# The Agreement as an Adaptive Architecture for Open Multi-Agent Systems

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,sascha.ossowski}@urjc.es,
carlos.cuesta@urjc.es

**Abstract.** Multi-Agent Systems (MAS) are increasingly popular in AI to solve complex problems; they are conceived to be flexible and to be able to adapt to different situations. However, this feature is often compromised because of the need to reach internal coordination. This paper suggests that this issue requires an architectural solution, distilled as a new construct, the *agreement*, designed to be *adaptive*. The paper starts discussing the coordination problem and some of their traditional solutions, leading to an organization-based approach. This is followed by the description of a *base architecture*, including organizations and services, capable of supporting the agreement structure. The concept itself, and some requirements, is discussed and outlined in a set of logical layers, which are also defined as intertwining facets. Afterwards a real-world case study, in the domain of medical emergencies, is presented, describing a situation where the agreement model is applied, and some requirements to shift from directed to emergent evolution are highlighted, indicating some lines of future work.

## 1 Introduction

Intelligent Information Technologies are related to the capabilities of software systems to *learn* and *reason*, and to manage *knowledge*. This approach has known several incarnations, having the notion of *agent* as the most sophisticated among them. This concept has evolved from a single, complex "intelligent agent", to the distributed complexity of sets of multiple, simpler agents, a.k.a. Multi-Agent Systems (MAS).

In theory, a MAS should be able to find the perfect balance between the strict rules which define a reasoning system (an agent) and the adaptive nature of a evolving "society" of autonomous individuals. Where the former assumes a closed-world and a static setting, the latter provides the means to *open* the system in a flexible manner, defining a *dynamic* structure able to tackle social complexity.

However in practice, many MAS are much less flexible than at first expected. They often stabilize a concrete structure during their training phase, which has to remain unaltered, in order to maintain their reasoning capabilities, and it is therefore unable to change and evolve, limiting its adaptiveness.

From this point of view, the MAS approach would not fulfil its original promise, as it does not guarantee that the system is indeed *adaptive*; in fact, sometimes its own structure precludes that. The reason is the need for internal *coordination*; as different agents in the MAS have different agendas, only a coordinated action can ensure a consistent behaviour. The main thesis of this work is that this coordination must not be provided by agents themselves, but by the containment structure ("architecture").

The first step in this direction has already been given in many MAS proposals by providing an intermediate step in the form of *virtual organizations*, i.e. groups of agents which share common features or constraints. This way, the MAS itself can be divided into smaller fractions, with different agendas. The main idea is that agents in an organization are automatically coordinated, and therefore the level of abstraction is raised. Organizations appear as a subjacent structure (often a hierarchy), and provide a flexible and powerful means to provide adaptiveness.

Obviously, with this secondary structure simplifies adaptation: it mainly reduces to agent *reorganization*, i.e. an agent segregates from a certain organization and regroups in another with different constraints. Of course this may have consequences, not only for the agent itself, but also for the source and target organizations, which may need to adapt to the different line-up.

At the same time, this setting implies a number of additional questions. First, about the inner role of organizations in the MAS, and their functional significance; and later, about the need to provide (second-order) coordination for organizations themselves, in order 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 our approach, as well as a direct connection to SOA [16][25]; we would only refer to it briefly.

Globally, the research referred to in this paper can be summarized as the pursuit of three main goals, namely:

- To evolve the classic agent-oriented approach, from an originally closed Multi-Agent System 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.

The final purpose is to fulfil the original promise of MAS, i.e. to define a generic problem-solving *intelligent* technology, capable of being used in an *open* context, and able to *adapt* to future evolution, supported by a self-organized architecture. In this paper we do not claim to provide a final solution; but we already have a suitable base architecture, and some firm steps in that concrete direction.

The rest of this paper is structured as follows: first, the coordination problem within a multi-agent system is exposed, and historical approaches to solve it are discussed, leading to organization-based proposals. Then the Agreement Model is presented as a potential solution in this context. After that, we describe our base architecture, which includes organizations, designed to support many variants of the Agreement Model. The five main perspectives to consider in the Agreement Model are then summarized,

and some implicit features of their relationships are distilled. Finally, a real-world example is provided, to illustrate the architectural and emergent nature of the concept, both in the present and in future developments; and some further steps in these directions are briefly discussed.

## 2 The Coordination Problem

As already noted, agents were originally conceived as single actors, which had to reconcile a number of conflicting requirements on their inside. But with the advent of Multi-Agent Systems, a different method has become possible, causing the focus to change. Now a separate agent can be devoted to a different goal; and the *whole* system gets the responsibility of reconciling these views and solving the problem. The trade-off continues, but has transformed into a *coordination* problem.

To "coordinate" can be defined [29] as "to harmonize in a common action or effort". Maybe one of the most accepted definitions of "coordination" in the MAS field is taken from Organizational Science. It defines coordination as "the management of dependencies between organizational activities" [20].

From a "micro" point of view (agent-centred) [28] coordination is understood as an *environment adaptation*. If new agents come into an ambient of another agent, this has to coordinate with them, reconsidering its goals, plans, and actions, regarding the new potential interdependencies. The result of coordination for the agent, in the new (multi-agent) environment, is better if it shows a high level of cooperation to the achievement of their goals.

On the other hand, from a "macro" point of view (MAS-centred), the consequences of coordination can be conceived as something global (plan, decision or agreement). This can be a "shared" plan [23] if the agents achieve an explicit agreement during coordination process; or it just can be the summa of individual plans, sometimes called "multi-plan" [21]. At this level, results of coordination can be evaluated as a conjunction of agent goals or taking into account the MAS functionality as a whole.

In summary, when using a MAS as a software solution, the problem of coordination is always present. In fact, if we had a self-organized agent structure, we could consider this structure as *optimal*, because it would solve the coordination issues. In such a context, the hypothesis is that agents are adequately organized, i.e. "they coordinate alone", as individuals in a (human) society. This self-organization would be, at least in part, emergent. The idea is that instead of working to force individual agents to behave as the designer needs, they should be organized such that they "spontaneously" react as desired, "moving" in the right direction. In fact, this has always been the implicit, underlying promise in the MAS approach. But for this to be completely fulfilled, the agent-oriented approach must be complemented with a self-organizing coordination structure: the solution is, therefore, at the architecture level.

### 2.1 Evolution of Coordination Approaches

Pioneer work, where agents and cooperation are related, was tackled in the AI area at the end of the 70's and beginnings of the 80's. In these works *cooperation* is seen as a required concept for the intelligent solution of complex problems [2].

At the beginning of the research and developments on cooperative systems, some of the main questions were how to find *cooperative models* for every kind of problem, how to *represent* these models in order to *execute* them, and which *methods* were necessary to make the cooperative systems actually effective.

Several solutions to the cooperation problem were developed. The *blackboard* architecture [13] provides cooperation between knowledge sources using a simple communication mechanism. Each subsystem takes data from the blackboard, produces its own results and them writes down in the blackboard. This architecture is useful for problems for which no deterministic solution strategies are known.

The *contract net* [25] proposes *negotiation* as a mechanism to coordinate and to assign tasks to different entities participating in problem solving. Each entity can be a *provider* or a *requester*, and the cooperation conditions are established by a signed contract. The coordination problem is dynamically solved.

The next step was the *reactive architecture*. These works describe the intent to obtain an intelligent behaviour from simple models, without knowledge representation, reasoning or learning mechanisms [7]. The emphasis of these models is on *interactions*, from which the group behaviour *emerges* to solve problems.

Finally, *agent architectures with organizational capacity* appeared. Beside domain knowledge, agents need to know about their own *social* features. This knowledge is important for the agents because it allows them to know their role in the organization, as well as rules, and other agents' capabilities.

### 2.2 Agent-Oriented vs. Organization-Oriented MAS

MAS can be designed with a fixed organizational structure, which is composed of *static* agents. Using evolutionary or flexible agents, organizational models that mimic human organizations (teams, communities, coalitions, etc.) can be obtained. In order to achieve that, it is necessary to be able to modify such properties as the roles of agents, levels of autonomy or control mechanisms, for example.

Pure *agent-oriented* MAS methodologies (such as MAS-CommonKADS [19], Gaia [31], MaSE [30], Tropos [18] or Prometheus [22], 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. In this approach, if external agents enter into the organization, the global behaviour of the system can be seriously affected unless some kind of *control*, norms, sanctions, etc. is not established.

On the other hand, in *organization-oriented* MAS methodologies, the system organization is taken into account from the beginning. The analysis is made from a global perspective (Agent-Group-Role [15], MESSAGE [8], ANEMONA [17], AML [10], OperA [12], Civil Agent Societies [11], MOISE [16], Electronic Institutions [14], HARMONIA [27], GORMAS [3], among many 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. The system is structured by the roles, agents' interactions and communication language they use. Socials norms are very important too 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 taken into account. This feature is particularly useful to design *open* MAS.

### 2.3 Towards the Agreement Model

Generically, individuals are organized into a loose structure of any kind, either a society or an architecture, by using two different kinds of devices; both are based in limiting the range of available actions. These are *controls*, i.e. devices which either *enforce* or *forbid* specific interactions –or connections–; and *protocols*, i.e. devices 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 the right approach for this problem. 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. Therefore, an important concept, although not so new, is that agents are grouped into *organizations*, unlike the classic plain MAS layout.

As already noted, the coordination problem is always present in a MAS. 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. What we need is agents that *organize themselves* in organizations and after that, as a next level, in agreement-based organizations. The main objective is to evolve from that emergent coordination to an *emergent agreement* between entities involved in a solution.

It is important to note that the concept of *agreement* is under development, and although there are elements to define it (see Section 4), the concept is not a closed one. As the structure that it defines, agreement is in constant evolution [1].

### 3 A Base Architecture for the Agreement Model

The architecture that gives support to the model has been defined both as an open Multi-Agent System (MAS) and also as a service-oriented, organization-centric, agent-based architecture. 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*. This multi-level and multiple viewpoint nature must be specifically enabled by the technical architecture, as it must present several different notions as the key concept of the system. The logical place to provide this support is the middleware, as the platform is conceived as a distributed system.

To be both a MAS and a Service-Oriented Architecture (SOA) at the same time, one of these structures and core concepts (agent, service) must be the basis of the other. Agents supporting services has been chosen as the obvious alternative. First, agents are well-known computational entities, with an implied granularity, and need to comply with an existing standard [19]. On the other hand, services are still defining its final role and ultimate relevance; though there are already a number of standards that services could comply with [7, 12, 16, 25], these are defined at different levels of granularity, and most of them just impose restrictions on the service's interface. Therefore it is easy to conceive a service as a way to present the operational

capabilities of an agent – even better, of a collection of agents, hierarchically amalgamated in an organization. In summary, the obvious choice is to have the platform defined as a SOA, built on top of a supporting MAS.

These concepts are intended to be built on top of existing and concurrent work, such as THOMAS [4] and GORMAS [3]. A brief discussion of the features could help to achieve a better understanding of the way in which the platform and middleware would provide the required capabilities.

As can be seen in Figure 1, the THOMAS architecture, including its middleware, is structured in three levels but they are not strictly layers. They are orthogonally supported by four components, organized in different subsystems, and providing capabilities to different levels. The three levels are:

- *Platform Kernel*. The lowest level, which supports all of the above. It is the actual kernel of the middleware. It provides all the required capabilities of a FIPA-compliant architecture [19]. Therefore, at this level of description, the platform is already an open Multi-Agent System.
- *Service & Organization Management*. This refers to the conceptual level composed of the OMS and the SF components. It is not specifically marked as a subsystem in the Figure, as it lacks a unified interface.
- *Organization Execution Framework*. It is the "space" where all the computational entities "live" and perform their functions. The purpose of the other levels in the platform is to provide this one with the required capabilities.

The main components of the platform, which provide the middleware with most of its capabilities, are summarized as follows:

- *Agent Management* System (AMS). It is the component which provides all the required capabilities and functions for an agent. It guarantees FIPA compliance and is the support subsystem for the first perspective (agent-based).
- *Organization Management System (OMS)*. It is the component which provides all the required capabilities and functions for an organization. It is the support subsystem for the second perspective (organization-centric).
- *Service Facilitator (SF)*. It is the component which provides the required capabilities and functions which allow that a certain selection of the operations in an organization behave as a unified service. It is the module which provides the top-level perspective of the system, and also the support subsystem for the third perspective (service-oriented).

The Platform Entities Management subsystem does not define a layer, and as already noted it is layered itself. Its function is to provide all the required capabilities to manage every active computational entity, i.e. agents and their organizations, considered either as aggregates or independent units.

The part of the technical architecture which has received the name of Organization Execution Framework defines a *service-oriented* framework, as supported by the SF. In this "space" the services are provided by organizational units, as managed by the OMS, leading to an *organization-centric* structure. And this structure is built on top of an open MAS layer, where agents, as part of an organization, provide the required services; therefore the whole platform is, in summary, *agent-based*.
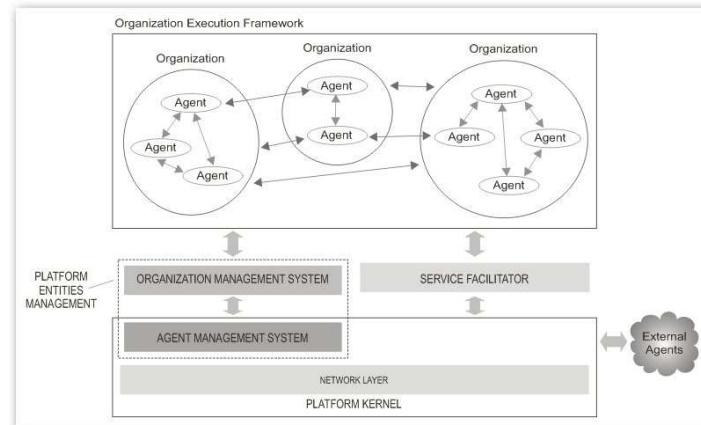
Figure 1: THOMAS Technical Architecture (inspired on [4])

Therefore, this architecture provides a satisfactory explanation of the multi-level, multi-aspect perspective which makes possible to reconcile the two different visions previously described. The provided platform presents itself as a starting point for the development of the proposed agreement-based coordination structure, and it is sufficient to test and simulate many variants of the agreement model.

### 3.1 The Role of Services

The concept of *service* is the cornerstone of the architecture. It is well known that a service has always a behavioural nature and presents a functional interface. However, it is not described by a single signature, but by a collection of them (except on the case of a unitary interface). A service is always provided by a service provider and by a concrete role within this provider.

There are basically two distinct sets of services, namely:

- *Base Services* or simply *Services*. These are user-level services, defined for every concrete application. In summary, the purpose of the platform is to serve as the support to define and use services of this kind.
- *System Services*. Those are not strictly "services" as they are not offered by a concrete user-level provider, and are usually at a lower level than a base service.

According to their function and the extent of their capabilities, however, we can identify three separate sets of services in the architecture, namely:

- *Structural Services*. These are the services which allow defining a certain organizational/architectural structure, by creating and registering organizations (units), their elements (roles) and properties (norms), and their relationships.
- *Information Services*. These are the services which provide specific information about components in an organization. They are always reflective in nature.
- *Dynamic Services*. These are the services which allow entities (agents, in particular) to dynamically enter or abandon an organization, as well as to adopt

existing roles. Differently from structural services, they do not alter (in principle) the system's style and generic architecture, its norms or vocabulary.

As noted above, there may be several implementations for the same service, described using the same interface and the same semantic profile. At the same time, each such implementation is semantically specified by a different service process model. Then in turn, that implementation, conforming to that process model, might be provided by several different service providers.

The service process model identifies three kinds of process in the structural description, namely atomic, simple and composite.

- *Atomic processes* can be directly invoked (by sending a message to a specific agent), 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. In fact, they are abstract processes, acting as placeholders; their place can be filled either by an atomic process or 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.

Therefore, the service process is also the basis for the definition of composite services. 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 orchestration-based service composition [24].

### 3.2 The Role of Organizations

Although the central concept of the architecture is that of service, the most important active element, and the unifying notion of the architecture itself, is the concept of *organization*. The organizational hierarchy is what makes possible to simultaneously define the architecture as service-oriented (at the high level) and as agent-based (at the low level).

The notion of organization defines a recursive structure. An organization is 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 an organization itself) or an autonomous nature (when the unit is also an agent). When a unit is also an *organization*, the recursive structure is unfolded.

In fact, the model can be described as organization-based, and the unit provides support for the rest of them. In this conception, the unit is the substrate which supports both the gathering of agents and the definition of services, describing the recursive plexus which explains the relationship between both perspectives.

### 4 The Agreement as a Layered, Multi-Faceted Structure

As already noted in previous sections, the central notion in our proposed approach is the *agreement* between computational entities (organizations, at the top levels; but also agents, at the lower levels), conceived as an architectural construct.

Several key research topics must be considered to propose that *agreement-based coordination*. They can be seen in a "tower" structure [1] where each level provides functionality and inputs to the one above (Figure 2). Therefore, the agreement must be seen as a *layered* structure, by definition. This makes sense with commonsense intuition: when an agreement is reached at a certain level, elements located at lower levels must respect it at their own level. If an organization reaches an agreement, the agents it contains must comply with the terms of that agreement.

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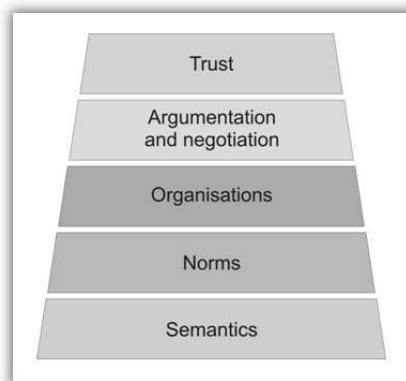


Figure 2: Agreement Technologies' original Tower (layered) Structure [1]

**Semantics**. It is the bottom layer, as semantic issues influence all others. The semantic alignment of ontologies [6] is extremely important to avoid mismatches and is needed to have a common understanding, for example, of norms. Another subject to explore is the automatic detection of the agents directly or indirectly involved in a certain interaction, for instance the current one.

The impact of semantics affects at least at two different levels, namely:

- *Semantic domain*. It gathers elements sharing the same, or at least a compatible ontology, which are therefore able to mutually understand each other,
- *Semantic interaction*. It defines the "semantic architecture", an implicit structure that takes into account semantic connections and connectors ("who do I want or need to speak with", "who am I told to speak with", etc.)

**Norms**. The next 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*. There are principles as abstractions of norms that may define a *normative domain*. They are also intertwined with the *interaction pattern* because they have some degree of consensus and have associated permissions, obligations, sanctions, etc. And of course, norms evolve and change.

**Organizations**. They are above norms, as 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 the relationship among them [5]. First, it defines "social" architectures that affects downwards and, on the other hand, organizational needs and issues may define structures, roles, dependencies, relationships, etc.

**Argumentation and Negotiation**. These can be seen as protocols that define the structure of an agreement. Negotiation between organizations forms a *large-scale agreement*, for example a convention or collective agreement; and a low-level one is a *pact* or individual agreement. A negotiation between an organization and an individual forms a medium-sized structure, as a *contract*, legal or formal agreement.

**Trust**. It is 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 [30]. A durable agreement creates a sense of trust and gives a notion of reputation. An agreement is built on top of the trust in order to have reliable relationships between organizations.

These five layers, of course, are not seen as isolated because they may well benefit from each other. A switch from the described "tower" into a multi faceted figure can be conceived because the agreement pervades (and is influenced by) all the facets/levels. In this sense, the facets are intertwined (see Figure 3), but *agreement* is still a layered structure – and in fact, layers bind both ways.
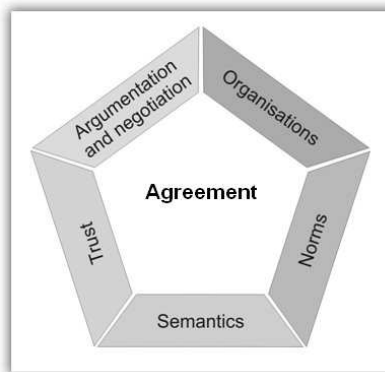


Figure 3: Multi-faceted Structure of an Agreement

In fact, the agreement is a crosscutting structure, which maintains a bidirectional relationship to every element it contains. The agreement defines the architecture: 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.

It is important to note that the multi-faceted perspective (the "pentagon") is not intended to replace the "tower" structure, as the architecture described in previous sections 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. To understand that multi-faceted structure at the implementation level, it suffices to consider that a single agent can be grouped under (or bound to) many *organizations*, even at the same time. Also, the layered structure is related to the organizational hierarchy; but even when it must be considered at the time of *publishing* services, this is not required at the time of *providing* them.

**5 Case Study: the mHealth System**

In order to illustrate both the self-adaptive nature of the agreement construct and the steps in our development methodology, this section presents a case study, describing a scenario from the medical emergencies domain. This example is related to the mHealth (mobile-Health) demonstrator, which is an evolutionary prototype currently under development in the Agreement Technologies project [1].

The main purpose of the demonstrator is the utilization and evaluation of research results in a real world scenario: the medical emergencies domain. The medical emergencies in the Autonomous Region of Madrid are managed by the SUMMA112 organization [32]. Important tasks accomplished by SUMMA112 have been used as a reference model for our analysis.

In the following we describe a real-world scenario describing an emergency (E1) in this system, which has to evolve to simultaneously react to a second one (E2). This describes how adaptiveness requires an agent *reorganization* within the agreement.

**E1**. There is a fire in Casa de Campo, the 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 following its protocols it starts to attend the sanitary emergency immediately. After analyzing the received information, SUMMA112 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. From an organizational approach all these elements form an organization, **O1**.

Considering these scenario in 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, and to solve problems.

**E2**. At this time of the year, much people travel on vacation, and there are may car crashes because of the agglomeration. One hour after the fire in emergency E1, there is a chain car crash (E2) in the tunnel of Paseo de Extremadura, a road near to E1

location. About 7 cars have crashed and initially, 2 of them are on fire. After a fast analysis SUMMA112 decides that this emergency requires 3 ambulances, and that 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.

To have into account potential interactions between both emergencies, we must consider both organizations. So, let's consider first O1, where all elements interact in a coordinated way to tackle the emergency E1. These elements reach an *agreement* to do that. At this point, the agreement construct can be defined as "the set of elements that interact 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. 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 *reorganization* within the O1+O2 composite.

Methodologically, some services which were provided by unit O1 are no longer required to solve E1, and are now re-mapped onto O2. 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.

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 *agreement* is created encompassing both units (and therefore, defining another one). This agreement would continue adapting to changes in both emergencies, even possibly reassigning its elements again if necessary.

A reconfiguration like that is manually managed; it is *directed evolution*. Ideally, and particularly in an agent-oriented environment, the goal is to achieve an *automatic reconfiguration*. The agreement (the global ecosystem) must carry out a series of reorganizations until it finds an optimal point and stops at it. To do so, it has to detect if the new configuration is *stable*. If it is not, it has to continue its evolution until it finds an optimal configuration or, at least, a local optimal point. Of course, this can perfectly be a continuous process, as the situation itself evolves.

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 3 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.

On the other side, 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.

The case of SUMMA112 shows clearly why we need to consider, not a "classic" *closed* system, not even a single-design *open* system, but a general *ecosystem*. In many senses, SUMMA112 is an independent, single institution, which could naïvely be described as a single system; but this would only describe the *internal* information system for that institution. However, it alone does *not* comprise the entire *emergency system* in Madrid, but only the *medical* emergency service. To actually provide the required response in an emergency, it has to coordinate the information systems from the Fire Department, the Police, every hospital in the area, and SUMMA112 itself. Of course these are separate information systems, which are designed and managed by different institutions and, most probably, have different origins. This implies that it is not possible to have a unified pre-*programmed* strategy to manage emergencies in the Region of Madrid, as it should be embedded in several independent systems which only sometimes gather to act together.

In summary, the example describes how to achieve coordination inside a service *ecosystem* (also, a system of systems), and also justifies why this behaviour cannot be completely pre-designed, and hence it must be *emergent*.

## 6 Conclusions and Further Work

This paper has described the implicit coordination problem which compromises the adaptiveness in the MAS approach, and has proposed to use an architecture-level construction, the *agreement*, as a solution for this problem. The main purpose of this work has been to delimitate some of the required features of this concept, as well as providing details about the base architecture to support it, its layered and multi-faceted structure, and its place in the evolution of historic approaches to this problem.

The example in the previous section succinctly describes an actual (i.e. simulated) coordination effort in the current SUMMA112 system. An existing MAS, structured in organizations, and implemented on top of the architecture described in section 3, has been used to model the system (and many others) and simulate several situations [10]. 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 our ultimate goal.

The key idea in the Agreement Model is that it creates an architectural *context*, in which agents (and organizations, and even the services they provide) 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* in charge of enforcing a self-organized system. 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, as described in section 4. The elements (agents, services, units) 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.

Further work will develop and implement variants of the agreement model, in order to refine it as necessary. The concept of agreement is still evolving and the process of defining its 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, and could fulfil the promise of generalizing the usefulness and extension of the MAS approach.

### Acknowledgements

### 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. Dignum, V.: A Model for Organizational Interaction: Based on Agents, Founded in Logic. PhD thesis, Utrecht University.
15. 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)
16. 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).

17. 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)

18. 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).

19. FIPA. FIPA Abstract Architecture Specification. Technical Report SC00001L, Foundation for Intelligent Physical Agents. FIPA TC Architecture (2002).

20. 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).

21. Giret, A.: ANEMONA: Una metodología multi-agente para sistemas holónicos de fabricación. PhD thesis, Universidad Politécnica de Valencia (2005).

22. 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).

23. 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).

24. 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).

25. 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).

26. Malone, T., Crowston, K.: The Interdisciplinary Study of Co-ordination. Computing Surveys 26 (1). ACM Press, pages 87–119 (1994).

27. Ossowski, S.: Co-ordination in Artificial Agent Societies, LNAI 1535. Springer (1999).

28. Padgham, L., and Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. In Proc. Agent Oriented Software Engineering (AOSE), 135–145 (2002).

29. Rosenschein, J., and Zlotkin, G.: Rules of Encounter – Designing Conventions for Automated Negotiation among Computers. MIT Press (1994).

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

31. Smith, R.: A Framework for Problem Solving in a Distributed Processing Environment. PhD thesis, Stanford University (1978).

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

33. The American Heritage Dictionary, Fourth Edition (2001).

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. Wood, M., DeLoach, S., and Sparkman, C.: Multiagent system engineering. Journal of Software Engineering and Knowledge Engineering 11:231–258 (2001).

37. 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).