RE4Gaia: A Requirements Modeling Approach for the Development of Multi-Agent Systems using Gaia[1]

David Blanes, Emilio Insfran, 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 extending the Gaia methodology. The approach focus on dealing with the organizational structure as a means to adequately capturing and understanding required roles and associated functions, in the context of a organization, prior to the analysis and design of the MAS using Gaia. In addition, a traceability framework is introduced in order to facilitate moving from the requirements models to the analysis and design models proposed in Gaia. The proposed requirements modeling approach is validated using a case of study.

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 used to solve problems that are difficult for an individual agent or monolithic system to solve. In the last few years, many agent-oriented methodologies [1] [2] [3] [4] [5] have been proposed to support the development of this type of systems following different approaches: goal-oriented, object-oriented, etc.

[1] This work is cofunded by the Spanish Department of Science & Technology, under the National Program for Research, Development and Innovation, META project (TIN2006-15175-C05-01), the European Regional Development Fund. and by the Quality-driven model transformations project (UPV).

The original publication is available at www.springerlink.com

Perhaps the most developed agent-oriented software engineering methodology is Gaia [2]. Gaia is based on the organizational metaphor and founded on the view of multi-agent systems as a computational organization composed of a number of autonomous interactive agents that live in an organized society in which each agent plays one or more specific roles [2]. In Gaia, requirements are just *statements*, independent of the paradigm used for analysis and design, rather than a model 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 along the MAS development. A better traceability mechanism could help to meet user needs, improve facilitate the maintainability of produced artifacts, and improve the overall quality of the developed software [3].

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

This paper is organized as follows. Section 2 presents the related work including 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

We presented in a previous work [3], the results of a Systematic Review of the Use of the Requirement Engineering techniques in the development of MAS. We identified that the majority of MAS methodologies focus only on the analysis and design and do not give support to the requirements phase. This is the case of the Gaia methodology and the MASIVE [4]. Others MAS development approaches give a partial support to the requirements phase through use cases or scenarios. This is the case of PASSI [5] or ROADMAP [6], which uses use cases changing its traditional semantics to capture interactions with a team of “ideal agents”. MaSE [7] gives a partial requirements elicitation support, offering to apply goals capture together with use cases. Perhaps the most developed and “well-accepted” approach in the community for dealing with requirements, analysis and design for MAS development is Tropos [8]. Tropos is based on the i* framework and is used to model early and late requirements, architectural and detailed design following a goal-oriented approach. This fact reveals that there is a dearth of alternative methods and techniques for appropriately dealing with requirements for MAS development [3]. Moreover, most of the alternative requirements methodologies proposed in the literature are focused in understanding the problem domain and/or communicate requirements among stakeholders. They lack traceability mechanisms to trace this requirements information towards analysis and design artifacts and backward. We believe that this fact is an important issue that constraints a wider use of these alternative proposals to Tropos.

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

3 The Gaia Methodology

Gaia is a methodology with the purpose of guide the design of open systems using organizational concepts. Gaia is tailored the analysis and design of MAS. We describe only the analysis phase, which is directly related to our proposal. This phase starts with the definitions of the global organization goal. It includes the *decomposition of the global organization into sub-organizations.*

The next step is to build the *environmental model*. This model list 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. For each resource is defined a symbolic name and the type of action that the agent can perform in it.

The *preliminary role model* is build to capture the basic skills. In this model, a role is represented is represented with an abstract and semiformal description. There are two types of attributes to describe a role: permissions and responsibilities. The permissions are used to identify the resources accessed by the role and establish the limits for the role. The responsibilities are used to indicate the expected behavior of a role. They are divided in two types: liveness properties and safety properties.

The *preliminary interaction model* determines dependencies and interaction between roles through protocols. The protocol is defined with a set of attributes: name, initiator role or roles, partner role or roles, inputs, outputs and a description to explain the purpose of the protocol.

Finally, the *organizational rules* represent the responsibilities of the organization. They are two types of organizational rules: liveness rules and safety rules. The *liveness* rules define the organization dynamic. The *safety* rules define time-independent global invariants that the agent must respect.

4 The Requirements Modeling Approach

The proposed requirements modeling approach is aimed to deal with the organizational structure as a means to adequately capturing and understanding required roles and associated functions, in the context of a organization, prior to the analysis and design 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*. ii) *A Requirement Analysis Process*. Figure 2 shows an overview of the proposed models and their relationships.

4.1 Requirements Modeling

The goal of the Requirements Model (RM) is to gather and represent the software requirements. This phase starts defining the Mission Statement, the Functional Refinement Tree (FRT), the Requirements Role Model (RRM), and the Domain Model (DM). The Missions Statement set the goals of the global organization. The FFT helps to determine the sub-organizations forming the global organization and it participant roles. The RRM is used to detect intheritance relations between role and reason about their structural relationships, detecting possible inconsistencies. Finally the DM is used detect the entities that could be possible organizational resources.

The **Mission Statement** is the most general service (the main goal) that the system to be developed provides to its environment [13]. It is written in natural language with typically one or two paragraphs.

The **FRT** is used to represent a hierarchical decomposition of the business functions independently of the future software system structure. We put in the *root* of this tree the Mission Statement and then is successively refined looking for the functions of the system that are represented as *leaf nodes* in the FRT. In this process, we can find several levels. The nodes between the root and the leaf nodes are *intermediate nodes*. We distinguish two levels of intermediate nodes. At the first level of intermediate nodes, we find Functional Groups which are called **sub-organizations**. A sub-organization is part of the system which is oriented to achieve a goal in the system and that weakly interacts with other parts of the systems (low coupling). At the second level of intermediate nodes, we find that 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 if it needs to collaborate with other roles or not.

The **RRM** describes the roles belonging to sub-organizations from the FRT. 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 showing only the roles as actors, labeled with the stereotype «*role*», and the relationships among them. An example is shown in Figure 4 (b).

The **DM** shows entities identified in the problem domain. The purpose is to gather key concepts and their relationships depicting a first structural view. First, relevant domain entities for the application domain are identified. These entities must make sense in the application domain, implementations details have to be avoided at this level. In addition to domain entities we represent associations among domain entities. We can annotate these associations with the multiplicity: *zero*, *one*, *many* or a constant. Finally, we can refine the model with the inheritance relationship. In order to graphically represent all these information, we use the UML Class diagram showing only the entities as classes and the relationships (associations and inheritance). An example of this diagram is shown in Figure 4 (a).

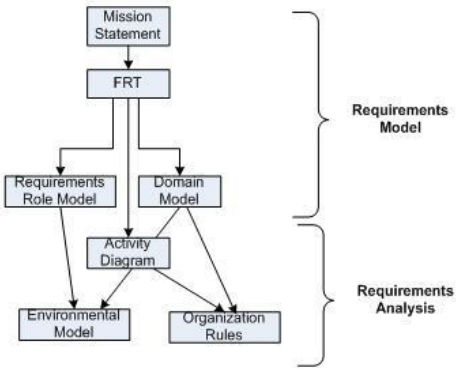


Fig. 1. Overview of RE4Gaia models and their relations

4.2 Requirements Analysis

The Requirements Analysis Process takes as input the identified elemental functions in the FRT and decomposes them into: tasks and protocols. We use for this purpose the UML Activity Diagram. Moreover the Activity Diagram is used to understand the internal flow of one role in order to determine its responsibilities into the Gaia role model. Once the activities and tasks are identified, we identify the resources (Environmental Model) refining the entities discovered into the Domain Model into resources and its permissions. Once we have the internal task and protocols of a role and his accessed resources, we define the organizational rules to define the behavior and restrictions in the dynamics of the organization as at all.

The **Activity Diagram** specifies dynamic constructions using activities and actions. This diagram shows a sequence of steps showing a workflow necessary to realize the identified functions. We use the Activity Diagram to help the analysts identifying role dependencies. A representation of the task flow can be useful to understand the logical flow of one role; helping to identify when a role needs to collaborate with others roles. Some guidelines regarding Activity Diagram are: i) It's necessary to build one Activity Diagram (at least) for each sub-organization identified in the FRT. Depending on the complexity of the problem; we can be build more than one diagram per sub-organization; ii) We suggest building one diagram for each role that is proactive, in the sense that the role is the initiator of other protocols; iii) We create at least one Activity Partition. Each partition models the role logical flow. We suggest that the left partition is used for the role that starts the protocol. The others partitions represent the other roles which interacts with the initiator role; iv) If we identify that the role needs to interact with other roles, then we add a new activity partitions in the diagram. If the new role has the ability to start protocols (initiator), we suggest create a new Activity Diagram to this role.

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 rightful resources accessed by the role. The information extracted from the Activity Diagram will help to the analyst to identify which **resources** are accessed. Finally we set the

permission type for access to the resource. We distinguish three types of permissions, in a similar way to Gaia: *read*, *write*, and *consuming*.

The last step in the Requirement Analysis phase is to define the **Organizational Rules**. They are written in natural language with the purpose 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) Organization dynamics properties, defining how should evolve the organization; ii) Organization constraints, defining restrictions that the organization must respect in every time.

5 Case Study

We introduce a case of study for an Auction System for a Fish Market. This case study was proposed in [14] 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 first phase is the **Admission**. If a buyer wants to enter in the fish market, he must register in the Admission Office. If he is one new user, then it is necessary to open a new credit account. The existing users must validate their credits. Meanwhile the buyers performs the admission, the sellers bring the goods. For each good the seller's admitter employed fix for each box: the weight, the fish quality and the price and send the good list the Auctioneer.

The **bidding round** takes place in the auction room. The auctioneer starts a new bidding round. First of all, he/she sends to each buyer the list of buyers taking part in the round, the list of goods in the auction room, and the next good to be auctioned. After that, the auctioneer starts broadcasting prices to all buyers in the auction room. When one good is sold, then the owner is informed of the new sale. The seller's earnings are updated. The buyers bids for the goods offered by the auctioneer. He calls the prices in descendent order following the downward bidding protocol. One bidding round can present several situations: i) *Proper sale:* the buyer has enough credit; ii) *Unsupported bid:* a buyer submits a bid but he hasn't enough credit; iii) *Collision:* if two or more buyers submit the same bid.

Settlements in the fish market are carried out in a straightforward way. Basically buyers must show up at the **settlements office** before leaving the market in order to collect a statement describing their purchases and pay them up, whereas sellers may collect their earnings at the settlements office once their lot of goods has been sold.

5.1 Requirements Modeling

The first step is to define the SM. 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 manage conflicts: control that are enough buyers, manage collisions, manage attempts to buy without enough credit.

Table 1. Excerpt of a FRT for the Settlement sub-organization

Sub-organization (Intermediate level 1)	Role (Intermediate level 2)	Function (Leaves nodes)
Settlement	Buyer	Purchase good
		Take good away
	Buyer Manager	Update credit
		Send Buyer List
		Send result verify credit
		Sanction buyer
	Seller Manager	Update the credit

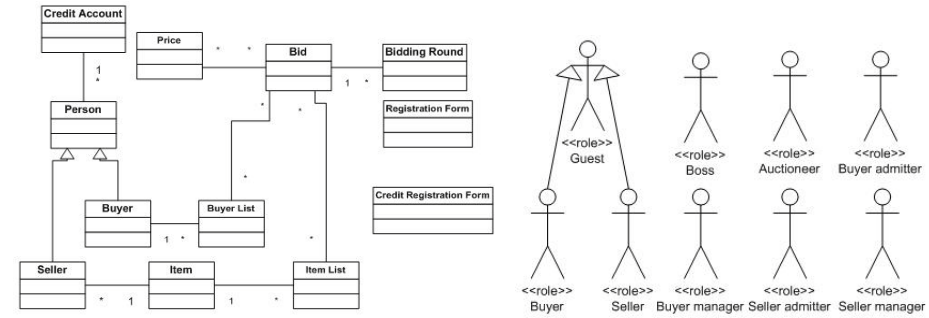
The next step is to build the FRT. Table 1 represents an excerpt of the FRT showing the branch of the sub-organization *Settlement* (in a tabular format). The *Buyer* role does the functions *Purchase good* and *Take good away*. The *Buyer Manager* role does the functions: *Update credit*, *Send Buyer List*, *Send result verify credit* and *Sanction Buyer*. Finally the *Seller Manager* role does the *Update credit* task.

The RRM is used to represent the system and to specify the inheritance relationships between 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, representing 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).

The main domain entities and relationships found 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.

5.2 Requirements Analysis

This phase starts with the construction of the Activity Diagrams. In order to analyze the Settlement sub-organization, we should use two activity diagrams. Each diagram it is to represent each 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. The *Buyer* when joining in the scenario can start the *Purchase* protocol to interact with the *Buyer Manager*. The 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 other partition, Auctioneer, is the *Buyer Manager*, which is waiting for the *Purchase* protocol requests. The role checks if the *Buyer* role has enough credit. If it is true then the *Update Credit* task is performed, otherwise it starts the Bad Credit protocol to communicate a “bad purchase” to the Auctioneer role.



a) b)

Fig. 2. a) Domain Model; **b)** Requirements Role Model.

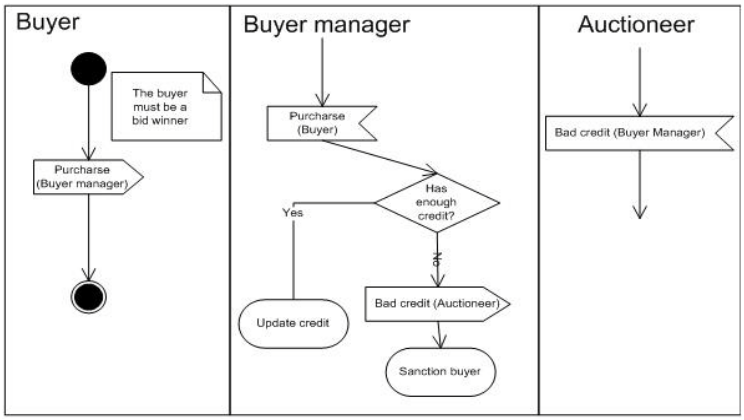


Fig. 3. Activity Diagram for the Buyer role in the Settlement sub-organization

For each role identified in the FRT, and also specified in the Role Model to consider inheritance relationships, we set the resources with permission granted to the role. Table 2 shows a partial view of this model, representing 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 a *Bid*, the *Buyer List* and *Item* info. Finally the *Buyer Manager* can read an *Account* of a user and modify the *Buyer List*.

At last we have the following organizational rules for our case study: i) One buyer can not join directly to the Auction phase. He/she must be registered in the admission phase before joining the auction, ii) One buyer cannot join in the closing if he/she is not a bid winner, ii) The Auctioneer must wait if there not enough goods to auction, iv) The Auctioneer must wait if there are not enough buyers in the auction.

Table 2. Partial Environment Model

Role	Resource	Permission
Auctioneer	Bid	Modify
	Bidding Round	Modify
	Price	Modify
Buyer	Bid	Read
	Buyer List	Read
	Item	Read
Buyer Manager	Account	Read
	Buyer List	Modify

5.3 Traceability Framework

In this section, due to space limitations, we briefly outline the traceability strategy used to relate the models build 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.

Related to the inner traceability, the main relationships are: i) The mission statement is related to sub-organizations identified in the FRT, ii) Each role identified in the FRT (and specified in the Role Model) is related to its corresponding sub-organization in the FRT, iii) Each role identified in the FRT (and specified in the Role Model) is related to one role into the Environmental Model, iv) Each entity identified in the Domain Model is related to one resource into the Environmental Model. The user decides with which roles is related the resource and with which permissions have access granted, v) An Activity Diagram is related to the corresponding sub-organization identified into the FRT. It is recommend having one activity diagram for each role that can be proactive in the sub-organization, vi) Each function in the FRT is related to a task or protocol in the Activity Diagram, depending if the functions needs collaboration with others roles or not.

Related to the outer traceability which the main relationships are: i) Each resource from the Environmental Model is related to one resource in the Gaia Analysis model. However, at this level the Environmental model is preliminary and new resources can be identified at the Gaia analysis. The permissions are mapped for each role can access to the resource, ii) 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 to the user to fill the responsibilities: liveness, and safety properties, iii) Each task or protocol identified into the Activity Diagram is related to one task or protocol in the Gaia Role Model, giving relevant information that is needed to specify in detail during the analysis and design, iv) The organizational rules in natural language are mapped to organization rules in the Gaia methodology. The user can decided if is converted into a liveness o safety rule, depending of his nature.

6 Conclusions and Further Work

We have presented a Requirements Modeling approach for the development of multi-agent systems extending the Gaia methodology. The approach deals with the organizational structure as a means to adequately capturing and understanding 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 providing a systematic approach to deal with requirements establishing 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.

Currently, we are applying the 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 using the Eclipse framework to implement the proposed approach. This tool will allow to follow a model-driven development approach including model2model transformations from RE4Gaia models to Gaia models.

References

1.Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: High Variability Design for Software Agents: Extending Tropos. In : ACM Trans. Autonom. Adapt. Syst. 2, 4, Article 16 (2007)

2.Pavón, J., Gomez, J.: Agent Oriented Software Engineering with INGENIAS. In: CEEMAS, Prague, Czech Republic, pp.394-403 (2003)

3.Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems vol. 8, num. 2, 203-236 (2004)

4.DeLoach, S., Wood, M., Sparkman, C.: Multiagent Systems Engineering. International Journal of Software Engineering and Knowledge Engineering 11(3), 231-258 (2001)

5.Lind, J.: Iterative Software Engineering for Multiagent Systems, the MASSIVE Method. (2001)

6.Zambonelli, F., Jennings, N., Wooldridge, M.: Developing Multiagent Systems:The Gaia Methodology. ACM Transactions on Software Engineering and Methodology (TOSEM), 317-370 (2003)

7.Blanes, D., Insfran, E., Abrahão, S.: Requirements Engineering in the Development of Multi-Agent Systems: A Systematic Review. In : Accepted for publication in the International Conference on Intelligent Data Engineering and Automated Learning, Burgos, Spain (2009)

8.Cossentino, M.: From requirements to code with the PASSI methodology. In : Agent-Oriented Methodologies. Idea Group Inc (2005) 79–106

9.Juan, T., Pearce, A., Sterling, L.: ROADMAP: extending the gaia methodology for complex open systems. In : AAMAS, Bologna, Italy, pp.3-10 (2002)

10.Insfran, E.: A Requirements Engineering Approach for Object-Oriented Conceptual Modeling., Valencia, Spain (2003)

11.Napoli, C., Giordano, M., Furnari, M.: A PVM implementation of the Fishmarket. In : IX International Symposium on Artificial Intelligence, Cancun (1996)