A Requirements Engineering Approach for the Development of Multi-Agent Systems

David Blanes, Emilio Insfran, and Silvia Abrahão
ISSI Research Group, Department of Information Systems and Computation
Universidad Politécnica de Valencia, Camino de Vera, s/n 46022, Valencia, Spain
dablado@posgrado.upv.es, {einsfran, sabrahao}@dsic.upv.es

Abstract

This paper presents RE4Gaia, which is a requirements modeling approach for the development of multi-agent systems that extends the Gaia methodology. The approach focuses on dealing with the organizational structure as a means to adequately capture and understand required roles and associated functions in the context of an organization prior to the analysis and design of MAS using Gaia. In addition, a traceability framework is introduced to facilitate moving from the requirements models to the analysis and design models proposed in Gaia.

Keywords: Requirements engineering, multi-agent systems, methodology, agent-oriented development.

1. Introduction

The Requirements Engineering (RE) process is recognized as being the most critical process of software development. Errors made during this process can have negative effects on subsequent development steps and on the quality of the resulting software. A Multi-Agent System (MAS) is a specific type of system that is composed of multiple, interacting intelligent agents that are used to solve problems that are difficult (or impossible) for an individual agent or monolithic system to solve. In the last few years, many agent-oriented methodologies have been proposed to support the development systems of this type following different approaches: goal-oriented, object-oriented, etc. Unfortunately, most of these proposals focus only on the analysis, design, and/or implementation phase of the MAS.

Perhaps the most developed, agent-oriented software engineering methodology is Gaia [1]. Gaia is based on the organizational metaphor and is founded on the view of multi-agent systems as a computational organization that consists of various interacting roles. According to Gaia, a MAS is viewed as being composed of a number of autonomous interactive agents that live in an organized society in which each agent plays one or more specific roles [1]. When applying Gaia, the analyst moves from abstract to increasingly concrete roles and interactions. Zambonelli et al. [1] suggest that analysis and design in Gaia is a process of developing increasingly detailed models of the system to be constructed. However, in Gaia, requirements are just statements, which are independent of the paradigm used for analysis and design rather than a model that is oriented to capture requirements relevant to a MAS. Another drawback for Gaia, from our point of view, is the lack of explicit traceability from requirements to the artifacts produced throughout the MAS development. A better traceability mechanism could help to meet user needs, improve their understanding of the systems,
facilitate the maintainability of produced artifacts, and improve the overall quality of the developed software [2].

In this work, we introduce RE4Gaia, which is a requirements modeling approach for the development of multi-agent systems that extends the Gaia methodology. This approach focuses on dealing with the organizational structure as a means to adequately capture and understand required roles and associated functions in the context of an organization prior to the analysis and design of MAS using Gaia.

This paper is organized as follows. Section 2 presents the related work and includes characteristics of some methodologies for the development of MAS. Section 3 briefly introduces the Gaia methodology. Section 4 presents our requirements modeling proposal. Section 5 describes a case of study used to validate our approach. Finally, section 6 presents the conclusions and further work.

2. Related work

In a previous work [2], we presented the results of a Systematic Review of the Use of the Requirement Engineering techniques in the development of MAS. Our findings showed that the majority of MAS methodologies focus only on the analysis and design phases and do not give support to the requirements phase. This is the case of the Gaia methodology and the Multi-Agent System Iterative View Engineering (MASSIVE) methodology [3]. Other MAS development approaches give partial support to the requirements phase through use-cases modeling or scenarios, i.e. ROADMAP [4], MaSE [5], or INGENIAS [6]. Perhaps the most developed and “well-accepted” approach in the community for dealing with requirements, analysis and design for MAS development is Tropos [7]. Tropos is based on the i* framework, which follows a goal-oriented approach to model early and late requirements as well as architectural and detailed design. This fact reveals that there is a dearth of alternative methods and techniques for appropriately dealing with requirements for MAS development [2]. Most of the alternative requirements methodologies proposed in the literature are focused on understanding the problem domain and/or communicating requirements among stakeholders. These methodologies lack traceability mechanisms to trace from requirements information to analysis and design artifacts and vice-versa. We believe that this fact is an important issue that constraints a wider use of these alternative proposals.

In summary, there have been many attempts to provide techniques and methods to deal with some aspects of the requirements engineering process. However, there is still a need for solutions that allow developers to go systematically from well-defined requirements models to design models in a guided or automated way.

3. The Gaia methodology

The aim of the Gaia methodology is to guide the design of open systems using organizational concepts. It models two levels: the micro-level (agent structure) and the macro-level (agent society). Gaia views MAS as a set of autonomous interactive agents that can play one or more roles. Gaia only deals with the analysis and design of MAS. In this work we limit our description to the analysis phase, which is directly related to our proposal. Figure 1 shows the models and their relationships in the analysis phase.
Figure 1. Gaia Analysis models and relations

The analysis phase starts with the definitions of the global organization goal. It includes the decomposition of the global organization into sub-organizations. Each sub-organization has a sub-goal to achieve. The next step is to build the environmental model, the preliminary role model, the preliminary interaction model and the organizational rules.

The environmental model lists all resources that one agent can access. Gaia suggests representing the resources as variables or tuples, where one agent can do three types of actions: sensing, effecting or consuming. This notation is inspired in FUSION [8]. A symbolic name and the type of action that the agent can perform in it and an optional description are defined for each resource.

The preliminary role model is built to capture the basic skills. In this model, a role is represented using an abstract and semiformal description. Two types of attributes are necessary to describe a role: permissions and responsibilities. Permissions are used to identify the resources that are accessed by the role and establish the limits for it. Responsibilities are used to indicate the expected behavior of a role and are divided in two types: liveness properties and safety properties. The liveness properties establish the order of the task of the role. The safety properties define the restrictions in the behavior of the role.

The preliminary interaction model determines dependencies and interactions, between roles through protocols. A protocol is defined with a set of attributes: name (brief textual description of the role); initiator role or roles (roles responsible for starting the interaction); partner role or roles (the responder roles); inputs (information used by the initiator roles); outputs (information supplied by the protocol responders); and a textual description to explain the purpose of the protocol.

Finally, the organizational rules represent the responsibilities of the organization. There are two types of organizational rules: liveness rules and safety rules. The liveness rules define the organizational dynamics (e.g. to indicate that certain roles can only be played if previous roles have been performed). The safety rules define time-independent global invariants that the agent must respect (e.g. to indicate the constraint: two entities cannot play the same role).

4. The requirements modeling approach

Our requirements modeling approach is designed to capture and to understand the required roles and associated functions in the context of an organization prior to the analysis and design of MAS using Gaia. It includes a set of techniques to gather and represent the software requirements for MAS. Specifically, our approach provides: i) A
Requirements Modeling Phase and ii) A Requirement Analysis Phase. Figure 2 shows an overview of the proposed models and their relationships.

4.1. Requirements modeling phase

The goal of the Requirements Model (RM) Phase is to gather and represent the software requirements. This phase starts by defining the Mission Statement, the Functional Refinement Tree, the Requirements Role Model, and the Domain Model. The Mission Statement establishes the goals of the global organization. The Functional Refinement Tree helps to determine the sub-organizations that makeup the global organization and it participant roles. The Requirements Role Model is used to detect inheritance relations among the roles and to reason about their structural relationships, detecting possible inconsistencies. Finally, the Domain Model is used detect the entities that are potential organizational resources.

Figure 2. Overview of RE4Gaia models and their relations

4.1.1. Mission statement: The Mission Statement is the most general service that the system to be developed provides to its environment [9]. It is written in natural language with typically one or two paragraphs. The purpose is to determine in a clear and concise way, what the system must do.

4.1.2. Functional refinement tree: One of the main advantages of Gaia is that it uses of the organizational metaphor. Gaia suggests dividing the system into sub-organizations to take advantage of the modularity and encapsulation principles. In order to benefit from this organizational metaphor, we use a Functional Refinement Tree (FRT) to group different areas of the organization that have low coupling.

The FRT is used to represent a hierarchical decomposition of the business functions, independently of the future software system structure. We place the Mission Statement in the root of this tree. This Mission Statement is then successively refined until the functions of the system are obtained. These functions are represented as leaf nodes in the FRT. There are several levels in this process. The nodes between the root and the leaf nodes are intermediate nodes.

We distinguish two levels of intermediate nodes in a FRT. At the first level of intermediate nodes, are the Functional Groups which are called Sub-organizations. A
Sub-organization is part of the system that is oriented to achieving a goal in the system and that interacts weakly with other parts of the systems (low coupling).

At the second level of intermediate nodes, the sub-organizations are decomposed into roles. A role is a representation of an abstract entity that provides (several) functions for the system. A function is a task performed by a role in the organization, independently of whether or not it needs to collaborate with other roles.

Figure 3. Example of a Refinement Tree

Figure 3 shows an excerpt of a FRT for a Fish Market system, which is introduced in Section 5. In this example, the system is divided into two sub-organizations: Admission and Auction. The Admission sub-organization has the following roles: Guest, Buyer, and Admitter. The Guest does the Register function, the Buyer does the Update Credit function, and the Admitter does the Process Register and Process Update Credit functions. The Auction sub-organization has the following roles: Auctioneer and Buyer. The Auctioneer role can do the Start Auction function, and the Buyer role can do the Assign Winner and Bid functions.

4.1.3. Requirements role model: The Requirements Role Model describes the roles that belong to sub-organizations of the FRT. We consider a role to be an abstract entity that provides (several) functions for the system. The purpose of this model is to represent the different roles discovered in the organization and to reason about their structural relationships. In particular, we need to identify common properties among the roles in order to create a hierarchy of roles using an inheritance relationship. In order to graphically represent this information, we use the UML Use Case diagram, but label the actor with the stereotype «role», and we also show their relationships with each other. An example of this diagram is shown in Figure 4 (b).

4.1.4. Domain model: The Domain Model shows domain entities that are identified in the problem domain. The purpose is to gather key concepts and their relationships in order to depict an initial first structural view. First, relevant domain entities for the application domain are identified. An entity can be a physical entity or a concept. These entities must make sense in the application domain. Implementation details should be avoided at this level. In addition to domain entities, we represent associations among them. We annotate these
associations with the multiplicity: zero, one, many or a constant. Finally, we refine the model using the inheritance relationship.

In order to graphically represent all this information, we use the UML Class diagram but we only show the entities as classes and their relationships (associations and inheritance) with each other. An example of this diagram is shown in Figure 4 (a).

4.2. Requirements analysis phase

The Requirements Analysis Phase takes the identified elemental functions in the FRT as input and decomposes them into tasks and protocols. For this purpose, we use the UML Activity Diagram. The Activity Diagram is used to understand the internal flow of a role in order to determine its responsibilities in the Gaia Role Model. Once the activities and tasks are identified, we identify the resources (Environmental Model) by refining the entities discovered in the Domain Model then the permissions of these resources are defined. Once we have the internal task and protocols of a role and its accessed resources, we define the Organizational Rules in order to define the behavior and restrictions in the dynamics of the organization as a whole.

4.2.1. Activity diagram: The Activity Diagram specifies dynamic constructions using activities and actions. This diagram has a sequence of steps that show the workflow that is necessary to perform the identified functions. The Activity Diagram is considered to be an extension of traditional flowcharts. Flowcharts model flow of control step by step and they are typically used to model algorithms. However, the Activity Diagram can also model sequential and concurrent flow of control. This characteristic makes the Activity Diagram a good option for modeling Human Organizations that perform operations concurrently.

We use the Activity Diagram to help the analysts identify role dependencies (see Fig. 5). A representation of the task flow can be useful in understanding the logical flow of a role; it can help to identify when a role needs to collaborate with other roles.

There are several elements in the Activity Diagram. The Initial Node shows the starting point of an Activity Diagram. It is represented as a solid circle with an outgoing arrow. The initial node cannot have incoming edges. The Activity Final Node indicates that the flow in an activity stops. It is represented with a filled circle with a border. An Action represents an activity that can occur in the Activity Diagram. It is represented by rounded rectangles. The flow between actions is modeled with a Control Flow. A Control Flow is an edge that starts an activity node after the previous one is finished. It is represented by an arrow in the diagram.

Task concurrency is modeled using Fork and Join elements. The Fork is represented by a black bar with one edge going into it and several leaving it. It denotes the beginning of parallel activity. The Join denotes the end of a parallel process. It is represented with a black bar with several edges entering it and one leaving it. The Decision Nodes are represented by a diamond with one flow entering and several leaving. The selected leaving flow depends on the evaluation of the guards at the outgoing edges.
An Activity Partition is a kind of activity group for identifying actions that have a certain characteristic in common. As Fig. 5 shows, a partition with two (usually vertical) parallel lines and a name that labels the partition. Any activity nodes as edges that are placed between these lines are considered to be contained within the partition. In order to represent the protocols, we use the Send Signal Action to model a protocol that is starting. A send signal is represented by a convex pentagon. We suggest adding a text inside the pentagon with the role that starts the protocol and the responder role in parenthesis. A protocol response is modeled with the Accept Event Action. An Accept Event Action waits for a call for a Send Signal Action. This is represented by a concave pentagon.

For reasons of brevity, we show only some of the guidelines regarding for using and Activity Diagram: i) At least one Activity Diagram must be built for each sub-organization identified in the FRT. Depending on the complexity of the problem, more than one diagram per sub-organization can be build; ii) One Activity Diagram should be built for each role that is proactive (the role is the initiator of other protocols); iii) Each Activity Diagram should have at least one Activity Partition. The left partition can be used for the role that starts the protocol. The other partitions can represent the other roles that interact with the initiator role; iv) If the role needs to interact with other roles, then new activity partitions can be added in the diagram. If the new role has the ability to start protocols (initiator), a new Activity Diagram should be created for this role.

4.2.2. Environmental model: The Environmental Model is a set of tuples with the following structure: role, resource, and permission. For each role from the Actor Model, we set the appropriate resources accessed by the role. The information extracted from the Activity Diagram will help the analyst to identify which resources are accessed. Then we set the permission type for access to the resource. There are three types of permissions: read, write, and consuming, which are similar to Gaia.

4.2.3. Organizational rules: The last step in the Requirement Analysis phase is to define the Organizational Rules. These rules are written in natural language in order to gather and represent general constraints for the Organizational behavior. These rules can be viewed as responsibilities of the organization as a whole and can be related to i) Organizational dynamic properties that define how the organization should evolve; ii) Organizational constraints that define the restrictions that the organization must respect at all times.

5. Case study
We introduce a case of study for an Auction System for a Fish Market. This case study was proposed in [10], [11], and we adapted it in order to illustrate our proposal. The complete specification of the case study can be found at: http://www.dsic.upv.es/~einsfran/RE4Gaia/casestudy.html.

The system starts when the Boss opens the market. He/she acts as supervisor of the auction process and is the last authority in the Auction House.
The first phase in the auction is **Admission**. If a Buyer wants to participate in the fish market, he/she must register in the Admission Office. If he/she is a new user, then a new credit account must be opened to make the payments. If the new Buyer is accepted, then he/she can go to the Auction Room. If an existing Buyer wants to participate in the Fish Market, then he/she must validate their credit.

The Sellers bring their goods to the market by unload the fish boxes from the ships at the harbor. The Sellers put their fish boxes on a conveyor belt that transport them towards the Seller Admitter. The Seller sets the following parameters for each box: weight, fish quality, and price. The list with this information is sent to the Auctioneer. Then the Seller waits until his/her Fish have been sold. The Seller receives the money from the Seller Manager only when the fish have actually been sold.

The **bidding round** takes place in the Auction Room. It is the most important phase of the auction system. The buyers bid for the Fish offered by the auctioneer. He calls out the prices in desendent order following the downward bidding protocol. A bidding round can have several situations: i) **Proper sale**: a buyer submits a bid. If he/she has enough credit to can back up the bid, then the sale is completed; ii) **Unsupported bid**: a buyer submits a bid. If he/she does not have enough credit to back up the bid, then the Buyer Manager sanction the bidder. The round is restarted by the auctioneer with a new price. This price is increased by 25%; iii) **Collision**: If two or more buyers submit the same bid, then the auctioneer declares a collision. The round is restarted. The starting price is increased by 25%.

A round can start or restart if the boss of the Fish Market gives permission to the Auctioneer to do so. The Boss can give permission to start or restart a round if: i) there is a minimum of three buyers; ii) there are enough goods to be auctioned.

When the Auctioneer starts a new bidding round, he/she sends each Buyer a list of the Buyers taking part in the round, a list of the goods in the auction room, and the next goods (Fish) to be auctioned. After that, the Auctioneer starts broadcasting prices to the Buyers in the Auction Room. When one of these goods is sold, the owner is informed of the new sale. The Seller’s earnings are updated.

**Settlements** in the Fish Market are carried out in a straightforward way. Basically, buyers must go to the Settlements Office before leaving the Fish Market in order to pick up a statement that describes the agreement purchase and to pay for it. The Sellers can pick up their earnings at the Settlements Office once their lot of goods has been sold. Buyers can also go to the settlements office to update their credit or to carry out other operations.

### 5.1. Requirements modeling phase

This phase involves the identification and modeling of the system requirements: define the **Mission Statement**, build the **RFT**, build the **Requirements Role Model**, and build the **Domain Model**.

#### 5.1.1. Mission statement:

The mission of the Fish Market system is to automate the management of admission, register the incoming bids, give support to the Auction process and manage the sale of goods. The main activity, the auction, involves other
subtasks like managing conflicts: control that there are enough buyers, manage collisions, and manage attempts to buy without enough credit.

5.1.2. Functional refinement tree: The FRT decomposes the system into functions taking as the root the mission statement. The intermediate nodes are obtained from the mission statement. These represent sub-organizations (level 1) and roles (level 2). The last level contains Functions (level 3). A function can be achieved by a role, in the context of a sub-organization independently of whether or not needs to collaborate with other roles.

Table 1 represents an excerpt of the FRT showing the branch of the sub-organization Settlement in a tabular format. The Buyer role performs the functions Purchase goods and Take goods away. The Buyer Manager role performs the functions: Update credit, Send Buyer List, Send results of credit verification and Sanction Buyer. Finally, the Seller Manager role performs the Update credit function.

<table>
<thead>
<tr>
<th>Sub-organization (Intermediate level 1)</th>
<th>Role (Intermediate level 2)</th>
<th>Function (Leaves nodes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settlement</td>
<td>Buyer</td>
<td>Purchase goods</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take goods away</td>
</tr>
<tr>
<td></td>
<td>Buyer Manager</td>
<td>Update credit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send Buyer List</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Send results of credit verification</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sanction buyer</td>
</tr>
<tr>
<td></td>
<td>Seller Manager</td>
<td>Update the credit</td>
</tr>
</tbody>
</table>

5.1.3. Requirements Role Model: The Requirements Role Model is used to represent the system roles identified in the FRT and to specify the inheritance relationships among them. In our case study, we identified the following roles: Guest, Buyer, Seller, Boss, Buyer Manager, Auctioneer, Seller Admitter, Buyer Admitter, and Seller Manager. There is an inheritance from Buyer and Seller roles to the Guest, which represents the fact that everything that can be done by a Guest can also be done by a Buyer or by a Seller. This information is shown in Figure 4 (b).

5.1.4. Domain model: The main domain entities and relationships found in the case study are represented in Figure 4 (a). We identify a Person entity that shares common properties with the Seller and Buyer. A Buyer List is related to multiple Buyers, and an Item List is related to multiple Items. An Item belongs to one Seller, and a Bidding Round is composed by multiple Bids. Each Bid has Buyer List, multiple Prices, and one Item List associated to it.
5.2. Requirements analysis phase

This phase involves the analysis of the previous functions identified in the FRT, using the information from the Role Model and the Domain Model. An Activity Diagram is proposed to refine these functions into: tasks, which are functions performed by one unique role; or protocols, which are functions performed by two or more cooperating roles. With the roles identified in the Role Model, the entities from the Domain Model and the information extracted from the Activity Diagram, the next step is to identify the Resources and their Permissions. Finally, a set of Organizations Rules are defined to regulate the organization behavior.

5.2.1. Activity diagram: We use Activity Diagrams to analyze in detail each sub-organization identified in the Function Refinement Tree. The goal of this diagram is to identify role dependencies and to refine the identified functions into tasks and protocols.

For this case study, we use two Activity Diagrams to analyze the Settlement sub-organization. Each diagram represents a role with a proactive behavior (role that is initiator of other protocols): the Buyer and the Seller. Figure 5 shows the Activity Diagram for the Buyer role. In the first partition, the Buyer can start the Purchase protocol to interact with the Buyer Manager. We add also a note to indicate a restriction: “The buyer must be a winner”. When the Buyer finishes the protocol Purchase, then he/she exits from the scenario. In the second partition the Buyer Manager waits for Purchase protocol requests. When the Buyer receives a Purchase protocol, then he/she checks if the Buyer role has enough credit. If the Buyer has enough credit, then the Update Credit task is performed, otherwise the Buyer starts the Bad Credit protocol to communicate a “bad purchase” to the Auctioneer role. In the third partition the Auctioneer is waiting for Bad Credit requests.
5.2.2. Environmental model: For each role identified in the FRT and also specified in the Role Model (to consider inheritance relations) we establish the permissions to access the resources. Table 2 shows a partial view of this model, which represents three roles: Auctioneer, Buyer, and Buyer Manager. The Auctioneer can modify a Bid, a Bidding Round, and the Price of a Bid. The Buyer can read the Bid, the Buyer List, and Item resources. Finally, the Buyer Manager can read an Account of a user and modify the Buyer List.

<table>
<thead>
<tr>
<th>Role</th>
<th>Resource</th>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auctioneer</td>
<td>Bid</td>
<td>Modify</td>
</tr>
<tr>
<td></td>
<td>Bidding Round</td>
<td>Modify</td>
</tr>
<tr>
<td></td>
<td>Price</td>
<td>Modify</td>
</tr>
<tr>
<td>Buyer</td>
<td>Bid</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Buyer List</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td>Read</td>
</tr>
<tr>
<td>Buyer Manager</td>
<td>Account</td>
<td>Read</td>
</tr>
<tr>
<td></td>
<td>Buyer List</td>
<td>Modify</td>
</tr>
</tbody>
</table>

5.2.3. Organizational Rules: The purpose of the Organization Rules specification is to define what is allowed for the role’s behavior and what is not. We use the natural language to specify these rules. We use the following organizational rules for the case study:

- A Buyer cannot participate directly in the Auction sub-organization. He/she must be registered in the Admission sub-organization before joining the auction.
- A Buyer cannot participate in a sale if he/she is not a bid winner.
- The Auctioneer must wait to start the auction if there are not enough goods to sell.
- The Auctioneer must wait to start the auction if there are not enough Buyers to participate.
5.3. Traceability framework

Due to space limitations, we briefly outline the traceability strategy used to relate the models built using the RE4Gaia approach to the Gaia analysis models. This traceability framework can be viewed from two perspectives: inner and outer. The inner traceability refers to the links established between elements in RE4Gaia. The outer traceability refers to the links established between elements in RE4Gaia and the Gaia analysis models.

The main relationships for inner traceability are the following:

- The mission statement is related to sub-organizations identified in the FRT.
- Each role identified in the FRT (and specified in the Role Model) is related to its corresponding sub-organization in the FRT.
- Each role identified in the FRT (and specified in the Role Model) is related to a role into the Environmental Model.
- Each entity identified in the Domain Model is related to a resource in the Environmental Model. For each resource the user decides which roles have permissions over it.
- An Activity Diagram is related to the corresponding sub-organization identified in the FRT. It is recommend having one activity diagram for each role that can be proactive in the sub-organization.
- Each function in the FRT is related to a task or protocol in the Activity Diagram, depending whether or not the functions need to collaboration with others roles.

The main relationships of outer traceability are the following:

- Each resource from the Environmental Model is related to a resource in the Gaia Analysis model. However, at this level, the Environmental model is preliminary and new resources can be identified in the Gaia analysis. The permissions for each resource are mapped from the Requirements to the Gaia analysis.
- Each role identified in the Requirements Role Model is related to a role in the Gaia Role Model. The information extracted from the Activity Diagram helps the user to find the responsibilities: the liveness properties and safety properties.
- Each task or protocol identified in the Activity Diagram is related to a task or protocol in the Gaia Role Model, giving relevant information during the analysis and design of the MAS.
- The organizational rules in natural language are mapped to organization rules in the Gaia methodology. The user can decided whether convert the rule as liveness or a safety rule. The rule information is completed in the analysis phase, using the FUSION based notation.

6. Conclusions

We have presented a Requirements Modeling approach for the development of multi-agent systems that extends the Gaia methodology. This approach deals with the organizational structure as a means to adequately capture and understand required roles and associated functions in the context of an organization prior to the analysis and design using Gaia.

We believe that our approach fills the gap in the development of MAS by providing a systematic approach to deal with requirements. This approach establishes better traceability mechanisms that help analysts to meet user needs, improve their understanding of the
systems, facilitate the maintainability of produced artifacts, and improve the overall quality of the developed software.

We are currently applying this requirements modeling approach for the specification of other agent-oriented systems with the intention to study the expressiveness and transformation capacities of our requirements models. We plan to perform controlled experiments with PhD students in order to empirically validate the effectiveness of our method. We are also working in the development of a tool that is based on the Eclipse framework to implement the proposed approach. This tool follows a model-driven development approach that includes model2model transformations from RE4Gaia models to Gaia models.

Acknowledgments
This research is funded by the MULTIPLE project (with ref. TIN2009-13838).

References

Authors

David Blanes is currently a PhD. student in Computer Science at Universidad Politecnica de Valencia. He obtained a Master Degree in Software Engineering, Formal Methods and Information Systems in the Department of Information Systems and Computation at Universidad Politecnica de Valencia, Spain. His research interests are: Agent-Oriented Requirements Engineering, Model-Driven Development, and Software Quality.

Emilio Insfran is a Tenured Assistant Professor Faculty of Computer Science at Universidad Politecnica de Valencia (UPV). He obtained the PhD in Computer Science in the Department of Information Systems and Computation at Universidad Politecnica de Valencia, Spain. His research interests are: Requirements Engineering, Model-Driven Development, and Software & Process Quality.

Silvia Abrahão is a Tenured Assistant Professor Faculty of Computer Science at Universidad Politecnica de Valencia (UPV). She obtained the PhD in Computer Science in the Department of Information Systems and Computation at Universidad Politecnica de Valencia, Spain, Summa cum laude (European PhD mention). Her research interests are: Empirical Software Engineering, Software Quality, Functional Size Measurement, Usability Engineering, Conceptual Model Quality, and Web Engineering.