SADIO Electronic Journal of Informatics and Operations Research

http://www.sadio.org.ar

vol. 10, no. 1, pp. 53-67 (2011)

The Role of Agreement Technologies in the Definition of Adaptive Software Architectures

J. Santiago Pérez-Sotelo[1] Carlos E. Cuesta[2] Sascha Ossowski[1]

[1] Centre for Intelligent Information Technologies (CETINIA)
Rey Juan Carlos University
28933 Móstoles (Madrid), Spain
e-mail: {josesantiago.perez, sascha.ossowski}@urjc.es

[2] Kybele Research Group
Dept. Computer Languages and Systems II
Rey Juan Carlos University
28933 Móstoles (Madrid), Spain
e-mail: carlos.cuesta@urjc.es

Abstract

The growing complexity of software systems is causing a re-conception of their development and maintenance strategies. Humans should be relieved from an important part of these tasks, which should be performed by systems themselves, leading to consider self-adaptation as a basic architectural concern. Simultaneously, Multi-Agent Systems (MAS) have been developed as a generic approach to solve complex problems. They describe self-aware structures, conceived to be flexible and to be able to adapt to different situations. Advances approaches use organizations to provide further structuring, taking the form of complex agent architectures. Among them, Agreement Technologies (AT) provides an explicit insight into those architectural abstractions. However, they still do not provide mechanisms to change their composition patterns and element types, which are necessary to achieve real self-adaptivity. In this article, we propose an architectural solution for this: the required dynamism will be supported by an emergent agreement - an evolving architectural structure, based on combining predefined controls and protocols. These are handled in the context of the service-oriented, agent-based and organization-centric framework defined in AT and provided by their implementation within the THOMAS platform. This work provides the first architectural abstractions to support this emergent structure. A real-world example showing the interest of this approach is also provided, and some conclusions about its applicability are finally outlined.

Keywords: Self-adaptivity, Adaptive Architecture, Multi-Agents Systems, Agreement Technologies, Dynamic Architecture

1 Introduction

It is well known that in recent years the software systems have grown in complexity. This level of complexity, which we could call "social" according to [14], is forcing software designers to rethink the strategy for handling it. Many routine tasks previously deferred to human users are now being handled by systems themselves; including many actions related to the systems own functions. Complex systems are now able to observe themselves, and to adapt its structure and behaviour as necessary. Therefore, this approach [18] has a global influence on the system, at many levels, leading us to consider self-adaptation as a basic architectural concern [19]. Simultaneously, Multi-Agent Systems (MAS) have been developed as a generic approach to solve complex problems. They describe self-aware structures, with learning capacity and conceived to be flexible and to be able to adapt to different situations. Advances approaches use organizations to provide further structuring, taking the form of complex agent architectures. However, existing structures still have limitations in order to reach actual self-adaptivity, i.e. not only having the capability of affect their settings, but also their own composition or element types. Our approach intends to go beyond more "classic" agent technologies and propose a solution based in Agreement Technologies [1] [24] to tackle the dynamism.

This article is organized as follows: in the second section a motivating example with two scenarios is presented to illustrate main ideas and the proposed approach, which is defined as service-oriented, organization-centric and agent-based. Next section discusses the core of our approach, in which an adaptive architecture emerges within a MAS context, and some references to related work are presented. The following section presents the concept that supports the structure of these technologies, the agreement structure itself, which is defined as crosscutting five conceptual layers, and comprises the basis of Agreement Technologies [1] [24]. The agreement structure is also built-in to define the THOMAS framework [3], which implements its concepts and features, and supports further developments and experiments. This is described in section 4, as well as its evolution. Afterwards, the paper discusses the concepts and mechanisms which must be layered on top of THOMAS, to be able to generate the structures able to define emergent organizations and ultimately, adaptive architectures. The first scenario of the motivating example is then re-examined in the context of the framework, to provide a glimpse of the way in which it would implement these adaptive features. Finally, some conclusions are drawn and further lines of work are outlined.

2 Motivating Example: Two Scenarios

In order to illustrate the situation in which an adaptive architecture would be the best solution to solve complex problems, this section presents a motivating example describing two scenarios from the medical emergencies domain. This example is hypothetical but based in real situations, which are related to a demonstrator currently under development in the AT project [1]: *m-Health* (mobile-Health). Although the proposed scenarios are closely-related to medical emergencies, they may apply to any crisis. The *mHealth* is an evolutionary prototype currently under development with the cooperation of SUMMA112 [34]. This entity manages the medical emergencies in the Autonomous Region of Madrid, Spain. More details about supporting medical emergencies by using standard MAS can be found in [8].

Scenario 1. There is an emergency (E1) in the system, which then has to evolve to simultaneously react to a second one (E2).

E1. There is a fire in a large urban park situated west of central Madrid, which contains a big leisure area. There are about 500 people at that moment. About 65 people present symptoms of asphyxia, and due to climate and wind, the fire is extending to adjacent areas at a very fast pace. SUMMA112 receives information related to this emergency (E1) and decides that 5 ambulances and one helicopter are needed. The coordination with Fire Department and Police is urgent in this situation. These entities inform that they will send 3 fire trucks and 5 police cars, respectively. From an organizational approach all these elements form an organization, O1. Considering these scenario as a MAS environment, each actor maps onto an agent. So, 14 agents are interacting in the organization O1. Each agent has its role, goals and plans inside the organization. Also, every organization has its norms and protocols, which make it able to function and operate to solve the emergencies.

E2. An hour later, there is a car crash (namely E2) in a road tunnel near to the E1 location. 7 cars have crashed and initially, 2 of them are on fire. SUMMA112 decides that this emergency requires 3 ambulances and they must contact hospitals near the area. After coordinating with Fire Department and Police, they decide to send one fire truck and 3 police cars. Again, all these (initially 7) elements form a second organization, O2.

This scenario can be solved using two alternative solutions:

- Deal with O1 and O2 as *separate* elements (organizational units), with no relation between them; or
- Deal with O1 and O2 as units *with some degree of relationship*.

The second one is the most efficient and sensible approach, as it must have into account potential interactions between both emergencies. So, let's consider first O1, where all elements interact in a coordinated way to tackle the emergency E1. 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. This situation is feasible 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. Methodologically, some services that were provided by O1 are no longer required to solve E1, and are now re-mapped onto O2. Taking into account the results (or some kind of metrics) in O1, for example, the system decides if one of the fire trucks is not necessary for E1 anymore, and can be assigned to E2. The same decision can be made about 2 ambulances and 2 police cars. These decisions are efficient also because both scenarios are close to each other. This also can be seen, in other words, as if O1 gets split in two organizations: one of them continues in E1, while the other starts working for E2. Additional elements are also assigned to E2 (as described above, it would require an ambulance and a police car). The original O1, now with a smaller set of elements, continues working in E1; and a new *agreement* is created around E2, defining the O2 organization. At the same time, a larger *arrangement* is created encompassing both units. The whole system would continue adapting to changes in both emergencies, even possibly reassigning its elements again if necessary.

Scenario 2. An organization is already working in the crisis area. One of its essential services (provided by an inside organization, or even an agent) is no longer provided. This can be caused by different reasons, e.g. the agent/organization is urgently required in another emergency, or it can not reach the area due to lack of resources, etc. Given that the organization must first detect that the agent is not available and then finds an alternative solution, replace the essential service by a similar one, for example.

It is clear that organizations are dynamic in both scenarios. Therefore, it is necessary to modify their structures, configurations and coordination. Particularly in an agent-oriented environment, the goal is to achieve an *automatic reconfiguration*. The system must carry out a series of *evolutionary steps* until it finds an optimal point. This can perfectly be a continuous process, as the situation itself evolves. This example justifies why this behaviour could not be completely pre-designed; it should be *emergent* and the coordination should be achieved inside the architecture, which is essentially a service ecosystem (i.e. a set of services which were separately created but must interact and coordinate within a certain context).

The following sections present concepts and tools that have been developed in order to define the foundations of a solution, and to validate our approach.

3 Towards an Adaptive Agent Architecture

The concept of *agent* has evolved, and nowadays MAS are increasingly popular in Artificial Intelligence as an effective way to solve complex problems. Different development strategies have been proposed in order to make them flexible and to coordinate themselves in order to adapt to changing environments. However, it is believed that MAS have not had a lot of success in the industry [12] [35], probably due to a different development culture.

The proposed approach is to bridge this gap using service-oriented concepts, which are also popular in industry. Moreover, if this approach demonstrates also the desired 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.

The architecture that gives support to the model has been defined both as *open MAS* and also as a *service-oriented, organization-centric, agent-based* architecture. In the following, we will briefly describe the basic layout defined by this architecture, to later extend it to be able to describe emergent structures, describing an actual self-adaptive (still service-oriented, still agent-based) architecture.

3.1 A Service-Oriented, Agent-Based Architecture

As already noted, the proposed approach is based on service-oriented concepts, which are popular in industry. Although services technology (SOA, in particular) is established and has various standards [6] [9] [13] [20]; its methodology and influence on other paradigms (such as agent-oriented architectures) is still under development. The feature *service discovery* provides some flexibility to a service-oriented architecture, but these are strongly bound by their semantics and choreographies. *Mashups* (or web application hybrids) can be considered as exceptions, but they still mean ad-hoc solutions [21].

Since the proposed environment must be truly flexible and dynamic, it requires the use of rich semantic and highly technological capabilities. Therefore, we consider a wise use of *agents* in a broader context, with an upper layer of services added to provide, in particular, the *interoperability* feature. 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*, which in turn provides *services*. A certain implementation could define this platform as a SOA, built on top of supporting MAS.

Using agents allows us the explicit treatment of *semantics*, a *structured coordination*, the use of a methodology to service development, to structure them into *organizations*, and the use of their *learning* capacity, among others features.

At this point, we can propose a research agenda in three phases:

1. The definition of a general platform to identify the underlying *agent-based, service-oriented and organization-centric architecture*, leading to the essential platform for Agreement Technologies;
2. The introduction of further structure, to make it *adaptive*;
3. The identification of the generic adaptive structure for organizations, in the form of the *agreement* construct, and its evolution.

The central notion is that of a *service*, the basic component of the architecture is the *agent*, and the structure gluing all this together is the *organization*, conceived as a hierarchic, recursive composition of agents. 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.

3.2 Related Work: the Role of Coordination

It is perhaps better to consider coordination previously to *adaptability*. From an MAS-centred point of view, the consequences of coordination can be understood as a global influence. This can be a "shared" plan [31] or the combination of individual plans (a "multi-plan") [23]. In few words, when using MAS as a software solution, the problem of coordination is always present, and as a consequence, adaptability is also compromised.

A coordination model should cover the issues of creation and destruction of agents, their communication, and spatial distribution among them, as well as synchronization and distribution of their actions over time [10]. In a coordination system the components are *entities* (also called *coordinables*, whose interactions is ruled by the model); *media* (the abstractions that rule interactions); and *laws* (defining the behaviour of the coordination media in response to interaction) [10].

According to [26], two kinds of coordination models can be taken into account: control-driven vs. data-driven models. While the former are focused on the act of communication, the latter are focused on the information exchanged during communication. In fact, instead of two different, opposing models, these can be considered as two different forms to *observe* coordination itself.

When dealing with a tuple-space meta-model [10] [23], the entities base their interactions (cooperation, competition, among others) on tuples. The coordination, obviously, takes place in a tuple space, by producing, consuming tuples, etc. In short, it is created by *generative communication*. One step forward in the evolution of coordination allow us to deal with *self-organized systems*, which have an increasing level of internal organization between components (agents, in our case) in term of interactions, their structure, etc. [2][10]

Recently, *self-organizing coordination* [7] is defined as the management of system interactions featuring self-organising properties, namely, where interactions are local, and global desired effects of coordination appear by *emergence*. Constructively, self-organizing coordination is achieved through coordination media spread over the topological environment, enacting probabilistic and time-dependent coordination rules.

One of the most significant –and increasingly popular– approaches in the context of MAS has been to consider agents within *organizations* instead of in isolation [4]. Our own approach, indeed, is an example of this, as shown in the next section. However, this can ultimately be considered as an approach to coordination: instead of plain system-wide coordination, organizations allow for *scoped* coordination. In the same spirit, there is some kind of intrinsic relationship between self-organized coordination, as mentioned above, and our intended approach, based on defining an adaptive architecture based in the definition of *emergent* organizations – hence, self-organized, scoped coordination.

4 Agreement Technologies: the THOMAS Framework

The central notion in our approach is the *agreement* between computational entities: organizations, at the top level, but also agents, at the lower levels. The concept is conceived as an architectural construct, and it must be capable to evolve, to allow the definition of an *emergent agreement* between these entities.

4.1 Agreement Technologies

In this work we use the set of technologies and approaches globally named as Agreement Technologies [1]. The topics that must be considered to propose an agreement-based coordination can be seen as a "tower" structure. Each level of the tower provides functionality and inputs to the one above (see Figure 1). Therefore, the agreement must be seen as a layered structure, by definition. This makes sense with some intuition: when an agreement is reached, elements located at lower levels must respect it at their own level. The agents contained in an organization must comply with the terms of the agreement.

The tower structure defines the set of layers which define the conceptual essence of an agreement. These are the following:

Semantics: the bottom layer, because semantic issues influence all others. The semantic alignment of ontologies [5] must to be taken into account to avoid mismatches, as well as to have a common understanding.

Figure 1: Agreement Technologies' original Tower (layered) Structure [1]

Norms: this layer is concerned with the definition of rules determining constraints that the agreements, and the process to reach them, have to satisfy. The norms may imply structural roles affecting (or controlling) the behaviour of agents, intertwined with the semantic domain.

Organizations: they imply a super-structure that restricts the way agreements are reached by fixing the social structure of the agents, the capabilities of their roles and relationships [4].

Trust is the highest level in the structure since its mechanisms can be used by agents to summarize the history of agreements and subsequent agreements executions in order to build long-term relationships between them [33]. An agreement is built on top of the trust in order to have reliable relationships between organizations.

A wider explanation for these key concepts related to agreements between computational entities can be found in [24] and [32].

The five layers may benefit from each other; in fact, the agreement is a crosscutting structure, which maintains a bidirectional relationship to every element it contains [27]. The agreement defines the architecture: only those elements who agree to be bound are contained in the structure; but at the same time, the architecture defines the agreement: it channels the forces in the environment and provides a concrete structure, defining roles which must be filled by specific elements. The agreement is shaped by those forces, but its existence also shapes the reaction to them, and models the future evolution of the system.

In summary, this approach 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. Further refinements can still be made; though the need for meta-elements has still to be considered, nothing excludes the definition of specific agents to perform support tasks for the agreement itself (such as *sensors*, *observers*, *controllers*, *planners*, etc).

4.2 The THOMAS Framework

This subsection presents the base architecture for the technologies previously discussed which were conceived to be supported by open MAS.

Current research in the platform is oriented to achieve a greater capacity and functionality by taking advantage of MAS features, but 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*. The resulting service ecosystem will be supported, not only technologically, but also methodologically.

These concepts are built on top of the THOMAS architecture [3]. Its design can be summarized as described in the following.

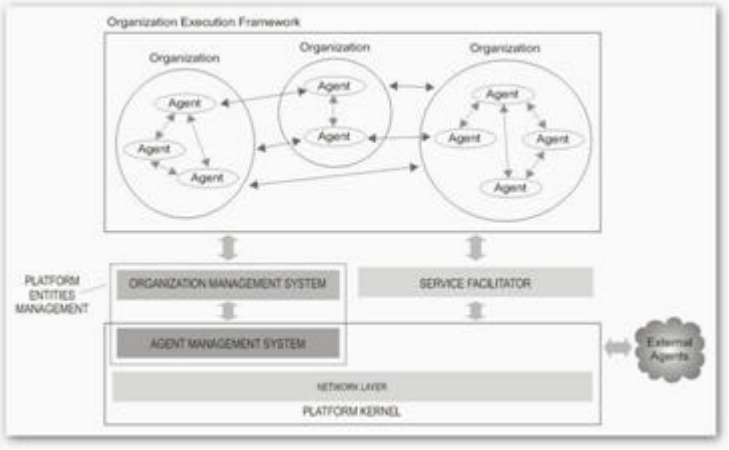

Figure 2: THOMAS Technical Architecture (inspired on [3])

The platform, including its middleware (see Figure 2), 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 [15]. 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 mentioned 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.

More details can be found in [3].

4.3 The Evolution of the Framework

The framework is under development, currently adapting to OSGi [22] specification. The main idea is to modularize applications into smaller entities called *bundles*. These entities can be installed, updated or removed on the fly and dynamically, providing the ability to change the system behaviour without ever having to disrupt its operation. The *Service Tracker* is distinguished among the services provided by this standard, especially for the proposed approach. This service lets tracking other registered services on the platform. It is used to ensure that the services to be provided are still available or not. This service is essential to face the second scenario presented in Section 2.

In a *bundle*-based system, producers of services, as well as consumers, can appear and disappear at any time. The standard provides a tool to facilitate the message passing between two entities belonging to different bundles, the *Whiteboard* pattern. This tool utilizes the service registry to maintain a list of listeners of the system, and delegates to the platform the life cycle control of the event producers and consumers. This control notifies the consumers when a producer disappears, and vice versa.

The current research, which is included as part of the OVAMAH project [25], is extending the objectives of the platform THOMAS. Besides providing the necessary technology for the development of virtual organizations in open environments, it will allow to facilitate dynamic answers for changing situations by means of the adaptation and/or evolution of the organizations. For example, agents forming an organizational unit could create (or remove) another unit, affecting the groups of the system; decide the moment to add or delete norms; the social relationship between roles could change at runtime, the conditions to activate/deactivate, as well as the cardinality of roles; the system topology (given by the relationships) could be changed also at runtime and then validate the changes with objectives and organizational type; the services could be matched to new roles; etc.

5 Defining the Emergent Agreement

When a complex problem is tackled by open MAS, the solution often requires certain adaptability. At the same time, the structure itself needs to be flexible to achieve coordination inside the system. And self-adaptation is an increasingly more important feature, as already exposed in the introduction.

The concept of *agreement among computational entities* seems to be a right approach to tackle the need for an adaptive structure. The aim of our approach is to discover a suitable structure so that it *emerges* as a global agreement. Therefore, in the following we try to define the concepts and elements which, building on the basic structures described in the previous sections would make possible to define an *emergent* organization on top of this *agreement*.

5.1 Driving Emergence: Controls and Protocols

Depending on concrete goals, any group of individuals can be arranged into certain structures (i.e. a society, architecture, hierarchy, etc.). The formation of these structures can be triggered by using two different kinds of elements, which are both based in limiting the range of available actions, namely *controls* and *protocols*.

The former, *controls*, can be seen as mechanisms that either *enforce* or *forbid* specific interactions (or architectural connections). Self-adaptive structures, being typically centralized [2], show many classic examples of this kind: most of them manifest explicit control loops, inspired in *regulators* of classic control theory.

On the other hand, *protocols*, which either *enable* or *channel* behaviour, are based on consensus and agreements. They can be described generically as the way to control decentralized (even distributed) structures [16]. Basically, when protocols are present, every agent knows the way to interact with the rest; it is necessary to comply with them to be able to communicate, but at the same time they are also *regulating* the development of the interacting structure itself.

These two mechanisms define a wide spectrum of regulation, in which agent organizations and their architectures are simultaneously harnessed by atomic, unary controls (such as *norms*, limits, *locks*, control loops or *constraints*) and multiple, connective protocols (such as *hubs*, *bridges*, *channels*, or *spaces*).

In software architecture, there are already many patterns and solutions based on implicit controls and normative constraints. On the other hand, our approach tries to base the solution on consensus: this is also an architecture-level approach, as the description of interaction is also relevant at this level of description.

It is important to note that the purpose of these mechanisms is to "discover" a suitable structure of controls and protocols so that a global structure can emerge (i.e. defining different shapes of the architecture). These elements will make possible to define the main inner structures in order to obtain agreement-based organizations. Therefore, though not a novelty, it is very relevant, as already noted in section 3, that our agents are grouped in organizations, unlike the classic plain MAS layout.

Once a primary structure can be defined, an elemental group emerges as a preliminary organization, which we will refer to as *an initiative*. This structure is explained in next subsection.

5.2 Defining an Emergent Agreement: the *Initiative*

As previously noted, a set of controls and protocols can be used to dynamically generate a preliminary organization inside a group of individuals (agents, in our approach; but also generic components).

Our approach defines and uses such a set of controls and protocols to generate certain structure (therefore several of them are considered as *generative controls* and *generative protocols*). This structure leads to an organization that grows with the environmental dynamics. The emergent organization is what we call *an initiative*: not yet fully established, but still evolving.

Nevertheless, the *initiative* can continue growing and mutating because of its *adaptive* nature, but when it has some stable structure, it can be called *organization*. This stable structure is achieved when all the participants can afford the necessary agreement in order to solve the problem or gain the main objective that caused their union. The resulting organization is then conceptually similar to other organizations in several MAS approaches, including the original THOMAS [3] itself.

The previous paragraph implies three important concepts in our approach:

- *An initiative*. It is a preliminary group of individuals (agents) which assemble in a certain structure, generated by a set of controls and protocols, as well as certain associative patterns;

- *An organization*. It is a established group; in our approach, it is dynamically originated from an initiative (though there are also static organizations; once they are created, both kinds are functionally equivalent);
- *An agreement*. It is the act by which an initiative became into a stable organization. In fact, this can be seen as the consensus which is reached between individuals inside the initial "seed" group.

This process can be seen as the system moving to a new state, in which the structure of the "past" is supplanted by a "new" emergent structure. Obviously, this novel structure admits new elements because of the dynamic environment, but now one of its goals is to reinforce its nature, and tends to perpetuate itself.

Therefore, one of the goals of an initiative is to grow; and opposing that, the main goal of an organization is to maintain itself.

Clearly, the coordination problem is always present in such fields, as presented in section 3. As the structures become more and more complex, it is clear that for some kind of problems we need that individuals organize themselves in organizations and after that, as already said, in stable, agreement-based organizations.

Let's consider our motivating example. In the first moments of the emergency E1 we can think that there are some police cars arriving to the place, but no one is the leader of the group. They follow a previous internal protocol to choose a leader (even hierarchy is a protocol), and this agreement generates a preliminary organization. This is what we call a *generative protocol*. When the individual follow this kind of protocols, they define implicit structural patterns.

An *initiative* can be generated from such patterns, named *agreement patterns*, where the term is used in an architectural sense. They are pre-designed from the required services of an initiative and the corresponding semantic refining. Some of them have been already identified (see forward Table 1), and receive such names as *Façade*, *Mediator*, or *Surveyor*, among others. (Caveat: though some of these are typical names for patterns, they are defined in a completely different context; in particular, these are *not* classic object-oriented patterns, but architectural patterns).

5.3 Self-Adaptive Organizations as *Pure Adaptation*

Concepts related to organizations in a growing process were discussed in the previous subsection, and self-organization is important due to structures construction containing the organizations. In the following, concepts related to changes suffered by organizations are presented. These organizations have reached a *quiescent* or safe state for adaptation, in a certain way. In this case, namely *pure adaptation*, the importance lies in the way that an existing organization has to adapt to a new behaviour. First, it has to realize that a change has occurred, i.e. a change can emerge in an intrinsic way [29], and then it has to adapt itself.

The second scenario in the motivating example presents an organization that is already working in the crisis area and one essential service is not available. We have identified four alternatives for adaptation:

- *Case 1* – with no modification of the organization's main objective: a search is made inside the organization, looking for a service similar to that is no longer available. The main idea is the direct replacement of the service.
- *Case 2* – with no modification of main objective: the internal search finds only a service with minimum similarity to which is no longer available. In this case, the responsible for that service must learn to answer as the one that is currently unavailable. A learning process is feasible since this is a MAS-oriented environment. The time spent in this task should be reasonable, according to the scenario characteristics.
- *Case 3* – with no modification of main objective: if the internal search fails, the organization is allowed to make an external search. This case can be considered as a state change of the organization. It comes back to the initiative level, which is maintained until reaches a quiescent or safe state, by agreement.
- *Case 4* – with change of the organization's main objective: in this case, the organization is "forced" to modify the objective, or divide it in partials goals. It is not possible to offer the original service.

These four cases are the first to be studied for a real adaptation of the organizations, due to they modify not only the structure but the type of constituent element. More cases like these are expected to develop in the medium term.

5.4 Adaptation Patterns

As already noted, the *agreement patterns* are pre-designed from the required services of an *initiative* and for the corresponding semantic refining. Some of them have been identified. According to [30] it is possible to classify the architectural design patterns as follows: monitoring (M), decision-making (DM), or reconfiguration (R) based on their objective. M and DM patterns can also be classified as either creational (C) or structural (S), as defined in [11]. Likewise, R patterns can also be classified as behavioural (B) and structural (S) since they specify how to physically restructure an architecture once the system has reached a safe state for adaptation.

Name	Category	Description
Façade	M, S	To be able to easily interact with an organization which still lacks a defined structure, some agent has to represent the organization itself in terms of interaction. This agent redirects any incoming communication; it needs not to be also a supervisor.
Mediator	R, B	During the emergence process, the organization is not yet established, and data services are probably not working. Some agent must act as a mediator, which makes possible to access to data sources, although indirectly, and also to perform the necessary translations – including, in our case, several kinds of *semantic* translations.
Surveyor	R, S	During the emergence process, at least one agent must monitor the growing of the initiative itself, both to decide when new elements are inserted, and also when the initiative stabilizes to form an organization. The surveyor has access to the pattern library – it decides when a certain pattern has been matched and must be triggered.

Table 1: Agreement Patterns: architectural design patterns.

Obviously, there are many more patterns, and not all of them describe roles. For instance, the *Surveyor Election* defines the protocol (one among many) to decide the next *surveyor*; and *Surveyor Change* describes a protocol to demote the current *surveyor* and forward its knowledge to a new one.

The patterns represent a fragment of a structure leading to a dynamic one, the initiative, reaching a stable form, the organization. As already noted, the system is ultimately conceived as a *service-oriented* architecture; so methodologically, our stable organizations must be conceived as the providers for certain high-level services. Therefore, these services must be proposed as the starting point for the functional definition of our organizations.

The functional decomposition of these services (or a hierarchical decomposition, from another point of view) will be also used to design the hierarchical structure of organizations. 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. Every (high-level) service unfolds into such a (semantic) process, which describes the coordination between lower level services; every such service is provided by a low-level organization, providing the structural decomposition from the previous, high-level organization. This process guides the (semantic) definition of any service-oriented organization, and it is used to define our structures [28].

Moreover, even our dynamic, emergent agreements *must* be consistent with these semantic definitions; so this process provides also a method to discover the required patterns for our *initiatives*. Of course, it has yet to be refined, and the desired methodology has yet to acquire a definitive form.

6 Case Study: the Motivating Example in THOMAS

In order to clarify the motivating example and show its usefulness, in this section the first scenario (and some of the corresponding event sequences) will be described, using concepts built on top of existing work.

These scenario present agents entering the system (i.e. the adaptive architecture), and playing different kind of roles, interacting within an *organizational unit*. Apart from providing domain-specific functionality, these scenarios also show generic system-level behaviour, in our case the foundations for adaptive behaviour.

As explained previously, SUMMA112, the Fire department and the Police send their own agents to attend the emergency E1. In this section, partial scenarios for these agents are presented, using a several built-in THOMAS [3] services. The protocol presents them in a basic ordering:

- **Unit Registration**: used to register a new unit (organizational unit) with a specific structure, goal and parent unit in the OMS (see subsection 4.2).

  RegisterUnit (UnitID, Type, Goal [,ParentUnitID]).

  In our example, O1 must be registered as an organization:

  RegisterUnit (O1, default, default, default)

- **Registration as a Member**. Each agent has to register as a member of the system and then join the corresponding organizational unit (see Figure 3), i.e. the emerging organization, O1. This process has two steps:
  1. The agent must register as a member of the Platform itself:

     AcquireRole(Member, Virtual)
  2. The agent registers as a domain-specific role in the organization:

     AcquireRole(Fireman, O1)

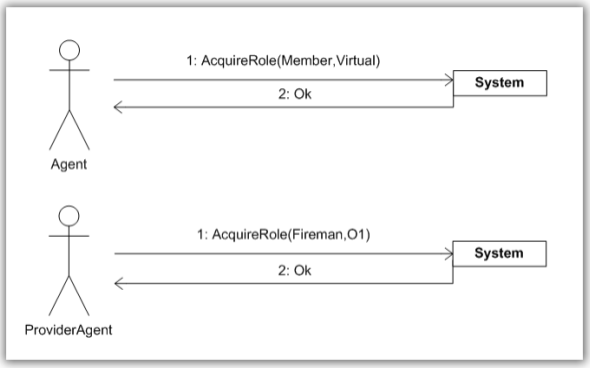

Figure 3: Role Acquisition during Fire Emergency, in THOMAS

In this way, organization O1 can add all the necessary agents to tackle the E1 emergency. O1 starts as an *initiative* until all its members have achieved the necessary agreement (i.e. internal consensus) to become a stable organization and be able to maintain its characteristics. The protocol to decide this is also a pattern: there may be many agreement establishment patterns, and the current *surveyor* is responsible to choose or decide among them – usually from environmental information.

O1 can also loose agents after they have completed their function in E1; these agents can then be added to O2 to face the E2 emergency, as already explained in the example (see section 2).

- **Expulsion**: this service is used to force an agent to leave a specific role.

Expel (AgentID, UnitID, RoleID)

Using the all of the above, consider for instance the set of actions carried out by a certain agent (let's say Fireman #1) in the E1-E2 example.

AcquireRole (Member, Virtual)	*– Activated as Fireman #1*
Hello (F#1, O1)	*– Reaches O1 and presents itself*
AcquireRole (Fireman, O1)	*– Joins O1 as a fireman*
(…)	*– … after some work…*
Expel (F#1, O1, Fireman)	*– Fireman #1 released from O1*
Hello (F#1, O2)	*– Reaches O2 and presents itself*
AcquireRole (Fireman, O2)	*– Joins O2 as a fireman*

Consider that the agent does not need to know if O1 is an already established organization (a *static* organization) or it is still growing (i.e. it is an *initiative*). In the first case it is registered as an organization within the OMS; in the second case, a façade agent gets in charge of capturing these messages. But the fireman agent, in both cases, uses system services to send the relevant messages.

In summary, the base system (the AT infrastructure implemented by the THOMAS platform) already provides the required elements to build an adaptive architecture, in terms of internal services able to join/leave an organization and define the structure according to their internal agreements. Therefore, to actually define an *emergent* agreement it is just necessary to provide the substrate to describe an actual working initiative, i.e. a library of structural agreement patterns, and the set of inter-level protocols that identify them and their roles (façade, surveyor, etc).

The developments from this scenario constitute work in progress. Using the concepts and patterns presented in the previous section, a set or structural agreement patterns are currently stored and accessed within the framework, and are being used for a number of experiments (including this one example). Both these concepts and the constructs they describe have already shown their relevance – but the current syntax and protocols cannot be considered as optimal. However, as already noted, performance is not our main concern now, but the essential dynamism and the adaptive functionality required by our architecture.

The second scenario also constitutes work in progress, and as already said in 5.3, the four cases presented are the first to be studied for a real adaptation of the organizations. We expect to develop more cases like these and their implementation in the medium term.

7 Conclusions and Future Work

This paper has explored structural concepts as the basis of an architectural approach to provide self-adaptivity to software systems. The proposed concept of *initiative* must be considered as a starting point to provide mechanisms to change the composition patterns and element types within such systems.

The required dynamism can be supported by an emergent agreement - an evolving architectural structure, based on combining predefined controls and protocols: these are handled in the context of the service-oriented, agent-based and organization-centric framework defined in AT, provided by the implementation in THOMAS platform, the OVAMAH project, and services compatible to OSGi standard.

The key idea is to create an architectural context, in which agents are *coordinated* and *reorganized* by inclusion in preliminary structures –i.e. agreement patterns– and then in stable organizations.

The platform described in Section 4, including modifications to be made by the OVAMAH project, provides services and facilities to carry out the system reconfiguration. The proposed concepts it can already be considered as a starting point to establish the necessary structures to achieve actual self-adaptivity.

Technologically, the existing (and concurrent) work is both FIPA [15] compliant, and also able to interact with JADE [17] agents. There are further developments in the works, particularly at the service level and at the agent level – affecting even performance.

Indeed, even when our approach seems promising, in the sense that is possible to achieve self-adaptation, these are the first steps.

Further work will develop and implement variants of this approach, in order to refine it. The concepts are still evolving and the process of defining their limits still continues – but even at this initial stage, the existing fragments of the approach have already proven its utility and expressive power. Current results suggest that the adaptive architecture is indeed feasible because the infrastructure developed can grow just adding new adaptive patterns. The results could fulfil the promise of generalizing the usefulness and extension of the MAS approach, adapting it to new and more agile technologies.

Acknowledgement

This work has been partially funded by the Spanish Ministry of Science and Innovation within Projects Agreement Technologies AT (CONSOLIDER CSD2007-0022, INGENIO 2010), OVAMAH (TIN2009-13839-C03-02) and MULTIPLE (TIN2009-13838); and by the EU RTD Framework Programme, through Action COST AT (COST Action IC0801).

References

1. Agreement Technologies (AT) Project: http://www.agreement-technologies.org/ (2011)
2. Andersson, J.; de Lemos, R.; Malek, S.; and Weyns, D. Modeling Dimensions of Self-Adaptive Software Systems. In Software Engineering for Self-Adaptive Systems, vol. 5525 Lecture Notes in Computer Science, pages 27-47, Springer, 2009.
3. 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).
4. 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).
5. Atienza, M., Schorlemmer, M.: I-SSA - Interaction-situated Semantic Alignment. Proc Int. Conf. on Cooperative Information Systems (CoopIS 2008).
6. 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)
7. Casadei, M., and Viroli, M.: Applying self-organising coordination to the emergent tuple organization in distributed networks. (Brueckner, S. et. al. editors) 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2008).
8. Centeno, R., Fagundes, M., Billhardt, H., and Ossowski, S.: Supporting Medical Emergencies by SMA. In Agent and Multi-Agent Systems: Technologies and Applications. LNCS, vol. 5559:823-833. Springer (2009).
9. Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C Consortium. W3C Note (2001)

10. Ciancarini, P.: Coordination models and languages as software integrators. ACM Computing Surveys, 28(2):300-302 (1996)
11. Cuesta, C.E., Fuente, P., Barrio, M., Beato, E. Dynamic Coordination Architecture through the use of Reflection. Proc. 16th ACM Symposium on Applied Computing (SAC 2001), pages 134-140, ACM Press, March 2001.
12. 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)
13. 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).
14. Fiadeiro, J. L.: Designing for Software's Social Complexity. In Computer - IEEE Computer Society, pages 34-39 (2007).
15. FIPA. FIPA Abstract Architecture Specification. Technical Report SC00001L, Foundation for Intelligent Physical Agents. FIPA TC Architecture (2002).
16. Galloway, A.R. Protocol: How Control Exists after Decentralization. MIT Press, 2004.
17. JADE - Java Agent DEvelopment Framework. http://jade.tilab.com/
18. Kephart, J.O. and Chess, D.M. The Vision of Autonomic Computing. IEEE Computer 36(1):41-50, January 2003.
19. Kramer, J. and Magee, J. Self-Managed Systems: an Architectural Challenge. In Future of Software Engineering (FOSE @ ICSE'2007), pages 259-268, IEEE, May 2007.
20. 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).
21. OMA: The Open Mashup Alliance: http://www.openmashup.org/ (2011)
22. OSGi: formerly known as the Open Services Gateway initiative, now an obsolete name. http://www.osgi.org/ (2011)
23. Ossowski, S.: Co-ordination in Artificial Agent Societies. LNAI 1535. Springer (1999).
24. Ossowski, S.: Coordination in Multi-Agent Systems: Towards a Technology of Agreement. LNCS 5244. Springer (2008).
25. OVAMAH - Organizaciones Virtuales Adaptativas: Técnicas y Mecanismos de Descripción y Adaptación http://www.cetinia.urjc.es/es/node/353 (2011)
26. Papadopoulos, G. A., and Arbab, F.: Coordination models and languages. In Zelkowitz, M. V., editor, The Engineering of Large Systems, volume 46 of Advances in Computers, pages 329-400. Academic Press (1998).
27. Pérez, J. S., Cuesta, C., and Ossowski, S.: El Acuerdo como Arquitectura Adaptativa para Sistemas Multiagente Abiertos. XV Congreso Argentino de Ciencias de la Computación – CACIC (2009).
28. Pérez, J. S., Cuesta, C., and Ossowski, S.: Agreement Technologies for Adaptive, Service-Oriented Multi-Agent Systems. II Workshop on Agreement Technologies WAT – XIII Conferencia de la Asociación Española para la Inteligencia Artificial CAEPIA (2009)
29. Prokopenko, M., Boschetti, F., and Ryan, A.J.: An Information-Theoretic Primer on Complexity, Self-Organization, and Emergence. Complexity, Vol.15-1:11-28. Wiley Periodicals, Inc. (2008).
30. Ramírez, A. J., and Cheng, B. H. C.: Design Patterns for Developing Dynamically Adaptive Systems. ICSE2010-SEAMS. Pages 49-58 (2010).
31. Rosenschein, J., and Zlotkin, G.: Rules of Encounter – Designing Conventions for Automated Negotiation among Co mputers. MIT Press (1994).

32. Sierra, C.; Botti, V.; Ossowski, S. (2010): Agreement Computing. In: KI - Künstliche Intelligenz Vol 24. ISSN: 0933-1875

33. Sierra, C., Debenham, J.: Information-Based Agency. Proc Intl. Joint Conference on AI (IJCAI-2007). AAAI Press, pages 1513-1518 (2007).

34. SUMMA112: http://www.madrid.org/cs/Satellite?language=es&pagename=SUMMA112%2FPage%2FS112_home (2009).

35. 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).