Agreement Technologies for Adaptive, Service-Oriented Multi-Agent Systems

J. Santiago Pérez[1], Carlos E. Cuesta[2], and Sascha Ossowski[1]

[1] Centre for Intelligent Information Technologies (CETINIA), and
[2] Kybele Research Group, Dept. Comp. Languages and Systems II
Rey Juan Carlos University,
28933 Móstoles (Madrid), Spain
{josesantiago.perez,carlos.cuesta,sascha.ossowski}@urjc.es

Abstract. Multi-Agent Systems (MAS) are increasingly popular in Artificial Intelligence (AI) to solve complex problems. They can be conceived flexible and able to adapt to different situations. However, these features are often compromised by the characteristics of the problem itself. On the other hand, MAS have not had a lot of success in the industry, probably due to a different development culture. To solve this, MAS techniques should be more accessible to the general public, and have a shorter learning curve. The proposed approach is to use service-oriented concepts, which are popular in industry, to simplify this step. Moreover, if this approach manifests also self-adaptive capabilities, it will fulfil the notion's original promise: to guarantee that the system is able to adapt to changing conditions of the problem. This work proposes a service-oriented framework, consisting on a supporting agent-oriented architecture, a development methodology for service-oriented MAS, and an infrastructure based on the concept of agreement, which makes it adaptive. The first section provides a brief introduction and summarizes the paper goals. This is followed by the description of the base architecture, designed to support the agreement structure. Next section discusses concepts about service layers and the role of organizations. After that, the service-oriented methodology as well as the agreement structure itself is presented. Finally, a real-world case study, in the domain of medical emergencies, is analyzed, some conclusions are drawn, and further lines of work are outlined.

Keywords: Multi-Agent Systems, Service-Oriented Architecture, agreement, coordination, adaptability.

1 Introduction

The concept of *agent* has evolved, and nowadays MAS are increasingly popular in AI as a generic approach to solve complex problems. Different development strategies have been proposed in order to make them flexible and able to adapt to different situations. However, these features are often compromised by the heterogeneity of components, the nature of problems themselves, or the dynamism in the environment. On the other hand, MAS have not had a lot of success in the industry [14][36], probably due to a different development culture. To solve this, MAS techniques should be more accessible to the general software community. The proposed approach is to use service-oriented concepts, which are popular in industry, to simplify this step. Moreover, if this approach demonstrates also self-adaptive capabilities, it will fulfil MAS original promise: to guarantee that the system is able to adapt to changing conditions in the problem to solve.

Before dealing with adaptability, it is perhaps better to consider coordination as a previous concept. A well-known definition of “coordination” within the MAS field is taken from Organizational Science: “the management of dependencies” between organizational activities [27]. From a “micro” point of view (agent-centred) [35], coordination is understood as an *adaptation to the environment*. On the other hand, from an MAS-centred point of view, the consequences of coordination can be understood as a global influence. This can be a “shared” plan [30] or the combination of individual plans (a “multi-plan”) [28]. In few words, when using MAS as a software solution, the problem of coordination is always present. In fact, when we have a self-organized agent structure, we can often consider this structure as *optimal*, because it would solve the coordination issues.

Some early steps in the direction of adaptability have been given by organization-oriented approaches. Obviously there are many other approaches, but this is one of the most interesting in our context: adaptive capabilities, using a MAS approach, seem to be most easily provided by organizations. These imply a number of additional questions: about the inner role of organizations in MAS and about the need to provide coordination for organizations to achieve adaptation. To answer to them, two additional concepts have to be defined; respectively, *services* of an organization and *agreements* between them. The former provides both a methodological basis for the approach, as well as a direct connection to SOA [26]. On the other side, the latter is a main topic of this paper, and it will be discussed in detail.

Globally, this paper pursuits three main goals, namely:

- To evolve the classic agent-oriented approach, from an originally closed MAS design into an *open* Service-Oriented ecosystem,
- To define the corresponding infrastructure and methodology to achieve this, using the notion of *organization* as the conceptual nexus, and
- To provide internal *coordination* by defining the *agreement*, conceived as an adaptive architecture-level construction, which would provide coordination as an emergent property, by containment.

This paper is organized as follows: second section describes the base architecture, designed to support the agreement structure. Next section discusses concepts about service layers and the role of organizations. Them, the service-oriented methodology as well as the agreement structure itself is presented. Finally, a real-world case study, in the domain of medical emergencies, is analyzed, some conclusions are drawn, and further lines of work are outlined.

2 A Base Architecture for Service-Oriented MAS

The architecture that gives support to the model has been defined both as an open MAS and also as a service-oriented, organization-centric, agent-based architecture. These two perspectives are not necessarily contradictory; they are not obviously compatible either.

For both descriptions to be true, the platform has to be capable of being observed at different levels and from different perspectives. This multi-level and multiple viewpoint nature must be specifically enabled by the technical architecture (see 2.2), as it must present several different notions as the key concept of the system. This requires an intertwining relationship which must be purposely provided by the infrastructure. As the platform is conceived as a distributed system, the middleware is the logical place to provide this support.

2.1 The Need for Organizations

As defined previously, the architecture that supports the model has been defined as open MAS, which is also service-oriented, organization-centric and, of course, agent-based. In this work, agents supporting services has been chosen as the solution alternative. First, agents are well-known computational entities in the academic environment, with an implied granularity, and need to comply with an existing standard [20]. On the other hand, although the services technology is established and has a number of standards [7][12][17][26], its methodology and influence on other paradigms are still under development. In order to allow the use of the rich semantic and technological capabilities of agents in a broader context, an upper layer of services can be added to provide, in particular, the *interoperability* feature. Therefore, it is easy to conceive a service as a way to present the operational capabilities of an agent or, even better, a collection of agents as an organization. One way to implement is to have the platform defined as a SOA, built on top of supporting MAS.

Implicit in the definition of MAS is the need to *register* agents in the system, to separate those ones who belong to the architecture from those who do not. The same approach will be used to identify services. To allow their external access, they will be explicitly registered and grouped as part of a service. This service could be later discovered by other entities within the distributed registry of the system.

Pure *agent-oriented* MAS methodologies (such as MAS-CommonKADS [24], Gaia [38], MaSE [37], Tropos [23] or Prometheus [29], among others) usually concentrate in the agent vision. It is assumed that the final behaviour of the system *emerges* from the interrelations between the designed agents. But the global behaviour is not analyzed in detail.

On the other hand, in *organization-oriented* MAS methodologies, the analysis is made from a global perspective (Agent-Group-Role [19], MESSAGE [9], ANEMONA [22], AML [11], OperA [15], Civil Agent Societies [13], MOISE [21], Electronic Institutions [18], HARMONIA [34], GORMAS [3], among others). The objectives describe the organizational purposes at a high level. This allows the determination of tasks, types of agents, resources assignation between members, etc. In this approach, norms are very important because they describe the desired behaviour of the members. These norms will derive in control, prohibitions, sanctions, etc. to achieve the expected global behaviour. Mechanisms to allow external agents to enter the organization and control their behaviour are particularly useful to design *open* MAS.

2.2 The Agreement Technologies Base Architecture

The set of technologies and approaches used in this work is globally named as “Agreement Technologies” [1]. This section presents the base architecture for these technologies, and, as it was noted in the previous section, it was conceived to be based in an open MAS.

One goal of the proposed approach is to take advantage of MAS features, so the research is oriented to achieve a greater capacity and functionality, with a lesser emphasis on efficiency or scalability. Moreover, and from this point of view, services are used to achieve interoperability, as mentioned earlier. The main idea is to export the agent system as a system of services, which will be supported, not only technologically, but also methodologically.

These concepts are intended to be built on top of existing and concurrent work. It is not the purpose of the article to give a complete description of the THOMAS architecture, which can be found in [4]. But briefly, its design can be summarized as described in the following.

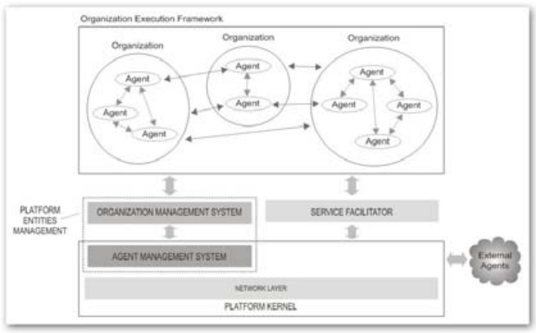

Figure 1: THOMAS Technical Architecture (inspired on [4])

The platform, including its middleware, (Figure 1) is structured in three levels but they are not strictly layers. They are orthogonally supported by four specific components, which are included as part of three different subsystems. The *Platform Entities Management* subsystem is actually layered in turn. The different layers of this subsystem are used to provide capabilities for different levels in the platform. The three levels are:

- *Platform Kernel (PK).* It is the actual kernel of the middleware; includes both the Network Layer and the Agent Management System (AMS) component. It provides all the capabilities of FIPA-compliant architecture [20]. Therefore, at this layer the platform is already an (open) Multi-Agent System.
- *Service & Organization Management*. This is the conceptual level composed of the Organization Management System (OMS) and the Service Facilitator (SF) components. Both components provide all the relevant features and abstractions for the Execution Framework.
- *Organization Execution Framework.* It is the "space" where all the computational entities "live" and perform their functions. Agents and their organizations, and the services they offer, are conceptually located in it. Every specific application would be conceived, designed and executed at this abstraction level.

The aforementioned three main components of the platform are: *AMS,* which provides all the required capabilities and functions for managing an agent; *OMS*, which provides all the required capabilities and functions for managing an organization, and maintains together the system as a whole; and *SF*, which provides the required capabilities and functions to allow that a certain selection of the operations in an organization behave as a unified service.

3 The Service-Oriented Layer

As already noted, the base architecture will be primarily conceived as a service-oriented. Hence, an important concept is that of *service*.

According to their provider, there are basically *base services* (user-level services, and they are defined for every concrete application); and s*ystem services* (not strictly "services" as they are not offered by a concrete user-level provider, they are provided by the system itself, i.e. they are the support services of the platform).

Taken into account their function and the extent of their capabilities three separate sets of services can be identified in the architecture:

- *Structural Services*. They allow defining a certain organizational/architectural structure, by creating and registering organizations, their roles and norms, and their relationships. They make possible to establish and modify both structural and normative specifications of the system and they are provided by the OMS.
- *Information Services*. They provide specific information about components in an organization. Also, some of them are published as registered services, while some others are just conceived for the use of the OMS and stay invisible.
- *Dynamic Services*. They allow entities to dynamically enter or abandon an organization, as well as to adopt existing roles. Units and roles have been previously defined and registered by using structural services. Dynamic services are just able to modify services, units and roles. These services provide dynamic reconfiguration.

3.1 The Role of Organizations

The *organization* is the most important active element and the unifying notion of the architecture itself. The recursive hierarchy of organizations is what would make possible to simultaneously define the architecture as service-oriented and as agent-based. The concept of organization is the nexus between both perspectives.

An organization can be seen from two points of view: externally, it can be considered as a context, a domain of influence, the scope of a set of norms and rules; and internally, it can be considered as a collection, the gathering of the set of individuals which would comply with the stated norms and fill the defined roles. An organization is also composed of units (or organizational units). A *unit* is an active entity with a definite, externally observable behaviour, and it can have either a collective nature (where the unit is itself an organization) or an autonomous nature (when the unit is just a single agent). The *unit* is therefore the substrate which supports both the gathering of agents and the definition of services.

The concept of organization is also used to solve the scaling problem of architecture, in the context of services. Since they generally are intended to be used in-the-large, it is necessary to use a compositional structure: the organization itself. In this vision, low-level services are essentially provided by individual agents, while system-level services are provided by roles in a complex organization. Intermediate levels can also provide their services, so the recursive organizational hierarchy defines the compositional “spine” for the system.

As implied before, from this point of view everything is a *unit*. The system itself must be conceived from within as a unit, and therefore, it is an organization too. As such, it gathers the contributions of both individual agents defining the small-scale MAS, as well as those from the middleware itself, which supports the technical architecture, as described in section 2.

4 A Service-Oriented Methodology

As already said, the proposed approach is to group agents into organizations, but this is not a simple task. Some questions arise, such as: Which agents belong to an organization? What criteria will be used to group them? Moreover, the process of exporting the capabilities of agents as services leads to another question: What services should be exported?

A methodology is proposed in an attempt to answer all these questions. A first step involves the functional decomposition of services, and this leads to define organizations. Then, as a second step, the composition of services is guided by the organizations and their structure.

The system is conceived as service-oriented, so, high-level services are proposed as the starting point. Their functional decomposition (or a hierarchical decomposition, from another point of view) will be also used to design the hierarchical structure of organizations.

A service is defined as a computational entity which gathers a set of operations, described in its standard interface, and comprised a semi-ordered sequence of activities, semantically described by an intentional profile and an explicit process model, which can in turn be split in several smaller processes. There may be several implementations for the same service and an identical profile, which are offered by different (possibly many) service providers.

The concept of *service process*, in this context, intends to provide a clear semantic perspective of a service's functionality, by describing it as a workflow.

The service process model identifies three kinds of processes in the structural description. This classification, designed from a semantic perspective [2], will be used to support the methodology, and assist in the design of the structure of organizations. These types of processes are:

- *Atomic processes* can be directly invoked, execute in a single step, and cannot be decomposed.
- *Simple processes* are also perceived to be executed in a single step, but cannot be directly invoked. They are abstract processes (placeholders) and can be filled either by an atomic process; or (acting as a simplified representation) by a composite process.
- *Composite processes* are decomposed in sub-processes, which can be defined in turn as atomic, simple or composite ones. This way, the service's functionality unfolds recursively as a hierarchic composite structure.

Simple processes (which are also services) allow a next level of decomposition. High-level services can be described as a set of simple processes. Those actually simple are described as *atomic* services (i.e. agent operations); and those that are more complex are considered as *composite* processes, which will be further decomposed. Organizations can be now identified by relating each service with its provider, unfolding their hierarchical structure.

From this point of view, the composition of services is given by the organizational structure itself. Though the approach here has a semantic nature, this is essentially the same approach which is also used for this purpose, from a behavioural perspective, in the context of service composition, based on orchestration [25].

In particular, both approaches use the process abstraction as the way to describe the behaviour of a service, and specifically the composition of (smaller-scale) services. Also, provide a number of control structures, which define a principled way to combine sub-processes into larger processes, providing compositionality and recursive structures.

There is an implicit relationship between these recursive structures: (composite) processes can be provided as services by (composite) organizational units; when these processes are decomposed, the resulting sub-processes can be provided in turn by other units. That is, sub-processes of a composite process would be provided by the members (units) of the composite organization which provided the upper level. When this happens, the recursive structure of processes mimics the recursive structure of organizations. The converse is also true: starting from simple tasks, a vertical composition method could help in the definition of the organizational hierarchy, defining at the same time the resulting complex (composite) processes. Like in the case of organizations, the recursion ends at the agent level.

Therefore, our approach provides the structure for the vertical composition of services. This way, a task that is often considered difficult –to design the service composition– is methodologically tackled, allowing at the same time to fully exploit the organizational structure of the agents. Then, there is a mutual support between these two concepts.

5 The Agreement Structure

Agents were originally conceived as single actors, but within the MAS approach, a different method has become possible. The need for a trade-off continues, but has it transformed into a *coordination* problem. As already said, the service can be conceived as a way to present the operational capabilities of an agent (or a collection of them) inside an organization.

The proposed methodology allows tackling the decomposition of services, but adaptability in the system is provided by the architecture. First, there is a decomposition of services to provide the required features; but after that it is necessary to address the structure of agreements which supports this decomposition, in order to make it adaptive.

So, an important notion is the *agreement* between computational entities (organizations, at the top levels; but also agents, at the lower ones) conceived as an *architectural construct*. The following subsections discuss the need for an adaptive structure, and the agreement model.

5.1 The Need for an Adaptive Structure

When using MAS as a software solution, as already noted, the problem of coordination is always present. When they define a self-organized structure, it sometimes implicitly solves the coordination issues; this approach could be considered as optimal.

When a complex problem is tackled in an *ecosystem* (or a system of systems), the solution requires certain adaptability. At the same time, this structure needs to be flexible to achieve coordination inside the ecosystem, and also this behaviour could be emergent.

Pioneer works related to cooperation define adaptiveness as a required notion for intelligent solution of complex problems [2]. Two approaches can be considered: *from the collaborative entity point of view* (cooperation is introduced as an additional mechanism to increase the effectiveness in solving problems); and *from the problem to solve point of view* (this intends to find the best way to structure and decompose a complex problem to solve it effectively). Taking into account these approaches, several solutions to the cooperation problem were developed. The *blackboard* architecture [16] provides cooperation between knowledge sources using a simple communication mechanism. The *contract net* [32] proposes *negotiation* as a mechanism to coordinate and to assign tasks to different entities participating in problem solving. The *reactive architecture* [8] tries to obtain an intelligent behaviour from simple models, without knowledge representation, reasoning or learning mechanisms. Finally, agent architectures with *organizational* capacity appeared: agents need to know about their own capabilities and *social* features.

Generically, entities are organized into a structure by using *controls*, which either *enforce* or *forbid* specific interactions –or connections–; and *protocols,* which either *enable* or *channel* them. Therefore, where the former are based on force or *imposition*, the latter are based on consensus and *agreement.*

The concept of agreement among computational entities seems to be a right approach to tackle the need for an adaptive structure. The objective is to "discover" a suitable structure of controls and protocols so that it *emerges* as a global structure, *the agreement*. This will make possible to define the main inner structures in order to obtain agreement-based organizations.

As the structures of agents are become more and more complex, it is clear that for some kind of problems we need not a superstructure, like the blackboard. Agents that *organize themselves* in organizations (and after that in agreement-based organizations) are needed. The main objective is to evolve from that emergent coordination to an *emergent agreement* between entities.

5.2 The Agreement Model

As already noted in previous sections, a central notion in this approach is the *agreement* between computational entities. Continuing with research efforts in the field of "Agreement Technologies" [1], the process of *agreement-based coordination* can be conceived as consistent with the normative context where agents are established and allow them, once accepted, to call for mutual services, and to be called by others.

Several key research topics must be considered and they can be seen in a "tower" structure [1] where each level provides functionality and inputs to the one above. Therefore, the agreement must be seen as a *layered* structure, by definition: when an agreement is reached at a certain level, elements located at lower levels must respect it at their own level. These "tower" levels, from bottom up, are:

- *Semantics*: the bottom one, as semantic issues influence all others. The semantic alignment of ontologies [6] is necessary to avoid mismatches and is needed to have a common understanding.
- *Norms*: is concerned with the definition of rules determining constraints that the agreements, and the process to reach them, have to satisfy.
- *Organizations*: implies a super-structure that restricts the way agreements are reached by fixing the social structure of the agents, the capabilities of their roles and the relationship among them [5].
- *Argumentation and Negotiation*: can be seen as protocols that define the structure of an agreement.
- *Trust*: the top level in the tower. Agents need to use trust mechanisms that summarize the history of agreements and subsequent agreements executions in order to build long-term relationships between them [31].

These five layers, of course, are not seen as isolated because they may well benefit from each other. For example, if changes in some norms or to take advantage of negotiation methods, the organizational model has to be modified. A switch from the described "tower" into a multi faceted ("pentagon") figure can be conceived because the agreement pervades (and is influenced by) all the facets/levels (Figure 2). In this sense, the facets are intertwined, but *agreement* is still a layered structure – and layers bind both ways.

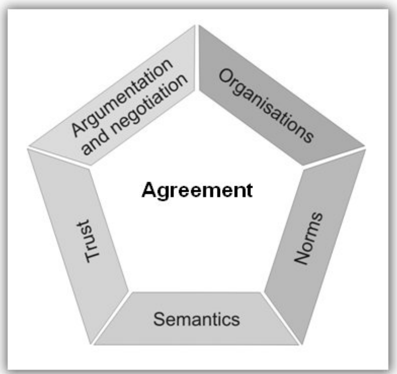

Figure 2: Multi-faceted perspective on the structure of an Agreement

In fact, the agreement is a crosscutting structure, which maintains a bidirectional relationship to every element it contains. The agreement defines the architecture but at the same time, the architecture defines the agreement. The agreement is *shaped* by those forces, but its existence also *shapes* the reaction to them, and models the future evolution of the system. It is important to note that the multi-faceted perspective is not intended to replace the "tower" structure, as the architecture described in previous subsection is still hierarchical in many senses, but the *agreement* itself can be considered not only as layered, but also as multi-faceted. Layers are just conceived to provide logical separation of concerns, and they are not always physical (contained) tiers. On the other side, in an MAS setting, a reconfiguration can also be triggered bottom-up; a single agent can react to a change in their surroundings by asking for some kind of change, such as a *move* to some other organization. Of course this change can cause some others in turn, and the effect would spread accordingly, causing even a global reorganization.

In summary, the system already provides the required elements to build an adaptive architecture; to actually define an *emergent agreement* would just require identifying the structural patterns, and the set of inter-level protocols. Some refinements can be made further, though the need for meta-elements has still to be considered, nothing excludes the definition of specific agents to carry out support tasks for the agreement itself (such as sensors, observers or even planners).

6 Case Study: *mHealth*

This section presents a case study in order to illustrate the proposed approach. Our purpose is to show the reason why an *agreement* between entities is not only necessary, but it can also be a natural solution to complex problems. As already said in section 3.1, the structure in organizations can be seen as a logical strategy to tackle complex situations, and has also several advantages. The need for a flexible and adaptive agreement construct can also be seen as the basis to create and evolve these organizations. Section 5 has described the structure of the *agreement* structure which could address that need.

The example is related to the *mHealth* (mobile-Health) demonstrator, which is an evolutionary prototype currently under development within the Agreement Technologies project [1]. It is inspired by work with SUMMA112 [33], the centre that manage medical emergencies in the Autonomous Region of Madrid, which is also involved in the project.

In the following, an initial emergency (E1) is described. The system has to evolve to simultaneously react to a second one (E2).

E1. There is a fire in Casa de Campo (a large urban park). There are about 500 people at that moment and about 65 of them present symptoms of asphyxia. *SUMMA112* receives information related to E1 and decides that 5 ambulances and one helicopter are needed. The coordination with hospitals near the area, Fire Department (*FD*) and Police (*P*) is also urgent. *FD* and *P* will send 3 fire trucks and 5 police cars. From an organizational approach, all these elements form an organization, **O1**. Each actor maps onto an agent considering this scenario as MAS. Then, there are 14 agents are interacting in the organization O1. Each agent has its role, goals and plans inside the organization, which in turn has its own norms and protocols.

E2. One hour after E1, there is a chain car crash (E2) in the tunnel of Paseo de Extremadura, a road near to E1 location. Several cars have crashed and 2 of them are on fire. *SUMMA112* decides that this emergency requires 3 ambulances. In this case, *FD* and *P* decide to send one fire truck and 3 police cars. Again, all these 7 elements form a second organization, **O2**.

Basically, this scenario can be solved using two alternative solutions: deal with O1 and O2 as separate elements, with no relation between them; or, deal with O1 and O2 as *units with some degree of relationship*.

The second is the most efficient and sensible approach, as it must have into account potential interactions between both emergencies. So, let's consider first O1, whose elements reach an *agreement* to tackle E1. At this point, the agreement construct can be seen as "the set of elements interacting in a coordinated way to solve a problem". But at the time to assign resources to E2, O2 is not considered in isolation from O1. Some resources that previously were mapped onto O1 now can be mapped on O2 because the conditions in emergency E1 may have changed during the last hour. This process of re-mapping implies a *reconfiguration* of unit O1, i.e. an agent's *reorganization* within the *O1O2 composite*.

Some services which were provided by unit O1 are no longer required in E1 and now can be re-mapped onto O2. This can be done at different levels (for instance, registering services at the unit level, with no structural changes); but the simplest and most efficient solution implies not only re-assigning *services*, but also the *agents* which provide them, i.e. doing a *reorganization*. For example, according to the observed results in O1 some services can be assigned to E2. Additional elements are also assigned to E2 to fulfil O2 necessities. O1, a smaller unit now, continues working in E1; and a new agreement is created around E2, defining the O2 organization. At the same time, a larger *agreement* is created encompassing both units (and therefore, defining another one). This agreement would continue adapting to changes in both emergencies as system evolves.

Elements participating in an *agreement* (*O1+O2*) must be capable to adjust themselves to environmental changes, to accomplish the goals in the agreement. This will often lead to changes, not in the elements themselves, but on their configuration. In fact, even the criteria used to decide if an agent belongs in an agreement should be managed the same way: this defines an *emergent agreement*, where not only part of the behaviour, but the structure itself emerges from the situation.

The base architecture described in Section 2 already includes all the services and facilities necessary to carry out any reconfiguration [4]. However, this is not enough to define a self-adaptive structure – the triggering of those services is essential. Of course *norms* (to define constraints) and *organizations* (to define their scope) can assist in the establishment of such a structure; and even the *negotiation* layer can be used to trigger the creation of the agreement itself.

7 Conclusion

It has been argued that MAS techniques should be more accessible to software community in general in this paper. As services are concepts very popular in industry and can simplify the transition, this work has proposed a service-oriented framework, consisting on a supporting agent-oriented architecture; a development methodology for service-oriented MAS; and an infrastructure based on the concept of agreement, which makes it adaptive.

The example shows why it is needed to consider a general *ecosystem*, instead a "classic" *closed* system or a single-design *open* system. To actually provide the required response in an emergency, SUMMA112 has to coordinate with the information systems from the Fire Department, the Police, and every hospital in the area. This implies that it is not possible to have a unified pre-*programmed* strategy to manage emergencies, as it should be embedded in several independent systems which only sometimes gather to act together.

The key idea in the Agreement Model is that it creates an architectural *context*, in which agents (organizations, services) are coordinated and reorganized by inclusion in a structure. In particular, there is *not* an architectural element in charge of reconfiguration, i.e. there is not a self-*supervisor*. Instead of that, every self-property in the system is conceived as *emergent*, and they will be “indirectly” provided by structural features of the agreement. The elements do just what they must to comply with the requirements of the location they occupy within the architecture; the relationships between the *agreement facets* will do the rest. Again, the case study discussed previously describes a simulated coordination effort in the current SUMMA112 system. In [10], MAS structured in organizations, and implemented in THOMAS architecture has been used to model systems and simulate several situations.

The reconfiguration process has also been modelled and tested using several different approaches; but this manual process is only the first stage of research. The next step is to develop a *model-driven* approach to guide the reconfiguration, and will be followed by a well-defined self-adaptive, emergent approach, which is the ultimate goal.

Acknowledgment: This work has been partially funded by Project AT (CONSOLIDER CSD2007-0022, INGENIO 2010) of the Spanish Ministry of Science and Innovation, and from COST Action AT (COST IC0801) from the EU RTD Framework Programme.

References

[1] Agreement Technologies (AT) Project: http://www.agreement-technologies.org/ (2009)

[2] Ana Mas: Agentes Software and Sistemas Multi-Agente: Conceptos, Arquitecturas y Aplicaciones. Prentice-Hall (2005).

[3] Argente, E.: GORMAS: Guidelines for ORganization-based Multiagent Systems. PhD thesis, Universidad Politécnica de Valencia (2008).

[4] Argente, E., Botti, V., Carrascosa, C., Giret, A., Julian, V., and Rebollo, M.: An Abstract Architecture for Virtual Organizations: The THOMAS Project. Technical report, DSIC, Universidad Politécnica de Valencia (2008).

[5] Argente, E., Julian, V., and Botti, V.: Multi-Agent System Development based on Organizations. Electronic Notes in Theoretical Computer Science 150(3):55-71 (2006).

[6] Atienza, M., Schorlemmer, M.: I-SSA - Interaction-situated Semantic Alignment. Proc Int. Conf. on Cooperative Information Systems (CoopIS 2008) (2008).

[7] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D.: Web Services Architecture. W3C WSA Working Group, W3 Consortium (2004)

[8] Brooks, R.: Intelligence without Representation. Art. Intelligence, 47:139-159 (1991).

[9] Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., and Massonet, P.: Agent-oriented analysis using MESSAGE /UML. LNCS vol. 2222:119–125 (2002).

[10] Centeno, R., Fagundes, M., Billhardt, H., and Ossowski, S.: Supporting Medical Emergencies by MAS. In “Agent and Multi-Agent Systems: Technologies and Applications”. LNCS, vol. 5559:823-833. Springer (2009).

[11] Cervenka, R., and Trencansky, I.: AML. The Agent Modeling Language. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkauser (2007).

[12] Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C Consortium. W3C Note (2001)

[13] Dellarocas, C., and Klein, M.: Civil agent societies: Tools for inventing open agent-mediated electronic marketplaces. In ACM Conf. Electronic Commerce (EC-99) (1999).

[14] DeLoach, S.: Moving multi-agent systems from research to practice. International Journal of Agent-Oriented Software Engineering - Vol. 3, No.4 pages 378 – 382 (2009)

[15] Dignum, V.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD thesis, Utrecht University.

[16] Erman, L., Hayes-Roth, F., Lesser, V., Reddy, R.: The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. ACM Computing Surveys 12(2), pages 213-253 (1980)

[17] Esteban, J., Laskey, K., McCabe, F., and Thornton, D.: Reference Architecture for Service Oriented Architecture 1.0. Organization for the Advancement of Structured Information Standards (OASIS) (2008).
[18] Esteva, M., Rodriguez, J., Sierra, C., Garcia, P., and Arcos, J.: On the Formal Specification of Electronic Institutions. Agent Mediated Electronic Commerce 1991, pages 126–147 (2001)
[19] Ferber, J., Gutkenecht, O., and Michel, F.: From Agents to Organizations: an Organizational View of Multi-Agent Systems. In Proc. AAMAS03 - Agent-Oriented Software Engineering Workshop (AOSE) (2003).
[20] FIPA. FIPA Abstract Architecture Specification. Technical Report SC00001L, Foundation for Intelligent Physical Agents. FIPA TC Architecture (2002).
[21] Gateau, B., Boissier, O., Khadraoui, D., and Dubois, E.: MOISE-Inst: An Organizational model for specifying rights and duties of autonomous agents. In der Torre, L. V., and Boella, G., eds., First Intl. Workshop on Coordination and Organisation (2005).
[22] Giret, A.: ANEMONA: Una metodología multi-agente para sistemas holónicos de fabricación. PhD thesis, Universidad Politécnica de Valencia (2005).
[23] Giunchiglia, F., Mylopoulos, J., and Perini, A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. In Proc. Workshop on Agent Oriented Software Engineering (AOSE), 63–74 (2002).
[24] Iglesias, A., Garijo, M., Gonzalez, J., and Velasco, J.: A methodological proposal for multiagent systems development extending CommonKADS. In Proc. 10th Banff Workshop Knowledge Acquisition for Knowledge-Based Systems (1996).
[25] Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guizar, A., Kartha, N., Kevin Liu, C., Khalaf, R., Koening, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluiri, P., and Yiu, A.: Web Services Business Process Execution Language (WSBPEL) 2.0. Organization for the Advancement of Structured Information Standards (OASIS) (2007).
[26] MacKenzie, C., Laskey, K., McCabe, F., Brown, P., and Metz, R.: Reference Model for Service Oriented Architecture 1.0. Organization for the Advancement of Structured Information Standards (OASIS) (2006).
[27] Malone, T., Crowston, K.: The Interdisciplinary Study of Co-ordination. Computing Surveys 26 (1). ACM Press, pages 87–119 (1994).
[28] Ossowski, S.: Co-ordination in Artificial Agent Societies, LNAI 1535. Springer (1999).
[29] Padgham, L., and Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In Proc. Agent Oriented Software Engineering (AOSE), 135–145 (2002).
[30] Rosenschein, J., and Zlotkin, G.: Rules of Encounter – Designing Conventions for Automated Negotiation among Computers. MIT Press (1994).
[31] Sierra, C., Debenham, J.: Information-Based Agency. Proc Intl. Joint Conference on AI (IJCAI-2007). AAAI Press, pages 1513-1518 (2007).
[32] Smith, R.: A Framework for Problem Solving in a Distributed Processing Environment. PhD thesis, Stanford University (1978).
[33] SUMMA112: http://www.madrid.org/cs/Satellite?language=es&pagename=SUMMA112%2FPage%2FS112_home (2009).
[34] Vazquez-Salceda, J., and Dignum, F.: Modelling Electronic Organizations. Lecture Notes in Artificial Intelligence 2691:584–593 (2003).
[35] Von Martial, F.: Co-ordinating Plans of Autonomous Agents. LNAI 610, Springer (1992)
[36] Weyns, D., Helleboogh, A., and Holvoet, T.: How to get multi-agent systems accepted in industry? International Journal of Agent-Oriented Software Engineering - Vol. 3, No.4 pages 383 – 390 (2009)
[37] Wood, M., DeLoach, S., and Sparkman, C.: Multiagent system engineering. Journal of Software Engineering and Knowledge Engineering 11:231–258 (2001).
[38] Wooldridge, M., Jennings, N., and Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. J. Autonomous Agent and Multi-Agent Systems 3:285–312 (2000).