Dealing with Adaptive Multi-Agent Organizations in the Gaia Methodology

Luca Cernuzzi[1,2] and Franco Zambonelli[1]

[1]Università di Modena e Reggio Emilia, DISMI
Via Allegri 13, Reggio Emilia, Italia
{franco.zambonelli, cernuzzi.luca}@unimore.it
[2]Universidad Católica "Nuestra Señora de la Asunción", DEI
Campus Universitario, Asunción, Paraguay

Abstract. Changes and adaptations are always necessary after the deployment of a multiagent system (MAS), as well as of any other type of software systems. Some of these changes may be simply perfective and have local impact only. However, adaptive changes to meet changed situations in the operational environment of the MAS may have global impact on the overall design. In this paper, we analyze the issue of continuous design change/adaptation in a MAS organization, and the specific problem of how to properly model/design a MAS so as to make it ready to adaptation. Following, the paper focuses on the Gaia methodology and analyzes – also with the help of an illustrative example – its suitability in supporting and facilitating adaptive changes in MASs organizations, and its advantages and limitations with this regard over a number of different agent-oriented methodologies.

Keywords: Agent Oriented Methodologies, Design for Change, Adaptive Organizations, Methodologies Evaluation

1 Introduction and Motivation

A great deal of efforts in the Agent-oriented Software Engineering (AOSE) area focuses on the definition of methodologies to guide the process of engineering complex software systems based on the multiagent systems (MAS) paradigm [6], [20]. AOSE methodologies, as they have been proposed so far, mainly try to suggest a clean and disciplined approach to analyze, design and develop MASs, using specific methods and techniques.

However, very few of the AOSE methodologies proposed so far explicitly take into account the maintenance phase of the MAS, that is, all those engineering work that has to be performed on the MAS after its deployment. In general, maintenance of a software system is required for several reasons. Corrective maintenance aims at fixing those errors that unavoidably will show up after the deployment of the system itself, independently on how extensively it was tested. Perfective maintenance is required to improve the functionalities and the performances of the system, and also to better fulfill the original requirements. Adaptive maintenance aims at tuning the software systems accordingly to changes in either the requirements or in the environment (operational or social) in which the system operates. While corrective and perfective maintenance typically have local impact only (i.e., in the case of MASs, on the internal structure of agents and on the structure of some communication protocols), adaptive maintenance may have global impact on the overall design of systems (i.e., on the overall architecture/organization of MASs).

Information systems studies outline that the phase of maintenance costs almost the 60% [1], [10] of the entire cost of systems over their lifecycle. Although there are no specific studies of this kind already available for MASs, it is reasonable to assume that MASs too will experience a similar trend, and possibly even more exacerbated as far as adaptive maintenance is concerned. While agents and MASs are often claimed as a promising approach to deal with the dynamism of modern scenarios, i.e., to deal with dynamic and open interactions and to interact in a dynamic environment, current AOSE methodologies typically promote the definition of static architecture design for the overall organization of a MAS (i.e., for the roles to be played by the agents of a system and for the relations among these roles), and are not conceived to be ready for changes in the MAS organization after its deployment.

From now to the moment in which we will be able to design and deploy – in a trustworthy and reliable way – fully autonomous and self-adaptive software systems, capable of re-organizing themselves to answer to changed conditions without any human intervention, we will probably have to wait several years. In the meantime, we may nevertheless need to better understand which the right directions to achieve this are, and we must provide engineers with suitable conceptual and practical tools to facilitate the adaptive maintenance of MASs. In other words, an AOSE methodology should not only facilitate the effective development of a MAS answering to specific requirements, but should also accompany designers through the entire software lifecycle and should facilitate developers work whenever adaptive software maintenance requires structural changes in the overall organization of a MAS.

In this paper, we focus on the design for change issue and on the issue of continuous design change/adaptation. We analyze, also with the help of a simple yet representative application example, how a MAS may require frequent and unexpected re-structuring of its global organization to adapt to changed situations. In particular, the aim of the analysis is also to outline the characteristics that an AOSE methodologies should exhibit to support the modeling and the development of adaptive MASs. The presence of such characteristics can notably reduce maintenance costs and, in the future, can facilitate the integration of self-adaptive features in MASs.

To ground the discussion, a specific attention is posed on the Gaia methodology [19], which exhibits some of the specific characteristics that make it somewhat more suitable than other methodologies to deal with adaptive changes. In particular, we show that Gaia facilitates engineers to face the likely changes that will appear in a MAS after its deployment, limiting the efforts required to re-model the evolving systems.

The following of this paper is organized as follows. Section 2 explores the need for adaptive MAS organization. Section 3 discusses the aspects of the Gaia methodology that can promote a design for change perspective. Section 4 compares with other AOSE methodologies. Section 5 concludes.

2 On the Need for Adaptive MAS

MASs, as well as the great majority of modern software systems, are likely to be subject to a large number of adaptive changes during their life-cycle, some of which may affect the very structure of the system.

In traditional software engineering discipline, special efforts have been devoted to the *design for change* issue, trying to anticipate the likely changes and adaptations that are frequently required to almost all software products after their deployments. However, those efforts have normally pointed out the anticipation of predictable changes that do not mainly influence the global design of the system under construction. Thus, it is yet an open issue how to undertake continuous design change/adaptation during the whole lifecycle of a system that may imply re-structuring its global organization. And such an issue is expected to be particularly critical for MASs, which are often conceived to operate in very dynamic operational environments.

Following in this section, we present the conference management system example (paradigmatic of a larger class of applications), illustrating how unexpected changes in the real-world organization forces important changes in the MAS organization, and discusses the requirements for an AOSE methodology to support adaptive MASs and the design-for-change perspective.

2.1 The Conference Management System Example

Let us consider an agent-based system for supporting the process of producing the technical program for an international conference. We assume the readers of this paper are mostly knowledgeable with this, but let us summarize in any case the key characteristics of this process.

The process may be subdivided into three phases:

- The *submission phase*: the program committee chair (PC Chair) and the organizer distribute the call for papers. The authors submit their papers. The papers are classified (according to specific criteria), a submission number is assigned to each paper and the authors are notified about that.
- The *review phase*: the PC Chair distributes the papers among the PC Members which are in charge of providing reviews for those papers. The PC Chair collects back reviews, decides upon the acceptance/rejection of papers, and eventually notifies authors of the decision. Considering all the accepted papers, the PC Chair prepares the conference program.
- The *publishing phase*: the authors of the accepted papers have to produce a revised version of their papers. The publisher has to collect these final versions and compose the proceedings.

The process clearly involves three loosely interacting phases, each involving different actors, and naturally leads to conceiving one MAS for supporting the activities of each phases. There, personal agents will be naturally associated to the actors involved in the process (authors, PC Chair, PC Members, reviewers) to support their work. It is also natural that the roles played by each agent reflect the ones played by the associated actor in the conference organization. This may require agents to interact both directly with each other (according to patterns that will reflect the patterns of interactions in the real-world organizations), and indirectly (via exchanges of papers and review forms).

This said, the process of designing a MAS to support the organization of a conference may at appear very simple and intuitive, as critical design choices (the types and roles of agents involved, the structure of the organizations and of inter-agent interactions) naturally derive from the structure of the real-world organization.

2.2 Unexpected Changes: a Real-World Example

What the above discussion of the conference management example misses in identifying is that, for a conference, the overall structure of the real-world organization may dramatically vary from year to year. First, since the organizers involved change from year to year, some changes in the organization may be directly induced by them based on personal attitudes and opinions. Second, factor such as the hotness of the conference topics and the effectiveness of the conference advertising may dramatically affect the number of submitted papers. Thus, the need of changing the structure of the management process may be forced by the need of keeping it manageable. This is particularly true for the reviewing phase, which involves a large number of actors, with different duties and variously interacting with each other.

To mention a real-world example, we can consider the biyearly ISAS/SCI conference series (ISAS Multiconference on Systemics Cybernetics and Informatics). ISAS/SCI started as a single mono-track conference in 1995 with 55 presented papers (the number of submitted papers being directly proportional to this). Then, the conference grew up very fast, to become a huge multi-conference and, in 2001, to reach a number of 1859 presented papers. As it can be seen from Table 1, such a pronounced growth has not occurred on a large time, and a dramatic growing is exhibited for any two consecutive editions. Personal acquaintances of the first author have confirmed that, although a continuous growth was indeed expected, no one in the organization would have expected such a dramatic trend of growth.

YEAR	NUMBER OF PRESENTED PAPERS
1995	55
1997	248
1999	754
2001	1859

Table 1. The Size of the ISAS/SCI Conference

Accordingly, to meet the increasing number of papers to deal with in the review process (as light as this can be, there is indeed some reviewing for papers in the conference), the ISAS/SCI conference had always to underwent serious and unexpected re-thinking of its organization. In 1995 it relied solely on a PC Chair and a limited group of PC Members for the review process, whose outcome were a single proceedings volume. In contrast, in 2001, there were a General PC Chair, Vice-Chairs for a large number of special-tracks (mini-conferences), each with their own PC, and hosting within a number of special-sessions, and were been published a total of 9 proceedings volumes.

In addition to the above ones, adopted by ISAS/SCI, a number of additional organizational structures can be though (and are often applied) for different conferences' sizes and characteristics. The PC Chair can partition papers among PC members which have in turn to recruit the necessary number of reviewers for their papers, or each PC member can be in charge of collecting a single review for papers. Reviewers can be asked to bid for papers, in a sort of "paper market" or can be dictated which papers to review. All of which can be organized into multi-level hierarchies on need.

What we think is most interesting in the example of conference management, is that information about what the size of the conference will be (and thus about what the most proper organizational structure to adopt is) is generally available only a few days before the review process has to begin, that is when submitted papers gets incoming. This clearly forces a dramatically fast re-structuring in the organization (unless one wants to stick to an unsuitable organizational structure) and, in the case the process is supported by a MAS, requires an extremely fast adaptation of the MAS structure. These problems, to different extents, occur in all those software systems devoted to support processes in an increasingly dynamic economy.

2.3 Requirements for Adaptive MAS and Design-for-Change

In very general terms, adaptation is the result of a bi-directional relationship between a system (e.g., a MAS organization) and the environment in which it situates (e.g., the real-world organization and its operational environment): modifications in the real-world organization or in the operational environment may imply modifications in the topological structure of MAS organization and in the control regime of its interactions.

Different studies exist that analyze the organizational aspects of MASs and their possible structures [7], some of which paying specific attention to adaptive MASs structures [13], [9]. However, a few of these studies constructively propose software engineering solutions to deal with continuously adapting MASs organizations.

Recently, several research efforts are being devoted to promote self-organization in complex software systems and, specifically, self-adaptive capabilities for multiagent systems [21]. These studies explore the possibility for complex MASs to either exploit adaptive self-organization phenomena or to promote self-inspect and self-reorganization in order to preserve specific functional and non-functional characteristics despite contingencies in the operational environment. A number of algorithms and tools are becoming available, but the time for deployment of self-adaptive software systems and MASs is far to come.

In any case, it is worth outlining that, even if effective mechanisms of self-adaptation were available, the problem of having a MAS properly capture not only internal needs of efficiency but also external needs of the stakeholders (e.g., the conference organizers in our example) is open. How can one MAS inspect and get feedbacks from the real-world organization to which it belongs to adapt accordingly? While waiting for self-adaptive MASs to come, an AOSE methodology should definitely promote a design-for-change perspective, enabling designer and developers to rapidly re-work the structure of a MAS to have it suit novel needs.

To promote such a design-for-change perspective we need *modularity* and *separation of concerns*. In particular, when dealing with both the design and development of a MAS, one should clearly separate those aspects of the system that are intrinsic to the definition of the problem itself from those that, instead, derives from contingent choices based on the actual characteristics of the operational environment and/or the real-world organization. For example, in the conference management example, this means separating those functionalities and inter-dependencies intrinsic in a process of reviewing (e.g. functionalities of PC Chair and of reviewers, and protocols for sending back review forms) from those that instead derives form a specific contingent choice (e.g., separating the role of PC Member from that of reviewer, and relying on paper bidding for assigning reviews). In that way, whenever unexpected changes occur, designers and developers are facilitated in identifying where to focus to restructure the MAS as needed without impacting on the whole system.

3 Modeling Adaptive MASs with Gaia

As far as we know, none of the currently available AOSE methodologies for MASs development explicitly accounts for a design-for-change perspective. Nevertheless, some of them already exhibit specific aspects which can at least promote a design for change. One of these, focus of this section, is the Gaia methodology [19].

3.1 Gaia in a Nutshell

Gaia focuses on the use of organizational abstractions to drive the analysis and design of MASs. Gaia models both the macro (social) aspect and the micro (agent internals) aspect of a MAS, and devotes a specific effort to model the organizational structure and the organizational rules that govern the global behavior of the agents in the organization. What makes Gaia somewhat suitable for a design-for-change perspective is its clear separation between the analysis and the architectural design phases.

The goal of the analysis phase in Gaia, covering the requirements in term of functions and activities, is to firstly identify which loosely couple sub-organizations possibly compose the whole systems and then, for each of these, produce four basic abstract models: *(i)* the environmental model, to capture the characteristics of the MAS operational environment; *(ii)* a preliminary roles model, to capture the key task-oriented activities to be played in the MAS; *(iii)* a preliminary interactions model, to capture basic inter-dependencies between roles; and *(iv)* a set of organizational rules, expressing global constraints/directives that must underlie the MAS functioning.

The above analysis models are used as input to the architectural design phase. In particular, the architectural design phase is in charge of defining the most proper organizational structure for the MAS, i.e., the topology of interactions in the MAS and the control regime of these interactions, which most effectively enables to fulfill the MAS goals. The definition of the organizational structure has to account for a variety of factors, including the need of somewhat reflecting the structure of the real-world organization in the MAS structure, the characteristics of the environment and of the patterns of access to it, the need of simplifying the enactment of the organizational rules, the need to respect any identified non-functional requirement, as well as the obvious need to keep the design as simple as possible. Once the most appropriate organizational structure is defined, the roles and interactions models identified in the analysis phase (which were preliminary, in that they were not situated in any actual organizational structure) can be finalized, to account for all newly identified interactions and possibly for newly identified roles.

Past the architectural design phase, the detailed design involves identifying: *(i)* an agent model, i.e., the set of agent classes in the MAS, implementing the identified roles, and the specific instances of these classes; and *(ii)* a services model, expressing services and interaction protocols to be provided within agent classes. The result of the design phase is assumed to be something that could be implemented in a technology neutral way.

3.2 Factors Facilitating Adaptivity in Gaia

As from the short description above, Gaia prescribes to clearly separate the analysis phase, in which the basic characteristics of the system-to-be are captured and organized, from the architectural design phase, where all the results of the analysis are put at work to identify the most suitable organizational structure. The above clear separation, together with the specific structuring of the analysis phase and of its models, are important factors to facilitate adaptive changes, according to what specified in Subsection 2.3.

The result of the analysis phase in Gaia is very modular, clearly separating basic characteristics/functionalities of the systems, (i.e., the preliminary roles and interactions models) from characteristics of the operational environment (i.e., the environmental model) and from any additional constraints that the MAS will have to respect (i.e., the organizational rules). This implies that whenever contingencies calls for a re-thinking of some of the MAS specifications, the clear separation of concerns of the Gaia analysis models is likely to avoid global re-thinking of the whole analysis and, depending on the types of contingencies, promote a local tuning of a limited set of models. For instance, some functional changes in "how" a sub-task is expected to be achieved will impact on the preliminary role model only; some changes in the global constraints the MAS has to respect implies changes in the organizational rules only.

The prescription to delay the identification of the organizational structure to the architectural design phase is also of paramount importance. In fact, more than the outcome of the analysis, it is the choice of a specific organizational structure that is more likely to be affected by contingencies. Besides properly structuring the functional requirements of the analysis phase, the choice of a specific organizational structure has to take into account and is affected by a number of non-functional requirements and by various characteristics of the operational environment and of the real-world organization. Thus, whenever contingencies call for adaptive changes in the MAS, it is very likely that these contingencies will call for a new organizational structure, which in Gaia can be selected without globally affecting the design.

In fact, the analysis outcome of Gaia is a set of preliminary roles and interactions models that exhibit no dependencies on a specific organizational structure. In the architectural design phase, after having chosen a specific organizational structure, the roles and interaction models can be finalized. Consequently, it is possible in the final roles and interaction models to clearly identify those roles and interactions which are intrinsic of the systems (i.e., those already identified from the analysis) from those that, instead, derives from the adoption of a specific organizational structure. Accordingly, whenever contingencies call for a new organizational structure, the designer is clearly facilitated in determining what parts of the system requires some sort of re-design and what parts, instead, can be left unchanged.

Thus, even if Gaia does not yet define any specific guidelines for adaptive maintenance, its structuring of the development process somewhat facilitates adaptive changes, and also enables an effective re-use of previous experiences and models. In fact, an expert designer can easily apply known organizational structures – possibly being supported by the availability of catalogues of organizational patterns – in the context of a particular system, so as to more easily chose and specify a specific organizational structure for a MAS-to-be, and – if this is the case – so as to easily re-shape the organizational structure of an existing system that requires some adaptation.

3.3 The Conference Management System in Gaia

To better ground the discussion and exemplify, we now try to put these concepts at work in the conference management system example. So, we orderly describe the various phase of the process of analysis and design of such system in Gaia,

Possible sub-organizations
As already stated, in the conference management example, three loosely coupled sub-organizations can be clearly identified, independently of the conference size. The first is the organization responsible for the submission process, the second is the organization responsible for the review process, and the third is the organization responsible for the publication of the proceedings. There are agents/roles (with specific competences) that participate in some organizations and not in others, while others like the PC Chair are likely to participate in all sub-organizations. Therefore, these three processes can dealt with by analyzing them as three separated MASs. For space reasons, hereinafter we focus on the review process only, and discuss the impact of the conference size on the actual design and on design changes.

Environmental Model
In the review process application, the environmental model simply reduces to a virtual computational environment of PDF papers (possibly enriched with XML semantic descriptions) and txt review forms. Agents can use some kind of shared database to manage the submitted papers, the reviewers information and the reviewers forms.

Preliminary Roles model
The analysis phase can clearly identify the tasks and the structure of the roles, independently of any contingent choice for the organizational structure, but based on the functional specifications only. Therefore, in the organization of the review process there exists a few clearly identifiable functional roles: the role in charge of selecting reviewers and assigning papers to them (ReviewCatcher), the role of filling review forms for assigned papers (Reviewer), the role in charge of collecting and ranking the reviews (ReviewCollector) and the role of finalizing the technical program

(DoProgram). An example of role schema for the ReviewCatcher role is presented in Figure 1.

Clearly, depending on the actual organizational structures chosen to fit the conference size, different actors (e.g., the PC Chair, the PC Members or External Reviewers) may be called to play such roles.

Role Schema: `ReviewCatcher`

Description:
This role is in charge of selecting reviewers and distributing papers among them.

Protocol and Activities:
`CheckPaperTopic`, `CheckRefereeExpertise`, `CheckRefereeConstraints`, `AssignPaperReferee`, `ReceiveRefereeRefuse`, `UpdateDBSubmission`, `UpdateDBReferee`

Permissions:

Reads	*paper_submitted*	// in order to check the topic and authors
	referee-data	// in order to check the expertise and constraint (i.e. the referee is one of the authors, or belong to the same organization
Changes	*DB Submission*	// assigning a referee to the paper
	DB Referee	// assigning the paper to the referee incrementing the number of assigned papers

Responsibilities:
Liveness:

```
  ReviewCatcher = (CheckPaperTopic.CheckRefereeExpertise.
                  CheckRefereeConstraints.AssignPaperReferee.
                  [ReceiveRefereeRefuse]|UpdateDBSubmission.
                   UpdateDBReferee)ⁿ
```

Safety:

- AssignPaperReferee =>*Referee* $\neq$ *authors* ^ *Referee_organization* $\neq$ *authors_ organizations*
- $\forall$ paper: *number_of_referees = 3*

Fig. 1. The *ReviewCatcher* functional role schema

Preliminary Protocols Model
As for the preliminary roles model, some preliminary interaction protocols may be identified to apply whatever the conference size (e.g., a protocol involving ReviewCatcher roles and Reviewers roles for assigning papers to review). However, until the organizational structure is defined, some of the protocols may remain dangling (i.e., without clearly identified roles involved) or fully unidentified.

Organizational Rules
Organizational rules in the conference management systems may dictate constraints on who can review what papers (i.e., to one to review his/her own papers), and on how the review process should proceed (i.e., by having at least three reviews by three different reviewer for each paper). Some examples of organizational rules related to the review process have been presented in [19]. Again, such rules typically express

constraints that are mostly independent from any specific internal definition of roles and that abstract from any specific organizational structure, i.e., the above rules must apply both for a small and for a large conference.

Choice of the Organizational Structure
Here comes the deal. The organizational structure is the aspect of the system that is more likely to be affected by the conference size (as already discussed in Subsection 2.2).

Let us firstly assume that the conference organizers expect a limited number of submissions, and then decide to organize the review process around a simple hierarchy (see Figure 2).

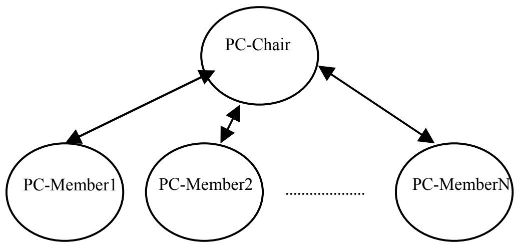

Figure 2. The *paper review organization* structure for a small conference

The PC Chair plays the ReviewCatcher role and distributes the papers among the PC Members, which simply acts playing the role of Reviewers. The PC-Members, eventually send back reviews to the PC Chair, which thus plays also the ReviewCollector role. Based on this, the preliminary roles identified in the analysis already suffice, and they can be simply organized (via properly completing the interactions model) into a hierarchy.

Completion of Preliminary Roles and Protocols Models
Once identified the organizational structure, the roles and protocol models can be finalized, by binding dangling references.

Adoption of a Different Organizational Structure
Now let us assume that the number of submissions is much higher than expected. At this point, the conference organizers may decide to adopt a different structure, i.e., a multilevel hierarchy, implying some change also in the underlying MAS supporting the process.

The multilevel hierarchy could be organized as follows. The PC Chair will have to play a new – previously not identified – role of ReviewPartitioner (see Figure 3), to partition papers "by areas" and distribute each partition to specifically appointed Vice Chairs, each in charge of handling papers in his/her area of competence. These Vice Chairs than have to act as ReviewCatcher for their assigned partitions, recruiting PC Members as Reviewers. Vice Chairs also play as ReviewCollector for their partition, and the same do the PC Chair for the whole set of reviews.

Now, what should a designer do if forced to switch from the "small conference" design to the "large conference" design? Well, due to the modularity of Gaia models and the clear separation from analysis and architectural design phase, the designer can easily re-use all previously identified models of the analysis, re-applying them in the sub-hierarchies of Vice Chairs and PC Members, and introducing the new role of "ReviewPartitioner" to define the upper level of the hierarchy.

Role Schema: `ReviewPartitioner`

Description:
This role is in charge of distributing papers among Vice-Chairs according to the area of competence.

Protocol and Activities:
`CheckPaperTopic`, `CheckViceChairArea`, `AssignPaperViceChair`, `UpdateDBSubmission`

Permissions:

Reads	*paper_submitted*	// in order to check the topic and authors
	Vice-Chair-data	// in order to check the area
Changes	*DB Submission*	// assigning the paper to a Vice-Chair area

Responsibilities:
Liveness:

```
  ReviewPartitioner = (CheckPaperTopic.CheckViceChairArea.
                       AssignPaperViceChair.UpdateDBSubmission)^ω
```

Safety:

- ∀ paper assigned to a ViceChairArea

Fig. 3. The new *ReviewPartitioner* role schema

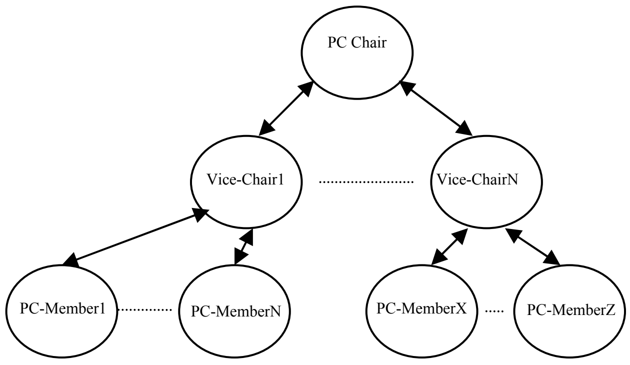

Figure 4. The *paper review organization* structure for a large conference

Detailed Design

Clearly, the detailed design of agents and services is not particularly affected by the specific organizational structure, as far as the "intrinsic" roles and interactions are concerned (Figure 5 shows the Agent model related to the reviewing process for a multilevel hierarchy organization). As far as the additional roles and interactions introduced because of a specific organizational structure are involved (e.g., the ReviewPartitioner role), these are very likely to be roles and interactions that recur over and over in the design of MAS organizations, thus making it possibly for designers to re-use from past experience of from catalogue of MAS organizational patters.

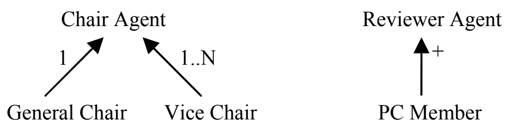

Fig. 5. The *Agent model* for a large conference

4 Other AOSE Methodologies

The issue of continuous design change/adaptation in MASs organizations has been the subject of several studies [9], [13]. For instance, the approach proposed in [14] is concerned with the agents generation at run-time in response to changes in requirements or in the environment. However, the specific problem of how to properly analyze, design, and develop a MAS so as to make it ready to adaptation is definitely under-studied.

Unlike Gaia, several other AOSE methodologies proposed in the literature simply miss in identifying a clear separation of the intrinsic aspects of a MAS (as identified in Gaia analysis) from the architectural aspects (i.e., the organizational structure in Gaia). For instance, methodologies such as Roadmap [15], Prometheus [17], MaSE [8], AOR [18], and DESIRE [2], simply consider the organizational structure to derive in an implicit way from the identification of roles/agents and of their interactions, without promoting any modularity and separation of concerns. Accordingly, even if these methodologies can "win" over Gaia for other aspects (cfr. [4]), they inherently introduce more problems in dealing with adaptive MAS.

More recently proposed AOSE methodologies explicitly face the problem of structuring the organization of the MAS, in ways different from that of Gaia, but nevertheless somewhat enforcing some degree of modularity separation of concerns that make them more suitable for adaptive change.

MASSIVE [16] focuses on organizational structures in terms of the society views and interaction views. The society view sees the MAS as a collection of agents structured according a particular organizational model. The interaction view is seen as a generalized form of conflict resolution, considering several generic form of interaction not limited to the traditional form of communication. The explicit definition of these views goes in the suggested directions of making the organizational aspects explicit, and can facilitate adaptive organizational changes.

To capture the organizational perspective, Tropos [3], [12] includes actors diagrams for describing the network of social dependency relationships among actors (modeling an agent, a role or a set of roles), and rationale diagrams for analyzing and trying to fulfill the specified goals of the actors. Also in the architectural design phase, more systems actors are introduced and goals and tasks assigned to the systems are deeper specified in term of sub-goals and sub-tasks. As already stated, this clear focus of Tropos on the definition of the organizational structure is a key requirement for promoting adaptive organizational changes.

Ingenias [12] proposes an approach which is nearest to that of Gaia in considering the organizational perspective. Moreover, it has the advantage of doing so according to a refinement approach. In the analysis-inception phase, organization models are produced to sketch how the MAS looks like (the MAS architecture). This result is refined in the analysis-elaboration phase to identify common goals of the agents and relevant tasks to be performed by each agents. In the design-elaboration phase workflows among the different agents are added to improve the organization model, and finally, in the design-construction phase social relationships of dependency (that clarify organization behavior) are defined. Again, we consider the Ingenias approach somewhat suitable in a design-for-change perspective.

The Agent Modeling Language - AML approach devotes special attention to the social/organizational aspects [5] introducing different diagrams to capture the social structure, the social behavior and the social attitudes. However, AML more than a complete methodology is a modeling language: one of its main contributions is its powerful notation being specified as a conservative extension of UML 2.0.

It is also important to highlight that most the methodologies (including Gaia) are concerned with the analysis and design processes only [4]; few are trying to cover the development and deployment of the system; less yet are concerned with the maintenance stage of the system. Thus, even when a methodology is more suitable for a design-for-change perspective, a specific attention to the maintenance process and the definition of proper guidelines for change and adaptation are lacking, which is a great limitation for modern methodologies.

As a final point, it is also worth outlining that the dynamism of modern scenarios and need of nearly continuous adaptive changes makes the traditional “waterfall” software process model, upon which most methodologies (including Gaia) explicitly or implicitly rely, very unsuitable [4]. Evolutionary process models and, more specifically, agile extreme process models may better facilitate engineers in the adaptive design maintenance of a MAS system. However, current agile and extreme software process models focus on small- to medium-size projects, and are not yet ready to tackle the complexity of developing large-scale adaptive MAS.

5 Conclusions and Future Work

In this paper, we have discussed the issue of continuous design change/adaptation that may affect a MAS during its lifetime. We used the conference management system as a representative example of adaptive MAS, to show how changes may require re-structuring the global organization of a MAS. Then, also with the help of the case study example, we have discusses how Gaia (i.e., the way in which Gaia models and organizes the identification of the organizational structure and of the rules governing the general behavior of the MAS) can to some extent facilitate engineers in tackling the likely changes that will appear in a MAS after its deployment. A comparison with other AOSE methodologies shows that other methodologies other than Gaia exhibit similar characteristics and are quite supportive of a design-for-change perspective.

Our current research work is focused on proposing more specific guidelines and conceptual tools to support engineers with in the adaptive maintenance of a MAS system, as well as, for the same purposes, in integrating in Gaia a more iterative and agile software process [4]. An additional issue that we consider very important to study relates to adaptation at the implementation level, i.e., how does changes in the design reflect in the implementation and what different problems may arise at this level that we have still not identified? The final long-term goal of these is to eventually reach a point in which we will be able to develop and deploy MASs that are able to autonomously self-adapt their behavior and to re-structure their internal organization in response to contingencies.

References

1. Boehm, B.: Software Engineering Economics, Prentice-Hall, Englewood Cliffs (NJ) (1981)
2. Brazier, F., Jonker, C., and Treur, J.: Principles of Component-Based Design of Intelligent Agents. Data and Knowledge Engineering, vol. 41, No. 2, (2002) 1-28
3. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J.: A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In: Proceedings of the 5th International Conference on Autonomous Agents. ACM Press, Montreal (Canada), (2001) 648-655
4. Cernuzzi, L., Cossentino, M., Zambonelli, F.: Process Models for Agent-based Development, Journal of Engineering Applications of Artificial Intelligence, Vol. 18, No.2, (2005) 205-222.
5. Cervenka, R., Trencansky, I. and Calisti, M.: Modeling Social Aspects of Multi-Agent Systems: the AML Approach. In this volume.
6. Ciancarini, P. And Wooldridge, M.: Agent-Oriented Software Engineering. Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering, Springer Verlag, LNCS, Vol. 1957, (2001) 1-24
7. Colman, A. and Han, J.: Organizational abstractions for adaptive systems, Technical Report No: SUTIT-TR2004.03/SUT.CeCSES-TR003, School of Information Technology, Swinburne University of Technology, June (2004)
8. DeLoach, S., Wood, M. and Sparkman, C.: Multiagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering, vol. 11, No. 3, (2001) 231-258
9. Dignum, V., Sonenberg, L., and Dignum, F.: Dynamic Reorganization of Agent Societies, In Vouros, G. (Ed.), *Proceedings of Workshop on Coordination in Emergent Agent Societies* CEAS at ECAI 2004, Valencia, Spain, 22-27 September (2004)
10. Ghezzi, C., Jazayeri, M., and Mandrioli, D.: Fundamentals of Software Engineering. Prentice Hall International, Upper Saddle River, NJ (USA) (1991)
11. Giunchiglia, F., Mylopoulos, J. and Perini A.: The Tropos Software Development Methodology: Processes, Models and Diagrams. Proceedings of Agent-Oriented Software Engineering (AOSE-2002), July 2002, Bologna (Italy), (2002) 63-74
12. Gómez-Sanz, J. and Pavón, J., 2003. Agent Oriented Software Engineering with INGENIAS. Proceedings of the 3rd Central and Eastern Europe Conference on Multiagent Systems, Springer Verlag, LNCS 2691, pp. 394-403

13. Horling, B. and Lesser, V.: A Survey of Multi-Agent Organizational Paradigms. The Knowledge Engineering Review. 2005, to appear.
14. Jayaputera, G., Zaslavsky, A. and Loke, S.: Approach to Dynamically Generated User-Specified MAS. In this volume.
15. Juan, T., Pearce, A. and Sterling, L.: ROADMAP: Extending the Gaia Methodology for Complex Open Systems. Proceeding of the First International Conference on Autonomous Agents and Multi-Agent Systems - AAMAS '02, July 15-19, 2002, Bologna (Italy), (2002) 3-10
16. Lind, J., 2001. Iterative Software Engineering for Multiagent Systems, the MASSIVE Method. Springer Verlag, New York, Secaucus, NJ, USA
17. Padgham, L. and Winikoff, M.: Prometheus: A Methodology for Developing Intelligent Agents. Proceedings of the First International Conference on Autonomous Agents and Multi-Agent Systems - AAMAS '02, Third International Workshop on Agent-Oriented Software Engineering AOSE-2002, July 15, 2002, Bologna (Italy), (2002) 135-146
18. Wagner, G.: The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. Information Systems, Vol. 28, No. 5, July, 2003, Elsevier, (2003) 475-504
19. Zambonelli, F., Wooldridge, M. and Jennings, N. R.: Developing Multiagent Systems: The Gaia Methodology. ACM Transaction on Software Engineering and Methodology, vol. 12, No. 3, (2003) 417-470
20. Zambonelli, F. and Omicini, A.: Challenges and Research Directions in Agent-Oriented Software Engineering. Journal of Autonomous Agents and Multiagent Systems, vol. 9, No. 3, Kluwer Academic Publishers, (2004) 253-283
21. Zambonelli, F. et al.: Spray Computers: Explorations in Self-organization. Journal of Pervasive and Mobile Computing, vol. 1, No. 1, (2004) 1-20