5).

2. THE AGENT PARADIGM

Software *agents* are autonomous entities that carry out tasks on behalf of users. Among many definitions of software agents, nearly all of them converge on qualifying *autonomy* – i.e., the capability of acting in autonomy rather then simply upon request – as the key peculiar feature of the agents in comparison with other computational entities. One of the most exploited definitions is the one reported by M. Wooldridge in [42]. It characterizes an agent as a computational entity that is situated in some environment, and which is capable of autonomous actions in this environment in order to meet its design objectives. To this end, an agent can exhibit three well-defined features: *reactivity*, *proactivity* (strictly related to *goal-orientedness*) and *sociality*. These three features are detailed in the following, and are the ones we take into consideration for our analysis of adaptive collaboration patterns. The *reactivity* feature of the agents derives from the fact that they are computational entities not isolated but included in an environment, which they are able to perceive. So, they can "react" to events/changes that occur in the environment in order to satisfy their design objectives. The reactivity feature alone leads to rather "passive" agents, able to act only in response to external changes. Instead, agents are more *proactive* entities, since they are able to take initiatives towards the satisfaction of specific internal design objectives. In other words, agents have goal-oriented behaviors, which do everything possible in order to achieve the goal of the agents even trying different ways (called "plans" in the agent terminology). The *sociality*, feature is the capability of interacting, communicating and coordinating among agents, i.e., in the context of a *multi-agent system* or *agent society* [13]. As an agent is not isolated from the environment where it lives, at the same time it is a "social" entity that may be in need to interact with other agents, either because this is the only way to accomplish its goals or simply because the other agents happen to live in the same environment. Thus, there are different means of expressing agent sociality: by implicit interactions of agents that insist on the same environment (environment-mediated interactions), or by explicit communication acts between agents. In the latter case, interaction can take the form of *cooperation* or *competition*. Agents can interact to collaborate if they belong to the same user or application, otherwise they can compete or negotiate for (limited) resources.

2.1. Connection with Autonomic Service Systems

Considering modern complex service-based systems, the identified features of agents and MASs are fundamental to promote autonomic behavior in services and service ensembles, via proper adaptive collaboration patterns. First of all, for a service component to adapt its behavior to the dynamic and possibly unexpected changes that occur around it, the component must be able to sens and *react* to such changes, and modify its behavior (i.e., select proper actions) accordingly. This directly connects with the *reactivity* feature of agents. This is a key base feature required by any service component to be able to exhibit at least some minimal form of adaptability, a feature that then somehow reflects at the level of ensembles, if all its service components are reactive. Of course, beside basic reactivity, which is often not enough to face complex situations in an autonomous way, one can think at arming service components with the agent-based feature of *goal-orientedness*. Again, this can be applied both to the level of individual service components, which have their own goals to achieve, and to ensembles of components, where common goals can be shared by all the components of an ensemble. This means that the components must have a description of the goal and the needed logic to achieve it, which must be adaptive because of the dynamism, unpredictability and heterogeneity of today's complex service systems. In any case, for a component ensemble to be adaptive, and thus exhibit autonomic behavior, it may

not be enough to have its individual components adaptive (whether reactive or goal-oriented). Rather, its components must be able to interact in flexible ways with each other, and possibly to determine at run-time the outcome of these interactions to make sure that the global goals are achieved independently of specific dynamic situations. This suggests that collaboration patterns among components must be adaptive in themselves, as it happens in agent *societies*. And, as in agent societies, sociality can assume the form of both collaboration and competition, the latter being seen as an alternative way to adaptively reach some common goal (e.g., dynamically assigning resources in a cost effective way).

All the above considerations clearly suggest that traditional Service Oriented Architectures (SOA) can hardly act as the basis for the development of autonomic service systems. In [9], we already proposed a detailed comparison between the agent paradigm and SOA, exploiting a case study in the field of territorial emergency to point out the strengths and the weaknesses of the two approaches. Despite the advantages of SOA in terms of robustness, simplicity and industrial support, in dynamic scenarios classical SOAs fall shorts in exhibiting the necessary adaptability. Agent-based design, instead, allows designers to deal with the complexity of the problem domain at an abstraction level higher than the functional problem decomposition typical of SOA. In addition, SOA results to be too rigid and it can hardly exhibit adaptive features, because its service components are passive entities and their interactions is statically defined (for instance, by the BPEL engine). Summarizing, we state that the agents' features are strictly connected to the possibility, in service systems, to achieve autonomic behavior via adaptive collaboration. Accordingly, in the following we start from the lessons and our experience from the agent paradigm to propose our taxonomy of collaboration patterns for adaptive service components.

3. PATTERNS TAXONOMY

A service component, as such, has some intrinsic features. It is recommended that a component, like an agent, is autonomous and can react to environmental changes; that is in some way social; and that has some goals that guide its acting (is goal-oriented). Then, to make a service component collaborate, it can be useful to classify what we call adaptive collaboration patterns that help components to interact one to each other, and at the same time to be adaptive in the collaboration. It is important to recall that a pattern can be defined as adaptive, when its decisions and the interactions outcome, are defined at run time, on the base of the current status of the entire system (environment, resources, interactions with other components, etc). In classifying collaboration patterns, it is not possible to forget the internal architecture of the single component. So, it is not possible to classify collaboration patterns alone, but we need to take into account the patterns that make a single component adaptive. Therefore, in the proposed taxonomy, we have to integrate also a classification of adaptive patterns that work on the single component and that describe its internal behavior. Adaptation is possible with the explicit or implicit presence of a *feedback loop*, which can be internal or external to the components itself. It can be defined as the part of the system that allows for feedback and self-correction and that adjusts its behavior according to the changes in the systems. So feedback loop are considered essential for understanding not only the pattern of adaptation and collaboration, but also the types of adaptive system. The different presented patterns differ depending on the kind of loop they adopt to make the achievement of the (component or ensemble) goals adaptive. In the next subsections we first describe a classification of adaptive patterns to describe each single service component (Subsection 3.1, and then we try to find out some macro areas of adaptive collaboration patters (Subsection 3.2). Both these classifications will help in building an integrated taxonomy of adaptive collaboration patters.

3.1. Adaptive Patterns for Service Components

An example of adaptive pattern is the one that describes a "reactive" component. This component is capable to modify its behavior in reaction to an external event. Components that adopt this pattern are sensitive to events that happen in its environment, on the base of such events, adapt their behavior and are able to interact with the environment itself. In this pattern, any general feedback loop is not present, so here adaptability can emerge only from the interactions with the environment, as we can see in Figure 1. An example of component that use this pattern is the one described one based in the Shaw's control architecture [27] that react to the environment changes making its sensors sense both the executing system and its operating environment, and then converting observations into modeled value. Another adaptive pattern to describe the single component presents "internal feedback loops", like patterns to describe goal-oriented agents. The component is not limited to react to external events, but it actively tries to adapt its behavior, even without waiting for external stimuli. So adaptation and awareness are more explicit inside the component (see Figure 2). This kind of adaptive pattern, when we consider the single component, but at the component ensemble, has to be executed inside collaboration schemes (e.g. patterns) that support argumentation and negotiation, because it will be necessary to integrate the internal goals of the component, with the ensemble goals.

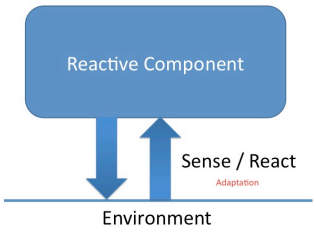

Figure 1. Diagram of a Reactive Pattern

Figure 2. Diagram of Pattern Based on "Internal Feedback Loop"

Here an example of component that uses this pattern in Jadex [7] that supports cognitive agents by exploiting the BDI model. Then other examples are the goal-oriented systems, like robots [23]. Another example of architecture that describes the component using this kind of patter can be the Rainbow architecture [11]. Rainbow monitors the run-time properties of the component in the system-layer through an abstract model maintained by the model manager in its architecture layer (internal loop). Finally, we have to consider adaptive patters with "external feedback loop", that has an external control system at the end of the loop itself (Figure 3). It can be considered the most dynamic adaptive pattern, that seems to be the merge of the previous two: the single service component has not internal feedback loop (like the latter pattern we described), but then it has an explicit control from outside (environment or other components), that close the component feedback loop. This kind of adaptive pattern, at the collaboration patterns level, has to take into account not only the collaboration between components, but we find out that there is a new meta-level of collaboration between external control entities that end each component's feedback loop. Here an example of component that uses this pattern is the one based on the *Autonomic Computing* paradigm, called MAPE-K [22]. It is a specification of a generic "autonomic feedback loop": the feedback loop is realized using an external autonomic manager. Another example of the use of this patter are FORMS (FOrmal Reference Model for Self.Adaptation), which can be viewed as an extension of the MAPE-K model because the basic component of FORMS can be easily mapped on the one of the autonomic computing model [41].

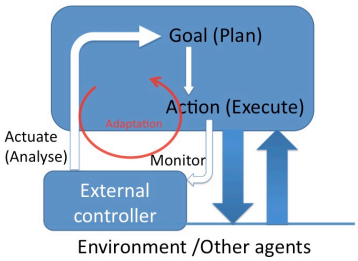

Figure 3. Diagram of Pattern Based on "External Feedback Loop"

3.2. Collaboration Patterns

Here we analyze the collaboration patterns, which are the more connected with the sociality feature of agents. It can be useful to classify these patterns with the aid of some features, like how much the pattern's goals are implicit or explicit; or how much the feedback (or control) loops that manage adaptation are implicit or explicit. For example, considering patterns based on "swarm intelligence connected with the environment" [6], here an explicit representation of the system goals is not possible: the collaborative and adaptive behavior is emerging at runtime. The internal goal of each single component (e.g. swarm, ant) is explicit, but it is not explicit in term of collaboration with components. The same happens for feedback loops; in fact they are implicit inside the sense-react-act cycle of each component – considered as a reactive component – which acts inside the environment. An example of the use of this pattern is Swarm Bot [35], a system composed of a swarm of s-bots capable to self-assemble and self-organize to adapt to its environment. The s-bots have connectors around their body and can connect to other s-bots to create physical structures. The system as a whole can dynamically self-assemble into different structures to perform certain tasks, and then split into s-bots to perform other tasks. Then we have to consider collaboration pattern based on "negotiation" [16], like contract net [37] or argumentation [31]. Here the ensemble goals are well defined and are explicitly represented inside components' interactions, and the representation of goals is also internal of the single component itself. However the feedback loop is implicit inside the messages sent between components. This kind of collaboration pattern can be applied with components that have a pattern of internal feedback loop, but also with components that have a pattern of external feedback loop, in the latter case collaboration (or better negotiation) oriented to adaptability can be viewed at controller level.
A system that uses this collaboration pattern is Hermes [21] that augments the classical decision making approaches by supporting argumentative discourse among decision makers. The system can be used for distributed,

Table 1. Input Data

Classes of patterns	Swarm intelligence	Negotiation	Competition	Electronic institutions
Reactive	Reactive swarm intelligence			Reactive norm-based
Internal feedback loop		Goal-oriented negotiation	Goal-oriented competition	
External feedback l.		Controlled negotiation	Controlled competition	

asynchronous collaboration, allowing users to surpass the requirements of being in the same place and working at the same time. Then, a collaboration pattern can be the one based on "competition" [2]. It can be strange to think at competition in term of collaboration, but this kind of pattern is based on auctions, which can be used also in collaboration systems, not only in competitive ones. Auctions are based on economic theories that start form negotiation mechanism to reach the global equilibrium in an adaptive way. Here it is interesting to note that the feedback loop is implicit, as the global goal. Each component has a specific goal, and in the collaboration pattern, the global goal has to emerge as the global equilibrium into an economic system. There are a lot of algorithms that describe competitions in auctions, like the ones described in [2] and [29]. Anther example of systems implementing one of these algorithms is the Michigan Internet AuctionBot [43], that is an auction server that supports both software and human agents. The server manages many simultaneous auctions by separating the interface from the core auction procedures. The last classified collaboration pattern is the one based on "interaction in electronic institutions" [15]. What makes possible distinguish electronic institutions from other systems is that the explicit presence of norms (i.e. they are also called norm-based systems) can be considered the representation of an explicit control loop: control rules at a global level also act as a feedback on single components. An example of the development of this collaboration pattern is the Harmonia framework [39], which unifies notions of norm, rule, procedure and policy. All these classified collaborations patterns, represent only the macro areas of the taxonomy, that is now only a preliminary work, that will be better refined in the future.

3.3. Taxonomy

Starting from the previous analysis of adaptive and collaboration patterns, in this subsection we propose a preliminary taxonomy. Figure 4 reports the possible connections between each component pattern and the collaboration patterns. We point out that only some specific adaptive patterns for single components can be associated to collaboration patterns, and viceversa. For instance, the reactive adaptive pattern matches the environment mediated swarm collaboration pattern, but not the negotiation and competition one. This is one of the reasons of the importance to study adaptive patterns for single components in connection with collaboration patterns, not only to suggest developers which kinds of patterns they can find, but also to guide designers that aim at proposing new adaptive collaboration patterns.

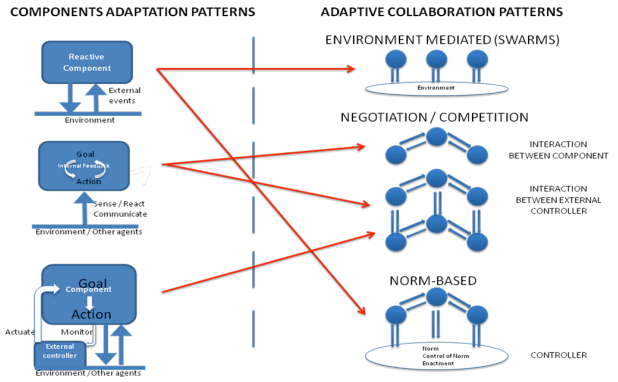

Figure 4. Patterns Summary

Considering together the features of adaptability and collaboration, our taxonomy can have the shape reported in Table 1. In the table, the *collaboration* classes of patterns are in the columns, while *adaptive* classes of patterns are in the rows.

3.4. Potential Advantages

Even if the work is at an early stage, we can identify two potential advantages in applaying the proposed taxonomy: *the reuse of existing experience* and *the support in choosing patterns*.

The fact that our classification is based on a settled set of agents' features can enable the reuse of the experience made so far. This means that the most studies performed on agents based on our three-feature definition can be applied to the classified patterns. In particular, the reuse concerns:

- *models*, which can applied to patterns with light adaptations;
- *methodologies*, or fragments of them, which guide developers during the development;

- *supporting tools*, which can be exploited during the de-velopment phases;
- *code libraries*, which are off-the-shelf implemented parts of software components that can be reused.

For instance, a well-known and exploited model is the BDI (Belief-Desire-Intention) one [33]. This model is based on information an agent has (beliefs), its goal (desire) and the actions to be performed to achieve the goal (intentions). Since this model is mainly related to the proactivity feature of agents, it can be useful in dealing with patterns classified as proactive, as well as in defining new proactive patterns. Another field that has been widely explored is the one related to agent auctions [36], a negotiation mechanism that well suits collaboration and competition among agents and that could be adopted in sets of general autonomous entities.

The second advantage is that our classification can help developers in choosing the most appropriate pattern on the base of scenario requirements. In fact, the class of a pattern potentially strongly characterizes its application to service components. A *reactive* pattern can be more suitable for components that provide functionality that is weakly dependent on the environment, and do not need to change their behavior frequently. In most cases, this simple kind of pattern is likely to be enough. A *proactive* pattern can instead be intended for situations where components are interested in adapting their behavior as much as possible, and they make everything they can to achieve this goal. Finally, a *social* pattern better can fit a set of coupled components that must collaborate.

4. RELATED WORK

Individual adaptation of components has been a very important thread of research since the early years of intelligent agents [20] and of reflective computing [40], and several architectures and mechanisms to enable *self*-adaptation has been proposed so far, typically based on goal-oriented agents and internal feedback loops (whether implicitly or explicitly architected within the component). For instance, the ACE component model defined in the context of the CASCADAS project[2] defines a very clean architecture enabling service components to dynamically adapt their behavior based on their context and on internally engineered feedback loops [5, 25]. The already mentioned autonomic computing approach of the BM has inspired several works related to enforcing adaptability via external feedback loops [22]. This approach is to most extent also shared by many works in the area autonomic communications [14, 32]. As another example, in the SELFMAN project [34], the idea is to enforce adaptation in components via an explicit phase of engineering external (possibly multi-fold) feedback loops. Complementary to individual adaptation researches is research related to adaptation of ensembles of software components, in order to meet new or unexpected requirements or contingencies. Such research thread, even if not new [1], still presents a relevant number of open issues [10], beside the ones that we have analyzed in this paper (and accounted for in our taxonomy). For instance, a key under-investigated issue relates to determining how it it possible to enable dynamic structural adaptations, i.e., dynamically changing the collaboration pattern in an ensemble of components, in order to preserving the required behaviors despite dramatic changes that could not be sustained by the originally chosen collaboration pattern. The possibility of dynamically self-selecting the most suitable adaptation pattern by an ensemble of component, which we called "dynamic self-expression", is one of the key methodological and technological innovations that will be pursed in the context of the ASCENS project. With this regard, an important lesson could come again from the area of multi-agent systems, and specifically of multi-agent systems, where the proper adaptation of a complex system made up of autonomous components may require the existence of "norms" that collectively regulate the activities of the components [12, 18, 38], possibly enacted via specific components of the set [28, 44]. In the area of biologically-inspired distributed computing, a large number of proposal for algorithms that can achieve adaptation in a distributed systems via decentralized self-organization have been proposed in the recent. These include a variety of nature-inspired algorithms and mechanisms [3, 8, 30] (see [24] for an in-depth survey), which, however, apply to specific problem instances and can hardly be translated to general-purpose schemes. What is still missing in general, despite some recent efforts in this direction [4], is a systematic classification of adaptive collaboration patterns – both at the level of individuals and ensembles, accompanied by methodological guidelines to choose among them.

5. CONCLUSIONS

In this paper, we have argued that agent-based computing can play a key role in supporting the design and development of future autonomic service systems, and have proposed a preliminary taxonomy of adaptive collaboration patterns, properly accounting for both the architecture of individual components and for the interaction patterns among ensembles of components. The benefit that our work would like to give, as preliminary as it can be, is twofold. Firstly, we think that a wide range of experiences in the area of agents and multi-agent systems can be reused to support adaptive collaboration patterns for service ensembles in complex applications scenarios, and our effort can represent a first useful step in this direction. Secondly, the taxonomy in itself can be useful

[2] http://acetoolkit.sourceforge.net/cascadas/

for the development of autonomic service components and ensembles, since it can suggest criteria according to which adaptive ensembles of components should be built.

Our future work will aim at studying and framing within the taxonomy as many patterns as possible, so to propose an almost complete classification of existing patterns. In particular, we will explore the existence of social patterns, or, better, to which extent existing negotiation and interaction protocols can be considered as social collaboration patterns. Of course, this will most likely require refining and extending the preliminary taxonomy here presented. Along with that, and possibly more interesting, we will aim at defining some more accurate guidelines for system developers, in order to help them in their work, and at promoting dynamic self-expression of collaboration patterns to enable service ensembles to dynamically chose the most suitable collaboration pattern.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] J. Andersson, R. de Lemos, S. Malek, and D. Weyns, "Reflecting on self-adaptive software systems", *Software Engineering for Adaptive and Self-Managing Systems, International Workshop on*, Vol. 0, pages 38–47, 2009.

[2] P.Anthony, W.Hall, V.Dang, and N.Jennings, "Autonomous agents for participating in multiple online auctions", IJCAI Workshop on EBusiness and the Intelligent Web, Seattle WA – USA, pages 54-64, 2001.

[3] Ö. Babaoglu, T. Binci, M. Jelasity, and A. Montresor, "Firefly-inspired heartbeat synchronization in overlay networks", 1st International Conference on Self-adaptive and Self-organizing Systems, Cambridge (MA) – USA, pages 77–86, 2007.

[4] Ö. Babaoglu, G. Canright, A. Deutsch, G. D. Caro, F. Ducatelle, L. M. Gambardella, N. Ganguly, M. Jelasity, R. Montemanni, A. Montresor, and T. Urnes, "Design patterns from biology for distributed computing", *TAAS*, Vol.1, No. 1, pages. 26–66, 2006.

[5] L. Baresi, A. Di Ferdinando, A. Manzalini, and F. Zambonelli, "The cascadas framework for autonomic communications", *Autonomic Communications*, Vol 2, pages 147-168, 2009.

[6] E. Bonabeau, M. Dorigo, and G. Theraulaz, SWARM INTELLIGENCE: FROM NTURAL TO ARTIFICIAL SYSTEMS, Oxford University Press, USA, 1999.

[7] L.Braubach, A.Pokahr, and W.Lamersdorf, "Jadex: a BDI-agent system combining middleware and reasoning", *Software Agent-based Applications, Platforms and Development kits*, pages 143–168, 2005.

[8] Y. Brun and D. Reishus, "Path finding in the tile assembly model", *Theor. Comput. Sci*., Vol. 410, No. 15, pages 1461–1472, 2009.

[9] G. Cabri, F. D. Mola, and R. Quitadamo, "Supporting a territorial emergency scenario with Services and Agents: a case study comparison", The IEEE 15th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Manchester - UK, pages 35-40, June 2006.

[10] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. D. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software engineering for self-adaptive systems: A research roadmap", *Software engineering for Self-Adaptive Systems*, Vol. 5525, pages 1-26, 2009.

[11] S. Cheng, V. Poladian, D. Garlan, and B. Schmerl, "Improving architecture-based self-adaptation through resource prediction", *Software engineering for Self-Adaptive Systems*, Vol. 5525, pages 71-88, 2009.

[12] Y. Demazeau, "From interactions to collective behavior in agent-based systems", 1st European Conference on Cognitive Science, Saint-Malo, pages 117-132, 1995.

[13] V. Dignum and F. Dignum, "Modelling agent societies: coordination frameworks and institutions" *Progress in Artificial Intelligence*, Vol. 2258, pages 7 -21, 2001.

[14] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications", *ACM Trans. Auton. Adapt. Syst*., Vol. 1, No. 2, pages 223–259, 2006.

[15] M. Esteva, J. Rodriguez-Aguilar, C. Sierra, P. Garcia, and J. Arcos, "On the formal specification of electronic institutions" *Agent mediated electronic commerce*, Vol. 1991, pages 126–147, 2001.

[16] P. Faratin, C. Sierra, and N. Jennings, "Negotiation decision functions for autonomous agents", *Robotics and Autonomous Systems*, Vol . 24, No. 3-4, pages 159–182, 1998.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, DESIGN PATTERNS, Addison Wesley, Reading (MA), 1995.

[18] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. W. Vasconcelos, "Constraint rule-based programming of norms for electronic institutions", *Autonomous Agents and Multi-Agent Systems*, Vol. 18, No.1, pages 186–217, 2009.

[19] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles", *IEEE Internet Computing*, Vol. 9, No. 1, pages 75–81, 2005.

[20] N. R. Jennings, “An agent-based approach for building complex software systems” *Communications of the ACM*, Vol. 44, No. 4, pages 35–41, 2001.

[21] N. Karacapilidis and D. Papadias, “Hermes: supporting argumentative discourse in multi-agent decision making”, Fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence American Association for Artificial Intelligence, Menlo Park (CA) - USA, pages 827-832, 1998.

[22] J. Kephart and D. Chess, “The vision of autonomic computing”, *IEEE Computer*, Vol. 36, No. 1, pages 41–50, 2003.

[23] D. Kortenkamp, R. Bonasso, and R. Murphy, ARTIFICIAL INTELLIGENCE AND MOBILE ROBOTS: CASE STUDIES OF SUCCESFUL ROBOT SYSTEMS, MIT Press Cambridge, MA - USA, 1998.

[24] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli, “Case studies for self-organization in computer science”, *Journal of Systems Architecture*, Vol. 52, No. 8-9, pages 443–460, 2006.

[25] A. Manzalini and F. Zambonelli, “Towards autonomic and situation-aware communication services: the cascadas vision”, DIS ’06: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications, Washington DC – USA, pages 383–388, 2006.

[26] R. Martin, “Design principles and design patterns”, *Object Mentor*, 2000.

[27] H. Müller, M. Pezzè, and M. Shaw, “Visibility of control inadaptive systems”, 2nd international workshop on Ultra-large-scale software-intensive systems, Leipzig - Germany, pages 23–26, 2008.

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

[29] E. Ogston and S. Vassiliadis, “A peer-to-peer agent auction”, First international joint conference on Autonomous agents and multiagent systems, Bologna - Italy, pages 151-159, 2002.

[30] P. Snyder, R. Greenstadt, and G. Valetto, “Myconet: a fungi-inspired model for superpeer-based overlay topologies”, 3rd International Conference on Self-adaptive and Self-organizing Systems, San Francisco – California – USA, pages 40-50, 2009.

[31] S. Parsons and N. Jennings, “Negotiation through argumentation a preliminary report”, 2nd International Conference on Multi Agent Systems, Kyoto – Japan, pages 267–274, 1996.

[32] R. Quitadamo and F. Zambonelli, “Autonomic communication services: a new challenge for software agents”, *Autonomous Agents and Multi-Agent Systems*, Vol. 17, No. 3, pages 457–475, 2008.

[33] A. Rao and M. Georgeff, “BDI agents: From theory to practice”, First international conference on multi-agent systems (ICMAS-95), San Francisco – USA, pages 312–319, 1995.

[34] P. V. Roy, “Overcoming software fragility with interacting feedback loops”, BCS International Academic Conference, London - UK, 2008.

[35] E. Sahin, T. Labella, V. Trianni, J. Deneubourg, P. Rasse, D. Floreano, L. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo, “SWARM-BOT: Pattern formation in a swarm of self-assembling mobile robots”, *IEEE International Conference on Systems, Man and Cybernetics*, Vol.4, 2003.

[36] T. Sandholm, “Algorithm for optimal winner determination in combinatorial auctions”, *Artificial Intelligence*, Vol. 135, No. 1-2, pages 1–54, 2002.

[37] R. Smith, “The contract net protocol: High-level communication and control in a distributed problem solver”, *Computers, IEEE Transactions on*, Vol. 100, No. 12, pages 1104–1113, 2006.

[38] N. A. M. Tinnemeier, M. Dastani, and J.-J. C. Meyer, “Roles and norms for programming agent organizations”, 8th International Conference on Autonomous Agents and Multiagent Systems, Budapest – Hungary, pages 121–128, 2009.

[39] J. Vázquez-Salceda. The role of Norms and Electronic Institutions in Multi-Agent Systems applied to complex domains. The HARMONIA framework. *AI Communications*, 16(3):209–212, 2003.

[40] T. Watanabe and A. Yonezawa, “Reflection in an object-oriented concurrent language”, ACM Conference on Object-Oriented Programming Systems, Languages, and Applications, San Diego - California - USA pages 306–315, 1988.

[41] D. Weyns, S. Malek, and J. Andersson, “FORMS: a formal reference model for self-adaptation”, The 7th international conference on Autonomic computing, Washington DC - USA pages 205–214, 2010.

[42] M. Wooldridge, AN INTRODUCTION TO MULTIAGENT SYSTEMS, Wiley, 2009.

[43] P. Wurman, M. Wellman, and W. Walsh, “The Michigan Internet AuctionBot: A configurable auction server for human and software agents”, The second international conference on Autonomous agents, Minneapolis – MN – USA, pages 301–308, 1998.

[44] F. Zambonelli, N. R. Jennings, and M. Wooldridge, “Developing multiagent systems: The Gaia methodology”, *ACM Transactions on Software Engineering and Methodology,* Vol. 12, No. 3, pages 417–470, 2003.