International Journal of Software Engineering and Knowledge Engineering

Adaptable Multi-Agent Systems: The Case of the Gaia Methodology

LUCA CERNUZZI

Departamento de Ingeniería Electrónica e Informática
Universidad Católica "Nuestra Señora de la Asunción"
Campus Universitario, C.C. 1683, Asunción, Paraguay
lcernuzz@uca.edu.py
http://www.dei.uc.edu.py/cernuzzi/

AMBRA MOLESINI

DEIS, Alma Mater Studiorum–*Università di Bologna*
viale Risorgimento 2, 40136 Bologna, Italy
ambra.molesini@unibo.it
http://apice.unibo.it/xwiki/bin/view/AmbraMolesini/

ANDREA OMICINI

DEIS, Alma Mater Studiorum–*Università di Bologna*
via Venezia 52, 47023 Cesena, Italy
andrea.omicini@unibo.it
http://apice.unibo.it/xwiki/bin/view/AndreaOmicini/

FRANCO ZAMBONELLI

Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia
via Amendola 2, 42100 Reggio Emilia, Italy
franco.zambonelli@unimore.it
http://www.dismi.unimo.it/Zambonelli/

Changes and adaptations are always necessary after the deployment of a multi-agent 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 new situations in the operational environment of the MAS may impact globally on the overall design. More specifically, those changes usually affect the organizational structure of the MAS. In this paper we analyze the issue of 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. Special attention is paid to the Gaia methodology, whose suitability in dealing with adaptable MAS organizations is discussed also with the help of an illustrative application example.

Keywords: Agent-oriented methodologies; adaptive/adaptable organizations; design for changes; Gaia methodology.

1. Introduction and Motivation

Maintenance of a software system is often 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 or verified. Perfective maintenance is required to improve the functionality and the performance 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 MAS (multiagent systems), 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 a system—i.e., on the overall architecture/organization of MAS.

In information systems studies [2, 19], it was considered that the phase of maintenance costs almost the 60% of the entire lifecycle of the system. Although there are no available specific studies for MAS, it seems realistic to assume that MAS too will experience a similar trend, possibly exacerbated as far as adaptive maintenance is concerned.

While agents and MAS 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 Agent-Oriented Software Engineering (AOSE) discipline presents some limitations [40]. In effect, a great deal of efforts in the AOSE area focuses on the definition of methodologies aimed at guiding the process of engineering complex software systems based on the MAS paradigm [8, 40]. AOSE methodologies, as they have been proposed so far, mainly try to suggest a clean and disciplined approach to analyze, design and develop MAS, using specific methods and techniques. However, very few of the proposed AOSE methodologies explicitly take into account the maintenance phase, that is, all those engineering work that has to be performed on the MAS after its deployment. Moreover, AOSE methodologies typically promote the definition of static architecture design for the overall organization of a MAS and are not conceived to cope with changes in the MAS organization after its deployment.

The possibility to actually design and deploy – in a trustworthy and reliable way – fully autonomous and self-adaptive software systems, capable of re-organizing themselves so as to answer to changed conditions without any human intervention, will probably take several years to be really feasible. In the meantime, we may nevertheless need to better understand the right directions to achieve this, and should then provide engineers with suitable tools – both conceptual and practical – aimed at facilitating the adaptive maintenance of MAS. In other words, an AOSE methodology should not only make it easy the effective development of a MAS answering to specific requirements, but should also accompany designers through the entire software lifecycle, and facilitate developers work whenever adaptive software

maintenance requires structural changes in the overall organization of a MAS.

Along this line, in this paper we focus on the issues of design for change and design change/adaptation. We pay special attention to the MAS organizations (which reflect the global structure of the MAS), the possible ways to adapt, and the critical issues related to the choice of a general structure. We analyze – also with the help of a simple yet representative application example – how a MAS may require re-structuring of its global organization in order to adapt to new situations. Moreover, we briefly outline the features that an AOSE methodology should exhibit to support the modeling and the development of MAS that could undergo organizational adaptations. The presence of such features could in the future facilitate the integration of self-adaptive features in MAS; therefore, we briefly discuss how some relevant current AOSE methodologies support such features.

To ground the discussion, specific attention is posed on the Gaia methodology [39], which exhibits some of the specific characteristics that make it somewhat suitable to deal with adaptive changes. In particular, we show how Gaia facilitates engineers facing the likely changes that will appear in a MAS after its deployment, limiting the efforts required to re-model the evolving systems.

Accordingly, the paper is organized as follows. Section 2 explores the need for adaptable MAS organizations. Section 3 introduces the Gaia methodology, formalizing its process according to the different phases. Section 4 discusses how the Gaia methodology promotes a design for change perspective and infers some general guidelines for AOSE methodologies. Section 5 compares with other AOSE approaches. Section 6 concludes.

2. Adaptation, MAS and Organization

MAS, as well as the great majority of modern software systems, are likely to be the subject of a large number of adaptive changes during their lifecycle. Changes may occur to role activities, to interaction protocols among roles, to introduce new role/s and consequently new interaction protocol/s, to the organization structure and perhaps introducing change to the agents which have to play specific roles, to the organization rules. Some of them may affect the very structure of the system. We use the term structure to identify the organization of the different components (mainly agents) of the system. Clearly, this term may be easily related to the term architecture as usually used in the software engineering discipline (the architecture specifies in the global system the set of modules and their interrelationships). However, in agent-based computing the term architecture has been sometimes used to specify a specific type of agent—for example BDI agents. Therefore, to avoid confusion we prefer to use the term structure in the remainder of this paper.

In the traditional discipline of software engineering, special efforts have been devoted to the issue of design for change, trying to anticipate the likely changes and adaptations required by almost all software products after their deployments. Nevertheless, those efforts have usually pointed out the anticipation of predictable changes

that regard a reduced set of components (typically agents in case of MAS), while not usually affecting 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. Such an issue is expected to be particularly critical for MAS, which are often conceived to operate in very dynamic operational environments.

Once more, some interesting insights to this problem may be found in software engineering approaches related to other application domains. For example, in object orientation paradigm we can find mature methodologies, such as the Rational Unified Process (RUP) [25], which cover in different way all the software lifecycle including the maintenance phase. Consequently, such methodologies propose methods and guidelines to cope with different types of changes that may affect the local behavior of a single component as well as the introduction of new components (modules) and indeed the architecture of the overall systems. Despite that, those interesting proposals are not considering the different fundamental abstractions characterizing MAS and consequently are less adequate to modeling and developing MAS.

The objectives of this paper are to analyze the adaptation problem for the case of MAS organizational structures, to discuss some useful features for AOSE methodologies supporting adaptation, and to analyze the Gaia suitability. Accordingly, in the rest of this section, we introduce some key concepts about adaptation in MAS organizations, we present the Conference Management System example as representative of a larger class of applications, and illustrating how unexpected changes in the real-world organization forces important changes in the MAS organization, and discusses some issues of a design for change perspective.

2.1. *Organization & Adaptation*

2.1.1. *Organization in MAS*

An organization consists in two aspects: a structural aspect (the static aspect) and a dynamic aspect. The structural aspect, usually defined at design time, includes the partitioning of the whole system into groups of agents, the relationships among those groups, and, for each group, the roles participating in with their relationships. The dynamic aspect, normally specifying the runtime behavior, is related to the patterns of interactions and rules governing roles, agents, groups of agents and the overall organization. Moreover, the MAS structure can either be explicitly designed or be the result of emergent behavior. Usually, in the case of explicitly designed structure there are goals for the society – which are external to the agent – that must be reached by the interaction of the agents—so, the desired behavior is external to the agents.

On the other hand, in emergent MAS, the organizational global behavior, which is perceivable by an external observer, is the result of a bottom-up process based on interaction and local control decisions. The structure is implicit and has a temporal determination (it is cumulative over time in a single direction, non-reversibly, and

determines the action of agents) which cannot be directly controlled by engineers or users. Thus, some authors consider the idea of adaptive MAS concerned with emergence (systems with emergent behavior). Accordingly, the organization is the result of interactions among self-organizing agents that cannot be specified "a priori" since all the unexpected changes which may arise in the environment are too much to be totally controlled.

Several types of organizational models and structures have been proposed for MAS [9]. These include command-based hierarchies, role-based organizations, norm-based ones, emergent organizations, etc. Whatever the case, any organization can be roughly described in terms of the relationships between the members of the organization (i.e., the interaction patterns of the agents in it) and the control regime of their interactions (command-based, peer-based, market-based, norm-based, etc.).

In this context, the development of a MAS should include the sensible phase of selecting an appropriate structure for the MAS. This is where AOSE discipline first becomes relevant: in order to make organization a first-class object in the engineering of MAS, agent-oriented methodologies should provide engineers with suitable organizational models (like OMNI [14] or RBAC-MAS [29]) along with a well-defined process to determine the most appropriate structure for MAS organization since the design phase.

2.1.2. *Change & Adaptation in MAS Organization*

Generally speaking, MAS are prone to different kind of changes due to their intrinsic nature. First of all, they are open systems – in which agents and organizations can evolve – typically designed for dynamic operational environments. Furthermore, they often execute to support evolving real-world organization. Depending on the specific type of MAS organizations as well as on the type and impact of the changes occurring in their operational environment, adaptation of their structure can be achieved in several way [13]: by behavioral changes at agent level; by modification of interaction patterns between agents; or by the adoption of a new organizational structure.

In general, adaptation is a relationship between a system and its environment. Adaptation of a MAS organization should reflect different situations. On one hand, situations in which the operational behavior of the organization has to change due to changes in the requirements. On the other hand, situations in which the structure of the organization changes because changes in the relationships between agents or due to the arrival/dismissing of some agents.

Different (nearly orthogonal) forces may drive the adoption of an appropriate organizational structure (i.e., of an appropriate topology and of an appropriate control regime). Such forces include: the need to achieve organizational efficiency (or, equivalently, the need for the MAS to properly handle the computation and coordination complexity of a problem); the need to respect organizational rules; and the need to minimize the distance from the real-world organization. All of which

may be in tension with the desire to keep the design simple—for a more detailed discussion on the topic please refer to [39].

Accordingly, whenever the forces that have driven the design choice of the organization structure changes (or whenever their relative weights changes), the original design choice may become inadequate. Thus, some adaptation to the organizational structure may be required to efficiently reach the purpose of the MAS. Accordingly, an AOSE methodology should not merely adopt organization as a first-class entity in the engineering of MAS, but also provide engineers with the suitable tools to accommodate changes.

Before we proceed to analyze how Gaia support adaptation in MAS organization, in the remainder of this section we first discuss some application examples, then we recapitulate the issues of design for change in MAS.

2.2. *Adaptation in MAS Organisation: Case Studies*

2.2.1. *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:

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.

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.

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 appear very simple, 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.

2.2.2. *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, factors 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 ISAS/SCI conference series [a]. 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 dramatic growing is exhibited for any two consecutive editions. Accordingly, to meet the increasing number of papers to deal with in the review process (as light as this can be), the ISAS/SCI conference had always to undergo 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 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

[a]ISAS Multiconference on Systemics, Cybernetics, and Informatics; http//:www.iiisci.org

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.

A possible way to anticipate such kind of changes is to predefine a catalogue of different organizations which may be adopted by the MAS. Nevertheless, it may be hard to really predict all possible situations.

2.3. *Design for Change in MAS*

There are two basic ways to make a system ready for adaptations. The first is to design it so that it supports a set of possible different behaviors and structures. The second way regards evolutionary approaches, usually inspired in those biology societies, which reflect self-organizing systems.

Several research efforts are being devoted to promote self-organization in complex software systems and, specifically, self-adaptive capabilities for MAS [38]. These studies explore the possibility for complex MAS 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 environmental contingencies. A number of algorithms and tools are becoming available, but the time for deployment of self-adaptive software systems and MAS is far to come.

In addition, it is worth emphasizing 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 feedback from the real-world organization to which it belongs to adapt accordingly?

While be waiting for self-adaptive MAS to arrive, 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. For this reasons, in this paper we do not explore the emergent behavior perspective while we focus on the need to design the MAS supporting different organizational structure which may be decided by designers. Consequently, organizational adaptations in the environment must be reflected in the design and then in the MAS with specific attentions to the agents interaction patterns in MAS structure.

3. The Gaia Methodology

Gaia is an agent-oriented methodology initially developed by Wooldridge et al. [36, 37], and subsequently revised and improved by Zambonelli et al. [39]. In the original version the development process underpinning Gaia was composed by only

two phases – analysis and design – and the key concepts adopted were *roles* along with the associated *responsibilities*, *permissions*, *activities*, and *protocols* [37]. Roles could interact with one another in certain institutionalized ways, which are defined in the protocols of the respective roles.

The Gaia methodology was one of the first appeared in the AOSE field, and worked a sort of benchmark for the development of the subsequent AOSE methodologies. However, the original version of Gaia was quite limited, in fact it did not take into account several issues like the design of self-interested agents, the design of dynamic and open systems, the organizational structures, and the design of cooperation protocols [37]. Such limitations are addressed in the new version of the methodology [39], where Zambonelli et al. introduced the analysis and the design of the organizational structures, the analysis of the environment, and the design-for-change perspective in order to support open and dynamic systems. These improvements led to a reformulation of the Gaia's process, including the introduction of a new phase – Architectural Design – and a number of new models.

So, in the remainder of this section we present and formalise the new Gaia's process.

3.1. *The Gaia Process*

Gaia focuses on the use of organizational abstractions to drive the analysis and design of MAS. 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 rules that govern the global behavior of the agents in the organization.

Fig. 1 represents the Gaia process by means of the Software & Systems Process Engineering Meta-model (SPEM) [30] activity diagram. SPEM 2.0 is an OMG standard meta-model for formally defining software and systems development processes [30]. The goal of SPEM 2.0 is not only the mere representation of one specific development process or the maintenance of several unrelated processes; instead, SPEM 2.0 also aims at providing process engineers with mechanisms to consistently and effectively manage whole families of related processes, thus promoting process reusability [30].

The Gaia general process is composed by three phases, as detailed in the following sub-sections.

3.1.1. *The Analysis phase*

The goal of the analysis phase (Fig. 2), which covers the requirements in term of functions and activities, is to identify the loosely-coupled sub-organizations that possibly compose the whole systems. This is done in the *Organization Definition* activity[b] depicted in Fig. 2. Then, for each of the sub-organizations, four basic abstract

[b] an activity defines basic units of work within a process [30].

models are produced by four complementary activities. In particular, in the *Environment Model* activity, engineers have to capture the features of the MAS operational environment; the *Preliminary Role Model* activity captures the key task-oriented activities to be played in the MAS in terms of preliminary roles; the *Preliminary Interaction Model* activity models the basic inter-dependencies between roles in terms of preliminary interactions; finally, the *Organizational Rules* activity identifies a set of organizational rules and norms, expressing global constraints/directives that govern the MAS behavior.

From Fig. 2 one may notice that all four activities can in principle be performed concurrently and are strictly interdependent, since the modification of one specific diagram typically affects the other diagrams, too. This strict connection among the activities is represented by a complex control flow in the activity diagram (Fig. 2).

For example, let us suppose that a new role is introduced in the sub-organization. The new role obviously modifies the preliminary role model, and may impact on both the preliminary interaction model – since the new role needs to interact with the other roles within the sub-organization – and the organizational rules—since the new role could introduce new organizational constraints. However, the modifications to the preliminary interaction model could lead to further modifications both in the preliminary role model – since the existing roles have to be tied to the new preliminary protocols introduced – and in the organizational rules—since the new protocols could generate new constraints.

3.1.2. *The Architectural Design phase*

The above analysis models are used as inputs to the Architectural Design phase (Fig. 3). 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, as done in the *Organizational Structure* activity (Fig. 3), has to account for a variety of factors, including: reflecting the structure of the real-world organization in the MAS structure, capturing the features of the environment and of its access patterns, simplifying the enactment of the organizational rules, respecting any identified non-functional requirement, as well as keeping 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 possibly newly-identified roles and interactions. This is done respectively in the *Complete Role Model* and *Complete Interaction Model* activities (Fig. 3). Again, as in the Analysis phase, these two activities are strictly related one to another because any of the modification of one of the models directly affects the other model. In fact, after the choice of the organizational structure, the designer knows which roles will have to interact with which others and which pro-

tocols will have to be executed. So, the completion of the role and protocol model activities amounts to [39]: (i) defining all the activities in which a role will be involved; (ii) defining organizational roles – those whose presence was not identified in the analysis phase –; (iii) completing the definition of the protocols required by the application, by specifying which roles each protocol will involve; (iv) defining organizational protocols.

After the conclusion of both *Complete Role Model* and *Complete Interaction Model* activities, other two activities – *Categorize Roles* and *Categorize Protocols* – are done concurrently. These two activities are crucial for the design for change perspective as explained in Subsection 4.4.

3.1.3. *The Detailed Design phase*

The detailed design involves two different activities (Fig. 4): the *Agent Model* activity that generates the agent model, i.e., the set of agent classes in the MAS, implementing the identified roles, and the specific instances of these classes; and the *Service Model* activity devoted to the generation of the 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.

4. The Conference Management System in Gaia

To better ground the discussion and exemplify the design-for-change perspective in Gaia, we adopt in the following the Conference Management System as our running example. Accordingly, we orderly describe the various phases of the process of analysis (Subsection 4.1) and design (Subsection 4.2 and Subsection 4.3) of such a system in Gaia. The discussion about adaptation in Gaia and the lessons learned from this case study follow respectively in Subsection 4.4 and Subsection 4.5.

4.1. *Analysis Phase*

4.1.1. *Sub-organizations*

As mentioned above, three loosely coupled sub-organizations can be clearly identified in the Conference Management System example, independently of the conference size: (i) the organization responsible for the submission process, (ii) the organization responsible for the review process, and (iii) the organization responsible for the publication of the proceedings. Due to limitations in space, hereinafter we focus on the review process only, and discuss the impact of the conference size on the actual design as well as on design changes.

4.1.2. *Environmental Model*

The Environment Model in Gaia prescribes the specification of all the entities and resources that the MAS can exploit, control or consume when it is working towards the achievement of the organizational goal. In our application example, the environmental model mostly reduces to a virtual computational environment made of reviewers' information, PDF papers and textual review forms, as specified in Table 2.

4.1.3. *Preliminary Roles Model*

The analysis phase can clearly identify the tasks and the structure of the roles, based on the functional specifications only, independently of any contingent choice for the organizational structure. 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 is presented in Table 3.

Every role schema is intended to be a semi-formal description of an agent's behavior. The permissions specify both what the role can and cannot use in terms of the resources available to the role. Activities are autonomous tasks or actions a role can take (i.e. CheckRefereeExpertise), and protocols are tasks or actions a role can take that involve interaction with other roles (i.e. AssignPaperReferee). There are two types of responsibilities (Table 3): *liveness* and *safety* properties. The former describe the "lifecycle" or generalized behavior pattern of the role. The latter are properties that the agent acting on the role must always preserve.

4.1.4. *Preliminary Protocols Model*

The protocol definition describes the high level purpose of the protocol, ignoring implementation details such as the sequence of messages exchanged. The protocol definition outlines the initiating role, the responding role, the input and output information, as well as a brief description of the processing the initiator carries out during the execution of this protocol. As for the preliminary roles model, some preliminary interaction protocols may be identified to apply whatever the conference size (e.g., the protocol involving `ReviewCatcher` roles and `Reviewers` roles for assigning papers to review). However, until the organizational structure is defined, some of the protocols may still be dangling – i.e., without clearly identified roles involved – or fully unidentified.

4.1.5. *Organizational Rules*

Organizational rules in the Conference Management Systems may dictate constraints on who can review which papers – i.e., to prevent a PC-Member 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 reviewers for each paper. Some examples of organizational rules related to the review process have been presented in [39]. Those rules were classified in terms of liveness and safety on roles and protocols. 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 for both a small and a large conference.

4.2. *Architectural Design Phase*

The results of Analysis phase are then refined in the Architectural Design phase which includes three models: the organizational structure, the final role model, and the final protocol model.

4.2.1. *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 2.2.2. 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 the defined roles. Hereinafter, different scenarios are presented.

First scenario

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 Fig. 5).

The PC-Chair plays the `ReviewCatcher` role and distributes the papers among the PC-Members, which simply act by playing the role of Reviewers. PC-Members eventually send back reviews to the PC-Chair, which indeed plays also the `ReviewCollector` role. Based on this, the preliminary roles identified in the analysis already suffice, and can be simply organized into a hierarchy by properly completing the interactions model. In Fig. 5, the grey zone in the PC-Chair circle means that the preliminary nature of roles in the analysis phase possibly leaves some features undefined, which however do not require further refinements in this specific case.

Alternative Scenario 1

Now let us assume that the number of submissions is higher than expected, reaching a middle size conference with a small number of Co-Chairs—two or three instead of a single PC-Chair. At this point, the Co-Chairs may decide to distribute the papers among them according to a peer-to-peer negotiation process before assigning each paper to PC-Members.

The resulting structure may be considered as an hybrid organization in which the upper level (corresponding to all the different Co-Chairs) adopts a peer-to-peer approach, while the PC-Members are hierarchically subordinated to the Co-Chairs'

paper assignments. Fig. 6 shows this alternative structure, and also show how the completion of the Co-Chair roles requires introducing some organization-dependent aspects (represented by the grey zone and the grey shaded interaction).

In this case, the GetPaper activity implies a previous negotiation process including a new protocol among the different Co-Chairs and, in this sense, depends on the choice of the particular organizational structure.

Alternative Scenario 2

Finally, 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 Fig. 7), to partition papers "by areas" and distribute each partition to a specifically appointed Vice-Chairs, each in charge of handling papers in his/her area of competence.

These Vice-Chairs then have to act as `ReviewCatcher` and as `ReviewCollector` for their partition, and the same do the PC-Chair for the whole set of reviews. In Fig. 7 the aspects that depend of the choice of a particular organizational structure are highlighted.

Gaia designers can easily re-use all previously identified models of the analysis, re-applying them (with some possible adaptations) in the sub-hierarchies of Vice-Chairs and PC-Members, introducing the new role of `ReviewPartitioner` to define the upper level of the hierarchy, and introducing the new corresponding protocol with the `ReviewCatcher`.

4.2.2. *Roles and Protocols Models*

As discussed in the above sub-section, different organizational structure may also require additional roles to be introduced—for example, as discussed in the Alternative Scenario 2, the `ReviewPartitioner` additional role should be added to the organization in the case of a large conference (see Table 4).

This is likely to influence the definition of specific roles, and also to require the definition of further protocols. Actually, the PC-Chair, playing the `ReviewPartitioner` role, needs to communicate with the different Vice-Chairs, playing the `ReviewCatcher` role, for example, to assign them the corresponding papers achieving some form of load balancing on the partitions. Thus, the GetPaper activity previously defined in the `ReviewCatcher` preliminary role needs to be modified to also include the ReceivePapersAssignement protocol along with the other liveness responsibilities: Table 5 illustrates the ReviewPapersAssignement new protocol.

Once the organizational structure has been identified, the roles and protocol models can be finalized, by complementing the preliminary models (specifying functional requirements) with organizational dependent aspects.

4.3. *Detailed Design Phase*

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 (Fig. 8 and Fig. 9 show the Agent model related to the reviewing process for a one level and a multilevel hierarchy organization respectively). 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 possible for designers to re-use from past experience or from catalogues of MAS organizational patterns.

4.4. *Discussion*

As discussed in the Conference Management System example, Gaia prescribes the clear separation between the analysis phase – where the basic features of the system-to-be are captured and organized – and the architectural design phase—where all the results of the analysis are put at work in order to identify the most suitable organizational structure. Along with the specific structuring of the analysis phase and of its models, such a clear separation is an essential factor in order to facilitate adaptive changes. The result of the analysis phase in Gaia is quite a modular one, keeping distinct the basic characteristics/functionalities of the systems, – i.e., the preliminary roles and interactions models – from the 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 call 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, to promote a local tuning of a limited set of models. For instance, some functional changes to *how* a sub-task is expected to be achieved would impact on the preliminary role model only; on the other hand, changes to the global constraints the MAS has to respect would imply changes in the organizational rules only.

The prescriptions to explicitly model the organizational structure and delay its identification to the architectural design phase are 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. Thus, whenever contingencies call for adaptive changes in the MAS, it is very likely that a new organizational structure will be required, which can be selected in Gaia without affecting the global design. Actually, as discussed in the Conference Management System example, modifications are mainly concerned with the architectural design phase, and do not affect in practice the analysis effort.

More specifically, as expected, one of the major adaptations is merely the explicit modeling of the organizational structure. If the model would be implicitly derived from the roles and their interactions – as proposed by some AOSE methodologies – accommodating the changes would likely be more complex, having to modify different analysis models of the agent system. Moreover, it is possible in Gaia to pre-define all the possible (or perhaps a great amount of the possible) alternative organizational structures as well as the contingencies that may cause a re-design. A similar proposal, reduced to architectural aspects, has been presented in [1].

In addition, it is interesting to observe that the analysis outcome of Gaia is a set of preliminary roles and interactions models that exhibit no dependencies on a specific organizational structure. As already discussed, new roles depending on different choices of organizational structures have to be included in the final role model and usually introduce some changes in the final interaction model, the agent model and the services model. However, it is easy to make a clear distinction between those functional roles and interactions that are intrinsic of the systems – i.e., those already identified from the analysis that may be considered functional roles and interactions – from those that, instead, derives from the adoption of a specific organizational structure. Such a distinction may facilitate the engineer's re-design work. Actually, whenever contingencies call for a new organizational structure, Gaia clearly helps the designer in determining which parts of the system would require some sort of re-design, and which parts, instead, could be left unchanged. As highlighted in Subsection 3.1.2 at the end of the Architectural Design phase, two crucial activities – Categorize Roles (Fig. 10) and Categorize Protocols (Fig. 11) – for the design for change perspective are done concurrently. Such activities allow the designer to clearly preserve the distinction between those features that are "intrinsic" (i.e., independent of the use of role/protocol in a specific organizational structure) – this is done respectively in the *Identify Structure Independent Role* and *Identify Structure Independent Protocol* tasks (Fig. 10 and Fig. 11), and those features that are "extrinsic" (i.e., derive from the adoption of a specific organizational structure)—this is done respectively in the *Identify Structure Dependent Role* and *Identify Structure Dependent Protocol* tasks (Fig. 10 and Fig. 11).

In our discussion, it is important to introduce some considerations about the development issue, that is, the code of the agents participating in the MAS organizations. As seen in the Alternative Scenario 1, changes in the organization structure affect the behavior of the Co-Chair agent. Consequently, the code of the Co-Chair has to be modified to implement the new GetPaper task requirements: this implies externally a negotiation process with the other Co-Chairs agents, and internally a new implementation of this function. On the other hand, in the Alternative Scenario 2 more changes have to be accommodated in the MAS. First of all, the new role `ReviewPartitioner`, the Vice-Chairs agents, and the new protocol between PC-Chair and Vice-Chairs are introduced. The Vice-Chairs act as the PC-Chair defined for the small conference scenario. Thus, they do not have to suffer changes at the coding level. Conversely, the `ReviewPartitioner` role implies the develop-

ment of new tasks (activities and protocols). However (see Table 3 and Table 4), many of such tasks are exactly the same of those of the PC-Chair agent defined for the small conference scenario—i.e. GetPaper, CheckPaperTopic and UpdateDBSubmission); also, the others – i.e. CheckViceChairArea and AssignPaperViceChair – are really similar to the previously defined ones—i.e. CheckRefereeExpertise and AssignPaperReferee. Indeed, it is possible to re-use a lot of the previously-written code, just introducing minor changes. For the new protocol between PC-Chair and Vice-Chairs, analogous consideration to those of Alternative Scenario 1 could be done. Therefore, we may conclude that both intra- and inter-agents coding changes are well separated from the other code and do not require a relevant re-codification of different modules of the MAS.

Summarizing, even though Gaia does not yet provide 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, models, and code. In fact, an expert designer could 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 choose and prescribe 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 adaptations. Moreover, such changes in the structure may be accommodated without requiring a great amount of re-work in the code development.

Despite these positive considerations, there are some open issues. An important aspect concerns the requirement elicitation for the organization. While Gaia starts the analysis phase by focusing on the possible sub-organizations, it does not pay specific attention to requirements elicitation that may probably influence the refinement process in the analysis and design phase. Furthermore, the process adopted by Gaia does not require too many steps from analysis to detailed design; nevertheless it is quite linear (a waterfall like), whereas a more incremental and agile process may facilitate the accommodation of adaptive changes whenever needed. In addition, Gaia does not prescribe any specific notation for modeling the organizational structure. This lack does not promote standardization, and may lead to ambiguities in the interpretation of the model. Finally, many of the issues discussed in this work focus on adaptable systems, while it is not totally clear what happens when engineers try to extrapolate them to fully self-adaptive systems.

4.5. *Lesson Learned for the Design-for-Change Perspective*

From the previous representative example and the discussion about Gaia, some general recommendations could be inferred, which should be taken into account in situations of adaptive changes. To promote the design-for-change perspective – and consequently choosing a different organization whenever circumstances claim for changes – we need to consider a number of different aspects. The explicit modeling

of relevant abstractions for organizations (i.e. organizational structure and control regime), the application of principles like modularity and separation of concerns, the adopted process of the methodology, among other factors, may help designers in this work. Actually, it is necessary to explicitly prescribe specifications of such aspects that are likely to undergo changes. In this sense, in the case of organizational changes we need to explicitly model the organizational structure of the MAS as well as the control regime [17] which may be affected by the circumstances. Without those models, the adoption of pre-designed solutions, such as a catalogue of organizations structures and/or organizations patterns of interactions among agents, becomes unfeasible. Considering the relevance of this aspect, one of the open issues concerns the notations adopted for such models: in our opinion, however, this is an open issue for all the methodologies, not just for Gaia. The challenge here is the need to combine the benefits of organizational theory notations with the representation of MAS abstractions. In this sense, some interesting studies propose the use of UML-based notations for modeling the social/organizational aspects including the organization structure. It may be useful to highlight that Gaia has already been extended adopting UML based notations [6]: however, such extension does not take into account the organizational structure model.

Moreover, some relevant principles from traditional software engineering have to be considered. Among them, the most relevant are modularity and separation of concerns—in our case, applied to the organizational perspective. 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, derive from contingent choices based on the actual features of the operational environment. This means that the analysis should not be affected by changes in the architectural/organization design. For example, in the Conference Management, this means separating those functionalities and inter-dependencies that are intrinsic in the process of reviewing – e.g. functionalities of PC-Chair and of reviewers, and protocols for sending back review forms – from those that instead derive 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. Additionally, separation of concerns should be applied in order to differentiate inter-agent organizational aspects from intra-agents ones. This separation may facilitate restructuring the MAS organization without re-thinking all of its roles and agents. In that way, whenever unexpected changes occur, designers and developers could more easily identify where to focus in order to properly restructure the MAS as required without impacting on the whole system.

Finally, the process proposed by the adopted methodology may strongly influence the effectiveness of accommodating the needed adaptations. Actually, if a methodology prescribes too many steps to define an organizational structure or specify interactions patterns that influence the general behavior of the MAS, this apparently reduces flexibility for adaptive changes.

5. Related AOSE Approaches

The issue of continuous design change/adaptation in MAS organizations has been the subject of several studies [9, 13, 18, 21, 23]. For instance, the approach proposed in [23] is concerned with the agents generation at run-time in response to changes in requirements or in the environment. However, the specific problems 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 the Gaia analysis – from the architectural aspects—i.e., the organizational structure in Gaia. For instance, methodologies such as Roadmap [24], Prometheus [31], MaSE [11], AOR [35], and DESIRE [3], simply consider the organizational structure to derive in an implicit way from the identification of roles/agents and of their interactions, promoting neither modularity nor separation of concerns. Clearly, as discussed in the Subsection 2.3, whenever methodologies do not prescribe to explicitly model the organizational structure of the MAS, accommodating their likely changes might be difficult. Accordingly, even if these methodologies may "win" over Gaia for other aspects (cfr. [5]), they inherently introduce more problems in dealing with adaptive changes in MAS.

Furthermore, as stated in Subsection 2.3, the application of some principles from the traditional software engineering – i.e. design-for-change, modularity, and separation of concerns – could make AOSE methodologies more suitable for adaptive changes. Indeed, some 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 and separation of concerns. Among them, Tropos, INGENIAS, ADELFE and SODA are four of the most interesting (still, similar considerations may be done for other interesting methodologies like MASSIVE [27]).

To capture the organizational perspective, Tropos [4, 20] 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, the clear focus of Tropos on the definition of the organizational structure is a key requirement for promoting changes. However, Tropos would need to explicitly model the topology in order to better help agent engineers to accommodate adaptive organizational changes. Additionally, in Tropos the separation of concerns for the organizational perspective is not well defined and possibly implies global re-thinking of the whole analysis to accommodate adaptive changes.

INGENIAS [32] 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 agent. In the design-elaboration phase, workflows among the different agents are added to improve the organization model; finally, in the design-construction phase, social relationships of dependency are defined, which clarify organization behavior. Again, we consider the approach somewhat suitable in a design for change perspective. Nevertheless, the process proposed by INGENIAS requires too many steps (to pass again from the analysis-inception up to the design-elaboration phases) in order to obtain a suitably-modified organization model.

ADELFE [18] proposes an approach where adaptive software is used in situations in which the environment is unpredictable or the system is open. To solve these problems ADELFE guarantees that the software is developed according to the AMAS (Adaptive Multi-Agent System) theory. According to this theory, building a system which is functionally adequate (which realizes the right desired global function) is achieved by designing agents with a cooperation-driven social attitude. Agents composing an AMAS ignore the global function of the system, only pursue a local goal and try to always keep cooperative relations with one another: they are called *cooperative agents*. In such a framework the emphasis is posed on the interaction rather than on the structure of the organization, which is instead our main focus here. Thus, transforming the MAS organization in ADELFE requires changes in the combination of the partial functions (agent adaptations more than organizational structure adaptation).

In SODA [28] the design-for-change perspective is supported by the *layering principle* and the *Carving* operation. The layering principle consists of two mechanisms, *zoom* and *projection*: zoom makes it possible to pass from an abstract layer to another, while projection projects the entities of a layer into another. Zoom is the only mechanism relating layers to each other: more precisely, in-zooming the entities of a (more abstract) layer leads to a more detailed layer, while out-zooming the entities of a (more detailed) layer leads back to a more abstract layer. Indeed, the *Carving* operation represents a sort of boundary between the Architectural Design and the Detailed Design, where the chosen system architecture is "carved out" from all the possible architectures. This is a crucial choice because it fixes the specific system architecture and, as a consequence, constrains both the detailed design and the implementation. So, the layering principle allows different system architectures to be managed in a uniform way: any change in the organizational structure implies – in principle – only a different *Carving* operation and consequently some modifications in the Detailed Design step.

Furthermore, even though not directly concerned with the agent paradigm, works on the "high-variability design" [26] propose an alternative approach based on goal-oriented background. Nevertheless, those approaches propose a non-direct way to introduce adaptation to the structure – i.e., organizational – level. Actu-

ally, a developer/designer has to start again from the requirement specification and perform a number of steps in order to model the new organization as the result of actors' dependencies.

There are some other interesting efforts in the AOSE arena which devote special attention to the organizational perspective. The Agent Modeling Language (AML, [7]) and MAS-ML [33] approaches devote special attention to the social/organizational aspects by introducing different diagrams to capture the social structure, the social behavior and the social attitudes. However, both AML and MAS-ML more than complete methodologies are modeling languages which main contribution is their powerful notation being specified as a conservative extension of UML, but they cannot offer to agent designer all the relevant aspects identified for the design for change perspective. In addition, Coutinho et al. [10] present an interesting study comparing different modeling techniques to represent social organization of a MAS such as AGR [16], MOISES+ [22], ISLANDER [15], and OMNI [14]. According to the authors' comparison, the more relevant modeling techniques are OMNI and MOISES+.

OMNI (Organizational Model for Normative Institutions) [14] is a framework for norms, structure, interaction and ontologies for modeling MAS organizations. It is a unification of two other proposals: OperA [12] and the HarmonIA framework [34]. OMNI considers three different levels of abstraction, namely abstract, concrete and implementation, for each of the three different dimensions: Organizational, Normative, and Ontological. In many aspects OMNI suggests to model the same abstractions proposed in Gaia, and also mimic the Gaia notations—see for example the role description and the interaction scene. With regards to the organizational structure, OMNI introduces the idea of Groups as well as two additional models: the Role Hierarchy graph and the Role Dependencies graph. These features may help to better capture the structure of the MAS. This methodology devotes special effort to the norms, rules, violations and corresponding sanctions. OMNI introduces some kind of modularity and separation of concerns considering three different perspectives. Nevertheless, the architectural design (organizational structure) is not clearly separated to the functional aspects specified by means the roles. Thus, possible changes in the structure may implies a relevant re-work for all the analysis. Moreover, in general, OMNI tends to introduce too many details and steps (across the three level of abstraction) to obtain the final specification of each dimension, especially for the Normative one.

MOISES+ (Model of Organization for multI-agent SystEms) [22] explicitly distinguishes three aspects in the modeling of a MAS organization: the structural, the functional and the deontic aspects. The structural aspect defines the agents' relations through the notions of roles, groups of roles and links – i.e. communication and authority delegation – among roles or groups. Moreover, it is possible to define interesting relationships between roles such as composition and inheritance. On the other hand, the deontic aspect describes the permissions and obligations for

the roles. Consequently, MOISES+ explicitly models the structure and the control regime by applying, to some extent, a separation of concerns between the structural and the functional dimensions. Nevertheless, MOISES+ does not consider some relevant abstractions for MAS like the environment and the agents' interaction with it. Moreover, this model is not meant to works as a complete methodology providing a clear general process and methodological guidelines for each model in the framework. Thus, it may be uneasy to adapt the MAS according to changes in the organizational structure without clear insight on the MOISES+ process.

From the software engineering process point of view, it is important to highlight that most of the methodologies (including Gaia) are concerned with the analysis and design processes only [5]; 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 AOSE methodologies.

As a final point, it is also worth outlining that the dynamism of modern scenarios along with the need of almost-continuous adaptive changes makes the traditional "waterfall" software process model – upon which most methodologies (including Gaia) explicitly or implicitly rely – very unsuitable [5]. 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.

6. Conclusions and Future Work

In this paper, we have discussed the issue of design change/adaptation that may affect a MAS during its lifetime, by focussing on the organizational aspects. 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, we have discusses how the Gaia methodology – or, more precisely, 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 organization after its deployment. Furthermore, we infer some of the main characteristics that agent-based methodologies should feature in order to deal with such kind of adaptation, that is, to explicitly model the organizational structure and organizational rule of the MAS-to-be, as well as the adoption of relevant software engineering principles such as modularity and separation of concerns. Also, we discussed other related contributions from the traditional software engineering, from methods or frameworks for specific domains other than agents, and also from other AOSE methodologies which devote special

interest to the organizational perspective.

Our current research work is focused on proposing more specific guidelines and conceptual tools to support engineers 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 [5]. Another very important issue concerns 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 our research is to eventually reach a point in which we will be able to develop and deploy MAS that autonomously self-adapt their behavior and re-structure their internal organization in response to contingencies.

References

[1] R. Allen, D. Garlan, and R. Douence. Specifying dynamism in software architectures. In *Proceedings of the Workshop on Foundations of Proceeding of the Workshop on Foundation of Component-Based Software Engineering*, pages 11–22, Zurich, Switzerland, September 1997.

[2] K.-D. Althoff, A. Birk, and C. Tautz. The experience factory approach: Realizing learning from experience in software development organizations. In *10th German Workshop on Machine Learning (FGML'97)*, pages 6–8. University of Karlsruhe, 1997.

[3] F. M. T. Brazier, C. M. Jonker, and J. Treur. Principles of component-based design of intelligent agents. *Data Knowledge Engineering*, 41(1):1–27, 2002.

[4] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[5] L. Cernuzzi, M. Cossentino, and F. Zambonelli. Process models for agent-based development. *Engineering Applications of Artificial Intelligence*, 18(2):205–222, Mar. 2005.

[6] L. Cernuzzi, T. Juan, L. Sterling, and F. Zambonelli. The gaia methodology. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems. The Agent-Oriented Software Engineering handbook.*, chapter IV, pages 69–88. Kluwer Academic Publishers, first edition edition, 2004.

[7] R. Cervenka, I. Trencansky, and M. Calisti. Modeling social aspects of multi-agent systems: The aml approach. In J. P. Müller and F. Zambonelli, editors, *Agent-Oriented Software Engineering VI*, volume 3950 of *LNCS*, pages 28–39. Springer Berlin / Heidelberg, 2006. 6th International Workshop (AOSE 2005), Utrecht, The Netherlands, 25–26 July 2005. Revised and Invited Papers.

[8] P. Ciancarini and M. J. Wooldridge. Agent-oriented software engineering. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering*, volume 1957 of *LNCS*, pages 1–24. Springer, 2001.

[9] A. Colman and J. Han. Organizational abstractions for adaptive systems. Technical Report SUTIT-TR2004.03/SUT.CeCSES-TR003, School of Information Technology, Swinburne University of Technology, 2004.

[10] L. R. Coutinho, J. Sichman, and O. Boissier. Modeling organization in mas: a comparison of models. In *Proc. of the 1st. Workshop on Software Engineering for Agent-Oriented Systems (SEAS'05)*, Uberlândia, Brazil, October 2005.

[11] S. A. DeLoach, M. F. Wood, and C. H. Sparkman. Multiagent systems engineering. *In-*

ternational Journal of Software Engineering and Knowledge Engineering, 11(3):231–258, 2001.

[12] V. Dignum. *A Model for Organizational Interaction, based on Agents, founded in Logic*. PhD thesis, University of Utrecht, 2004.

[13] V. Dignum, L. Sonenberg, and F. Dignum. Dynamic reorganization of agent societies. In G. A. Vouros, editor, *International Workshop on Coordination in Emergent Agent Societies (CEAS'04)*, pages ?–?, ECAI 2004, Valencia, Spain, 23–24 Aug. 2004. Proceedings.

[14] V. Dignum, J. Vázquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, editors, *Programming Multi-Agent Systems*, volume 3346 of *Lecture Notes in Computer Science*, pages 181–198. Springer Berlin / Heidelberg, 2005. Second International Workshop ProMAS 2004, New York, NY, July 20, 2004, Selected Revised and Invited Papers.

[15] M. Esteva, J. A. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J. C. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *Lecture Notes in Computer Science*, pages 348–366. Springer Berlin / Heidelberg, 2002. 8th International Workshop, ATAL 2001 Seattle, WA, USA, August 1-3, 2001, Revised Papers.

[16] J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In P. Giorgini, J. P. Müller, and J. Odell, editors, *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer Berlin / Heidelberg, February 2003. 4th International Workshop on Agent Oriented Software Engineering, AOSE 2003, Melbourne, Australia, July 15, 2003. Revised and Invited Papers.

[17] M. S. Fox. An organizational view of distributed systems. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(1):70–80, Jan. 1981.

[18] J.-P. Georgé, B. Edmonds, and P. Glize. Making self-organising adaptive multiagent systems work. In F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 16, pages 319–338. Kluwer Academic Publishers, June 2004.

[19] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall International, Upper Saddle River, NJ, USA, 1991.

[20] F. Giunchiglia, J. Mylopoulos, and A. Perini. The tropos software development methodology: Processes, models and diagrams. In F. Giunchiglia, J. Odell, and G. Weiß, editors, *AOSE*, volume 2585 of *Lecture Notes in Computer Science*, pages 162–173. Springer Berlin / Heidelberg, January Agent Oriented Software Engineering III. Third International Workshop, AOSE 2002 Bologna, Italy, July 15, 2002 Revised Papers and Invited Contributions.

[21] B. Horling and V. Lesser. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316, 2004.

[22] J. F. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In G. Bittencourt and G. L. Ramalho, editors, *Advances in Artificial Intelligence*, volume 2507 of *Lecture Notes in Computer Science*, pages 118–128. Springer Berlin / Heidelberg, London, UK, August 2002. 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002 Porto de Galinhas/Recife, Brazil, November 11–14, 2002 Proceedings.

[23] G. Jayaputera, A. Zaslavsky, and S. Loke. Dynamically generated user-specified mas. In J. P. Müller and F. Zambonelli, editors, *Agent-Oriented Software Engineering VI*,

volume 3950 of *Lecture Notes in Computer Science*, pages 139–153. Springer, Berlin / Heidelberg, August 2006. 6th International Workshop, AOSE 2005, Utrecht, The Netherlands, July 25, 2005. Revised and Invited Papers.

[24] T. Juan, A. Pearce, and L. Sterling. ROADMAP: extending the gaia methodology for complex open systems. In C. Castelfranchi and W. L. Johnson, editors, *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 3–10, New York, NY, USA, Jul 2002. ACM.

[25] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley, Boston, MA, USA, 1998.

[26] A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu. Towards requirements-driven autonomic systems design. *SIGSOFT Software Engineering Notes*, 30(4):1–7, Jul 2005. Proceedings ICSE 2005 - Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005).

[27] J. Lind. *Iterative Software Engineering for Multiagent Systems: The Massive Method*. Springer-Verlag New York, Inc., Secaucus NY, USA, first edition edition, 2001.

[28] A. Molesini, E. Denti, and A. Omicini. Agent-based conference management: A case study in SODA. *International Journal of Agent-Oriented Software Engineering*, 2009.

[29] A. Molesini, E. Denti, and A. Omicini. RBAC-MAS & SODA: Experimenting RBAC in AOSE. In A. Artikis, G. Picard, and L. Vercouter, editors, *Engineering Societies in the Agents World IX*, volume 5485 of *LNCS*. Springer, 2009. 9th International Workshop (ESAW'08), 24–26 Sept. 2008, Saint-Étienne, France. Revised Selected Papers.

[30] Object Management Group. Software & Systems Process Engineering Meta-Model Specification 2.0. `http://www.omg.org/spec/SPEM/2.0/PDF`, Apr. 2008.

[31] L. Padgham and M. Winikof. Prometheus: A methodology for developing intelligent agents. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Agent-Oriented Software Engineering III*, volume 2585 of *LNCS*, pages 174–185. Springer, 2003. 3rd International Workshop (AOSE 2002), Bologna, Italy, 15 July 2002. Revised Papers and Invited Contributions.

[32] J. Pavòn and J. J. Gòmez-Sanz. Agent oriented software engineering with INGENIAS. In M. Pěchouček, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications III*, volume 2691 of *Lecture Notes in Computer Science*, pages 394–403. Springer Berlin / Heidelberg, January 2003. 3th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03), Prague, Czech Republic, 16–18 June 2003, Proceedings.

[33] V. Torres da Silva, R. Choren, and C. J. P. de Lucena. A UML based approach for modeling and implementing multi-agent systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 914–921, Washington, DC, USA, 2004. IEEE Computer Society.

[34] J. Vázquez-Salceda and F. Dignum. Modeling electronic organizations. In M. Pěchouček, P. Petta, and L. Z. Varga, editors, *Multi-Agent Systems and Applications III*, volume 2691 of *Lecture Notes in Computer Science*, pages 584–593. Springer Berlin / Heidelberg, January 2003. 3th International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS'03), Prague, Czech Republic, 16–18 June 2003, Proceedings.

[35] G. Wagner. The agent-object-relationship metamodel: towards a unified view of state and behavior. *Information Systems*, 28(5):475–504, July 2003.

[36] M. Wooldridge, N. R. Jennings, and D. Kinny. A methodology for agent-oriented analysis and design. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 69–76, New York, NY, USA, 1999. ACM.

[37] M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[38] F. Zambonelli, M.-P. Gleizes, M. Mamei, and R. Tolksdorf. Spray computers: Explorations in self-organization. *Pervasive and Mobile Computing*, 1(1):1–20, Mar. 2005.

[39] F. Zambonelli, N. R. Jennings, and M. J. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, July 2003.

[40] F. Zambonelli and A. Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283, Nov. 2004. Special Issue: Challenges for Agent-Based Computing.

Table 1 caption: The Size of the ISAS/SCI Conference.
Table 2 caption: The Environment Model for the Review Sub-organization.
Table 3 caption: The ReviewCatcher functional role schema.
Table 4 caption: The ReviewPartitioner organizational role schema.
Table 5 caption: The ReceivePaperAssignement interaction protocol.

Table 1. The Size of the ISAS/SCI Conference.

Conference Year	Number of Presented Papers
1995	55
1997	248
1999	754
2001	1859

Table 2. The Environment Model for the Review Sub-organization.

Action	Environment Abstraction	Description
Reads	Paper Submitted	the Web site receives a paper
	Review Submitted	the Web site receives a review
Changes	DB Submission	insert in the data base the paper or the review received; one per each track
	DB Reviewer	insert in the data base the personal data of the reviewer, the topic of expertise and the maximum number of papers the referee accepted to review

Table 3. The `ReviewCatcher` functional role schema.

Role Name: `ReviewCatcher`		
Description:		
This role is in charge of selecting reviewers and distributing papers among them.		
Protocol and Activities:		
GetPaper, 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` = (GetPaper.CheckPaperTopic.CheckRefereeExpertise. CheckRefereeConstraints.AssignPaperReferee.[ReceiveRefereeRefuse] | UpdateDBSubmission.UpdateDBReferee)^n	
Safety:	$\forall$ *paper: number_of_referees* $\geq$ n Referee $\neq$ Author Referee_organization $\neq$ Author_organization	

Table 4. The `ReviewPartitioner` organizational role schema.

Role Name: `ReviewPartitioner`		
Description:		
This role is in charge of distributing papers among Vice-Chairs according the area of competence.		
Protocol and Activities:		
GetPaper, 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` = (GetPaper.CheckPaperTopic.CheckViceChairArea. AssignPaperViceChair.UpdateDBSubmission)^w	
Safety:	$\forall$ *paper assigned to a ViceChairArea*	

Table 5. The *ReceivePaperAssignement* interaction protocol.

Protocol Name: ReceivePaperAssignement		
Initiator: `ReviewCatcher`	**Partner**: `ReviewPartitioner`	**Input**: *paper_submitted*
Description: The `ReviewPartitioner`, having checked the area of the paper, assigns the paper to the corresponding `ReviewCatcher` (the Vice-Chair in charge of that area).		**Output**: *The paper is assigned to a specific area and the DB Submission is updated*

Figure 1 caption: The Gaia process
Figure 2 caption: The activity diagram of the Analysis phase
Figure 3 caption: The activity diagram of the Architectural Design phase
Figure 4 caption: The activity diagram of the Detailed Design phase
Figure 5 caption: The paper review organization structure for a simple hierarchy
Figure 6 caption: The paper review organization structure for a hybrid structure
Figure 7 caption: The paper review organization structure for a multilevel hierarchy
Figure 8 caption: The Agent model for a one level hierarchy organization
Figure 9 caption: The Agent model for a multilevel hierarchy organization
Figure 10 caption: Categorize Roles activity
Figure 11 caption: Categorize Protocols activity

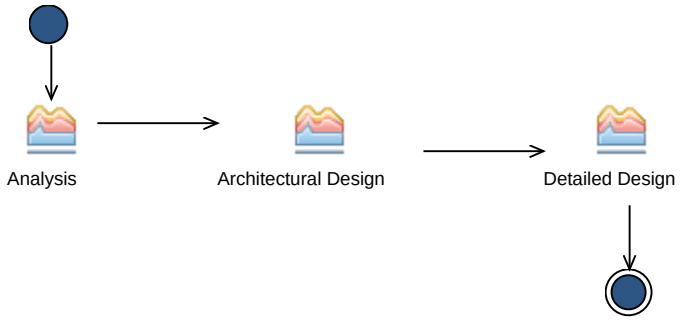

Fig. 1. The Gaia process

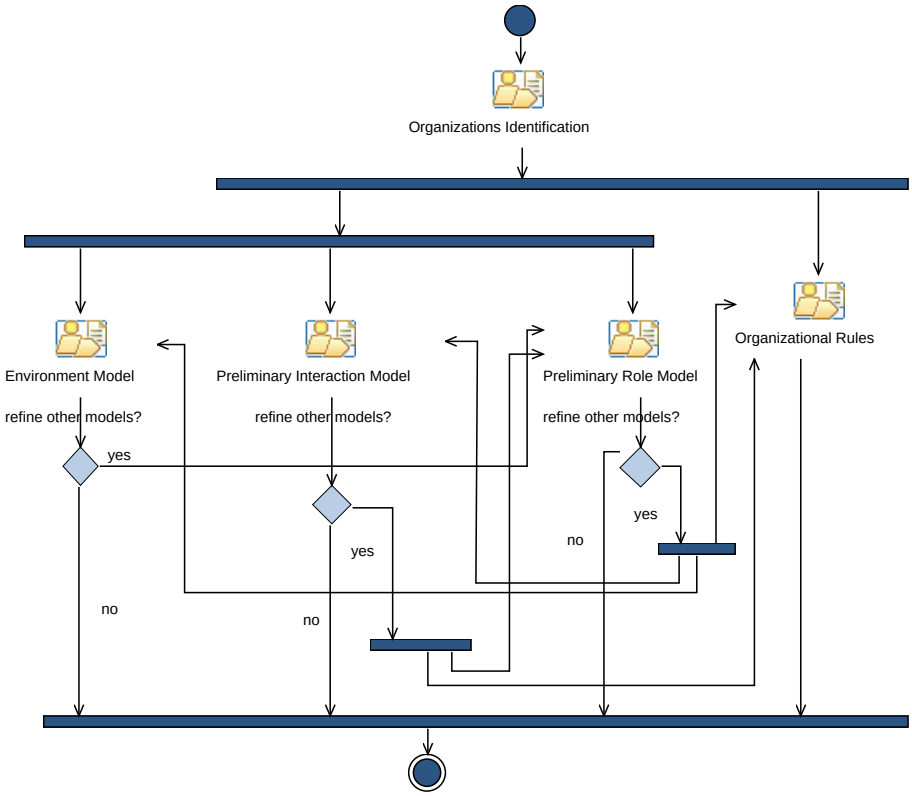

Fig. 2. The activity diagram of the Analysis phase

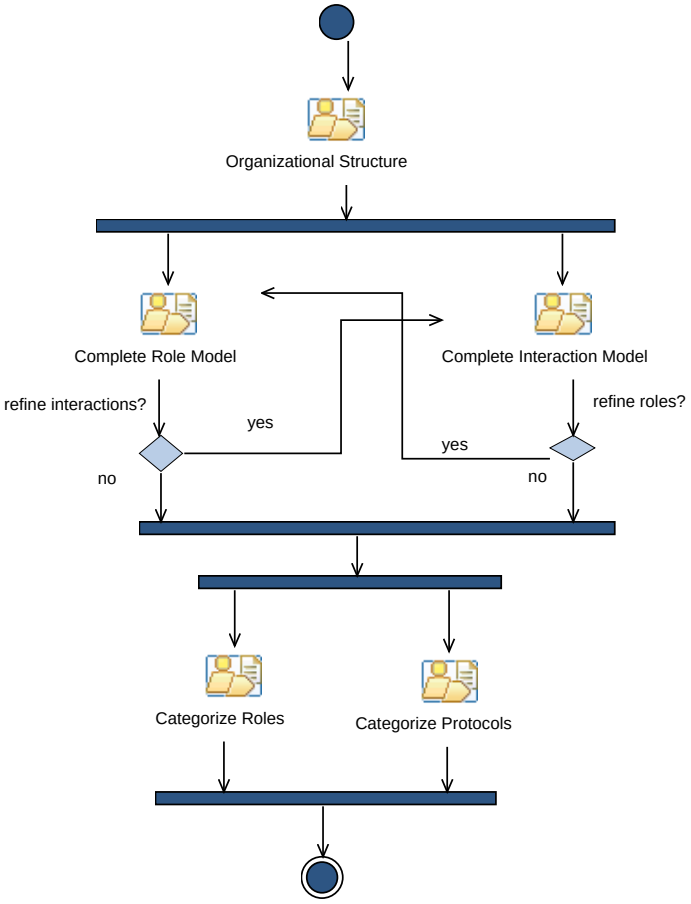

Fig. 3. The activity diagram of the Architectural Design phase

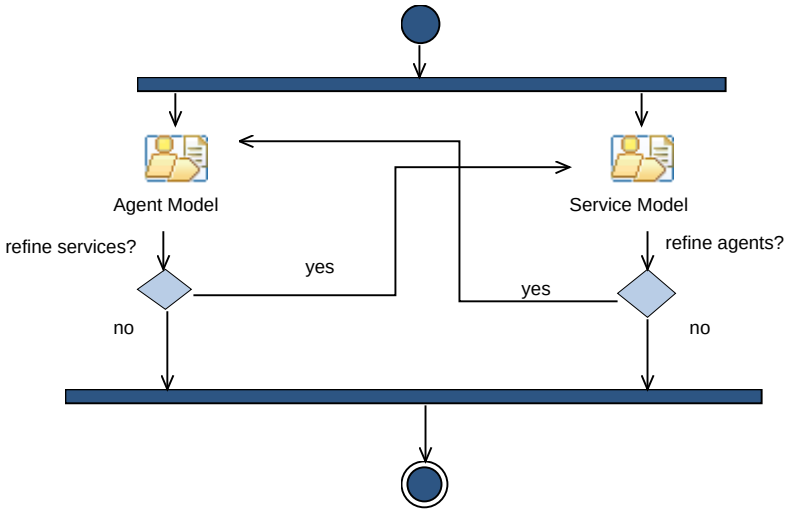

Fig. 4. The activity diagram of the Detailed Design phase

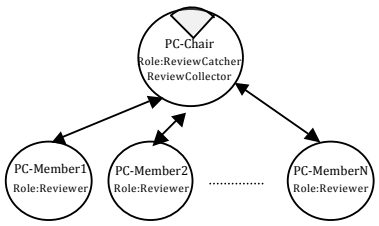

Fig. 5. The paper review organization structure for a simple hierarchy

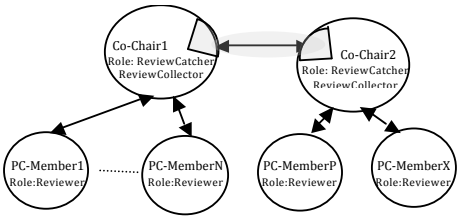

Fig. 6. The paper review organization structure for a hybrid structure

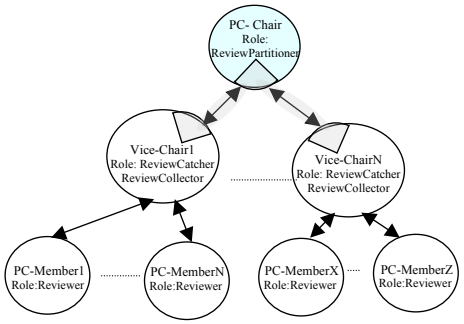

Fig. 7. The paper review organization structure for a multilevel hierarchy

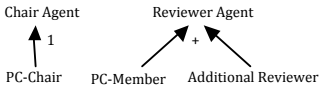

Fig. 8. The Agent model for a one level hierarchy organization

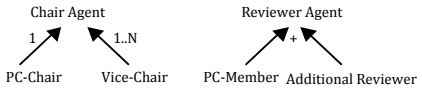

Fig. 9. The Agent model for a multilevel hierarchy organization

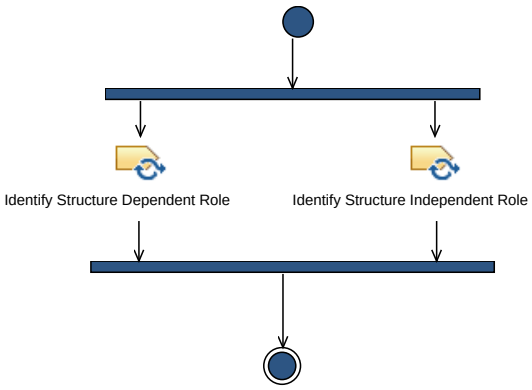

Fig. 10. Categorize Roles activity

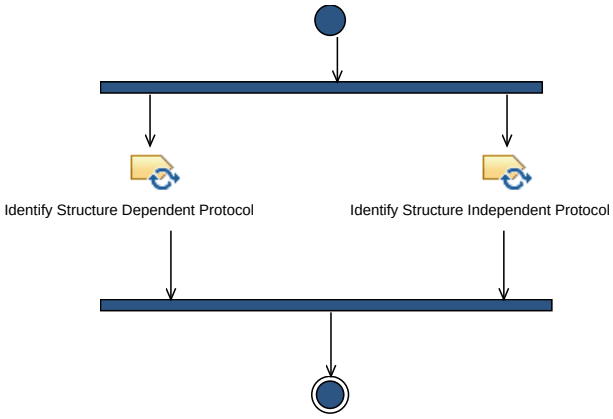

Fig. 11. Categorize Protocols activity