1

Requirements Modeling for Multi-Agent Systems

Lorena Rodriguez[1], Emilio Insfran[1] and Luca Cernuzzi[2]
[1]*ISSI Research Group, Department of Information Systems and Computation, Universidad Politécnica de Valencia, Camino de Vera s/n 46022, Valencia,*
[2]*DEI, Universidad Católica "Nuestra Señora de la Asunción", Campus Universitario, C.C. 1683, Asunción*
[1]*Spain*
[2]*Paraguay*

1. Introduction

The inadequate management of system requirements is a major cause of problems in software development (Anton, 2006). Requirements Engineering is a branch of Software Engineering and includes a set of activities concerning the identification, specification, validation, and management of system requirements. However, a traditional approach to requirements engineering may not be fully effective for certain complex systems that require high levels or specific kinds of abstractions, and the need to define more specific requirements engineering processes for these types of systems thus arises.

A Multi-Agent System (MAS) is a specific type of system that is composed of multiple intelligent agents that interact with each other to achieve certain objectives. These systems can be used to solve problems that it is difficult or impossible for a monolithic or a single agent system to resolve. In recent years, various methodologies have been proposed to guide the development of multi-agent systems, such as Tropos (Giorgini et al., 2005), Ingenias (Gómez-Sanz & Pavón, 2003), Gaia (Zambonelli et al., 2003), etc. However, despite the importance of the requirements phase in the development of software systems, many of the proposed methodologies for the development of MAS do not adequately cover the requirements engineering phase (Cernuzzi et al., 2005), focusing mainly on the design and implementation phases. Moreover, a recent study on the application of requirements engineering techniques in the development of a multi-agent system (Blanes et al. a, 2009) found that 79% of the current methodologies for MAS development use requirements engineering techniques which have been adapted from other paradigms (object orientation, knowledge engineering, etc.) (Anton, 2006). However, these techniques and notations may not be sufficient to cover the nature of MAS, since these systems, along with their organizational, structural, or functional properties, characteristics that are not normally necessary in conventional software systems such as *pro-activity, adaptability, collaboration, truth, or disposition* (Blanes et al. a, 2009). These characteristics are denominated as *social behavior* (Hector & Lakshmi, 2005). Therefore, there is a need for new methods and techniques that enable the appropriate acquisition and treatment of MAS requirements.

(Blanes et al. b, 2009) and (Rodriguez et al., 2009) present two proposals for the acquisition and modeling of requirements for the Gaia (Zambonelli et al., 2003) methodology which covers the analysis and design phase of MAS. Based on the experience using these approaches, this chapter presents an evolution of these earlier proposals to support the acquisition and modeling of requirements regardless of the analysis and design methodology used, and covers four essential perspectives of a MAS: *organizational*, *structural*, *functional,* and *social behavior*.

It is also worth mentioning that agent technology is useful in many complex domains: e-commerce, health, stock market, manufacturing, games, etc. In particular, we are interested in the game development domain since it comprises a set of characteristics such as collaboration, negotiation, trust, reputation, etc., which specially can be dealed with a MAS. According to Google Trends and the ESA annual report (ESA, 2010), games development is one of the business markets that has undergone most growth in the last few years. In addition, the agent-oriented paradigm is one of the most promising for modeling such business market due to the social behavior characteristics (negotiation, cooperation, etc.) of the agents and the complexity that MASs can support. For this reason, in the next chapter, we illustrate the feasibility of our approach by applying the requirements modeling process to the development of the strategic board *Diplomacy Game* (Fabregues et al., 2010).

The structure of this chapter is as follows. Section 2, discusses the related works. Section 3, describes the organizational, structural, functional and social behavior perspectives that were considered when modeling MAS. Section 4, presents the requirements modeling proposal. Section 5, focuses on the case study of the *Diplomacy Game*. Finally, section 6, concludes and introduces future research work.

2. Related work

The importance of the requirements phase in software development is widely known. However, capturing and modeling the requirements of a system is not a trivial task. In particular, MAS require abstractions, techniques, and notations which have been specifically tailored to this domain. We propose four basic perspectives for the modeling of MAS requirements: *organizational*, *structural*, *functional,* and *social behavior*. This section, presents some proposals for the acquisition and modeling of requirements that cover these four perspectives of a MAS.

The organizational perspective is supported in proposals such as GBRAM (Anton, 2006). GBRAM is a relevant traditional goal-oriented requirements engineering proposal. It provides a procedural guide for the identification and development of goals and introduces techniques that assist in their methodical and systematic analysis. GBRAM has a great deficiency in terms of formality. This includes the lack of models, formal notations and tools that support the modeling that the method uses. Nevertheless, the guidelines and the level of clarity it offers are very good. Moreover, GBRAM also emphasize the verification of the requirements through its refinement stage which specifies certain guidelines to follow, thus making this process more reliable. Therefore it is possible to track the requirements captured, and this is reflected in the traceability offered by the method.

Another proposal for requirements modeling that supports the organizational perspective is the i * framework (Yu, 1997). This framework has been established as the basis for the Tropos methodology (Giorgini et al., 2005). Tropos has been appropriately adapted to the acquisition and modeling of the actors in the system and its environment, i.e., the actors,

goals, tasks, interactions, dependencies, resources needed, etc. However, it does not permit a full representation of constraints nor does it propose a modeling environment. Since we consider goal orientation to be of particular interest in the capturing of requirements for MAS, we believe that it is necessary to analyze other methods which are complementary to this approach.

The structural perspective is supported by proposals such as AUML (Odell et al., 2000). AUML tends to be asserted as a notational standard in various methodologies; one of the most common proposals for the requirements phase is the adoption of Use Case diagrams. This formalism has shown good results for the representation of functional requirements and is also a good tool for communication with stakeholders. Nevertheless, Use Cases have limitations in capturing qualitative aspects of the environment and interactions with it. In addition, a interesting contribution of AUML is the Agents Interaction Protocol (AIP), which constitutes a central aspect for MAS, specified by means of protocol diagrams.

Another proposal that covers the structural perspective is KAOS (Van Lamsweerde et al., 1998), a proposal for modeling requirements through goals. KAOS consists of a specification language, a method of elaboration, and the meta-level knowledge which is used as a guide. A KAOS model contains goals, information system requirements, expectations about the system environment, conflicts between goals, obstacles, entities, agents, etc. One of the strengths of the proposal is that of its use of formality to achieve correction. Moreover, the idea of constraint is useful in identifying some of the external problems of integrity, and this contributes to the robustness of the system. However, the successful implementation of the method depends heavily on the developer's experience in the domain and how well defined the problem to be solved is (Huzam & Maibaum, 2006).

Other proposals do not support the organizational and structural perspective. This is the case of CREWS (Maiden, 1998), which focuses on the perspectives of functional and social behavior. CREWS is based on system object models that are abstractions of the key features of the different qualities of the problem domain. CREWS uses these models to generate normal course scenarios, and it then uses the theoretical and empirical research into cognitive science, human-computer interaction, collaboration systems and software engineering as a basis to generate alternative courses for these scenarios. A potential weakness of the CREWS approach is that the generation of scenarios is domain-oriented, in contrast with the goal-oriented scenario analysis and the task-oriented Use Case modeling. If the scenarios are intended to validate the requirements, these requirements should be oriented towards the generation of scenarios.

In summary, the organizational perspective is covered by proposals such as GBRAM and i *, and the structural perspective is covered in proposals such as KAOS and AUML. Most of the proposals presented in some way cover, either totally or partially, the functional and social behavior perspective, as in the case of CREWS. However, to the best of our knowledge no methods that completely cover all four perspectives needed for the development of a MAS exist.

3. Different perspectives for multi-agent systems

This work aims to provide a solution to the lack of RE modeling approaches that appropriately cover the four perspectives of MASs: *organizational*, *structural*, *functional*, and *social behavior*. In order to contextualize these perspectives, an overview of them is presented

which emphasizes both social behavior and organizational aspects, since these are key aspects for the development of MASs.

3.1 Organizational perspective

In the organizational perspective, the organization is represented as a system that has certain goals. The organization attains these goals through consistent actions, which use system resources and alter the desired system state (Falkenberg, 1998). Some authors such as (Zambonelli et al., 2003) consider that the human organizational metaphor is very adequate for systems which are situated in open and changing environments. They define a MAS as a software system that is conceived as the computational instantiation of a group of interacting and autonomous individuals (agents). Each agent can be seen as playing one or more specific roles: it has a set of responsibilities or goals in the context of the overall system and is responsible for pursuing these autonomously. Furthermore, interactions are clearly identified and localized in the definition of the role itself, and they help to characterize the overall structure of the organization and the agent's position in it. The evolution of the activities in the organization, which is derived from the agents' autonomous execution and from their interactions, determines the achievement of the application goal.

3.2 Structural perspective

The structural perspective shows the system architecture in terms of entities and the static relationship between them. The modeling of these entities and relationships provides an abstract structural perspective of the system. We believe that this perspective is necessary to identify the entities that will be needed to build the future MAS. If the static and structural relationships are to be captured accurately, the development method must include formalisms and techniques to specify relationships of hierarchy (inheritance), semantic dependency (association) and part-of relations (aggregation).

3.3 Functional perspective

The functional perspective shows the semantics associated with the organizational roles' services that are motivated by the occurrence of events. In this context, we understand an organizational role to be the representation of an abstract entity that provides (multiple) system methods or services. An event is something that occurs in the environment and to which the organizational role reacts by running a method or service. This perspective focuses to model the functional requirements to be met by the roles in the future MAS.

3.4 Social behavior perspective

The social behavior perspective shows the possible sequences of events or services to which an agent can respond or that occur in its lifetime, along with interaction aspects such as communication between agents, and this is often represented as state or activity diagrams. As is discussed above, in addition to organizational, structural, and functional properties, a MAS also requires characteristics that are not normally required in conventional software systems, such as *pro-activity, adaptability, collaboration, truth, or disposition*. These characteristics are denominated as *social behavior*. We therefore believe that covering this perspective in a proposal for modeling requirements for MAS is an important contribution towards the development of such systems, since the essence of these systems is the performance of complex tasks that other types of systems are not capable of solving.

3.4.1 Classification

In order to properly structure and organize the features of social behavior requirements, we briefly present the classification scheme of agent characteristics defined in (Hector & Lakshmi, 2005). According to the authors, three main attributes of an agent are defined: (a) *autonomy*, which refers to the fact that an agent should run independently, with little or no human intervention, (b) *temporal continuity*, which signifies that an agent should run continuously rather than simply perform a task and finish, and (c) *social skills*, which signifies that an agent should possess some form of social skills, since the agent's advantages lie in its interactive communication with other agents. In addition to these core attributes, an agent can also be classified according to the following *social behavior* characteristics:

a. *Pro-activeness*: this refers to how the agent reacts to -and reasons about - its environment, and how it pursues its goals. The agent can directly react to stimuli in its environment by mapping an input from its sensors directly to an action, or it can take a purely planning, or goal-oriented, approach to achieve its goals. This last approach relies upon utilizing planning techniques.
b. *Adaptability*: this describes an agent's ability to modify its behavior over time. In fact, the term "agent" is often taken to implicitly mean "intelligent agents", which combine traditional artificial intelligence techniques to assist in the process of autonomously performing tasks. This feature includes other sub-features such as learning and sub-submission.
c. *Mobility*: this refers to the agents' capability of transporting their execution between machines on a network. This form of moving can be physical, where the agent travels between machines on a network, or logical, where an agent which is running on a single machine is remotely accessed from other locations on the Internet.
d. *Collaboration*: collaboration among agents underpins the success of an operation or action in a timely manner. This can be achieved by being able to coordinate with other agents by sending and receiving messages using some form of agent communication language, and permits a high degree of collaboration, thus making social activities such as distributed problem solving and negotiation possible. Moreover, it is possible for agents to collaborate without actual communication taking place. The interaction of agents with resources and their environment may lead to the emergence of collaborative or competitive behavior.
e. *Veracity*: this refers to the agent's ability to deceive other agents via their messages or behavior. An agent can thus be truthful in failing to intentionally deceive other players. Moreover, an agent that is untruthful may try to deceive other agents, either by providing false information or by acting in a misleading way.
f. *Disposition*: this refers to the agent's "attitude" towards other agents, and its willingness to cooperate with them. An agent may always attempt to perform a task when asked to do so (benevolent), or may act in its own interests to collaborate with other agents only when it is convenient to do (self-interested), or it might try to harm other agents or destroy them in some way (malevolent).

The above characteristics in the classification represent to some extent abstraction of human social behavior, and are those that differentiate agent paradigms from traditional software development. In this work, we use this classification to study the characteristics of social behavior and to propose mechanisms for the definition and specification of requirements of

these types. In particular, and as a starting point, in this work we will focus on the following characteristics: proactiveness, collaboration, veracity, and disposition. Other characteristics such as adaptability or mobility will be considered in future work.
Social behavior is a skill that must have an agent in a MAS. Moreover, if we consider the organizational metaphor, an agent can, at different times in its life-cycle, play one or more specific roles, which in turn have a set of responsibilities and goals. We therefore propose to identify these features of social behavior in the requirements modeling process at role level, through an analysis of the goals that need to be attained. Therefore, in the later phases of the software development, when an agent has to be defined, the corresponding roles of which a given agent will be composed will determine the agent's complete social behavior.

4. Modeling requirements for multi-agent systems

To support the *organizational*, *structural*, *functional* and *social behavior* perspectives, we propose a requirements modeling process which is decomposed into two main activities: *Requirements Definition* and *Requirements Specification*. The user's specific needs are identified in the *Requirements Definition* activity. In particular it is identified: the organizational structure of the system; the roles that are required in each sub-organization; the roles goals; the social behavior needed for the roles to carry out their goals; and relevant entities of the environment. The detailed requirements for developers are specified in the *Requirements Specification* activity. The specifications extracted from the *Requirements Definition* activity are refined, and the level of detail increased, in order to identify artifacts which are closer to the analysis and development of the system: activities and interactions, resources of the system, the permissions that roles have in those resources and organizational rules.
Moreover, the process is based on the definition of models needed to describe the problem in more concrete aspects that form the different perspectives of the system. In particular, in the *Requirements Definition* activity it is possible to identify: (a) a *Refinement Tree,* which identify and represent the goals of the system and their hierarchy, the roles that will carry out these goals in an organizational context, and the organizational structure of the system, and (b) a *Domain Model* with which to represent the entities that could be used as the organization's resources. In turn, the *Refinement Tree,* as is specified by the above description, represents: (i) *Mission Statement*, which is the main objective that the system under development provides the environment with in order to identify the overall goal within the organization as a whole; (ii) *Organizational Model*, to represent the sub-organizations of which the global organization is composed; (iii) *Role Model*, to represent the roles involved in each sub-organization, the inheritance relationships between them, and the social behavior needed between roles to accomplish their goals; and (iv) *Goal Model*, to represent the goals associated with each role.
Moreover, in the *Requirements Specification* activity it is possible to identify: (c) a *Behavior Model*, to represent the decomposition of goals into tasks and protocols in order to understand the internal flow of a role to determine its responsibilities, (d) an *Environment Model*, to represent the permissions of the roles identified in the *Role Model* with regard to the resources of the *Domain Model*; and (e) an *Organizational Rules Model*, to represent the constraints of the organization's behavior.

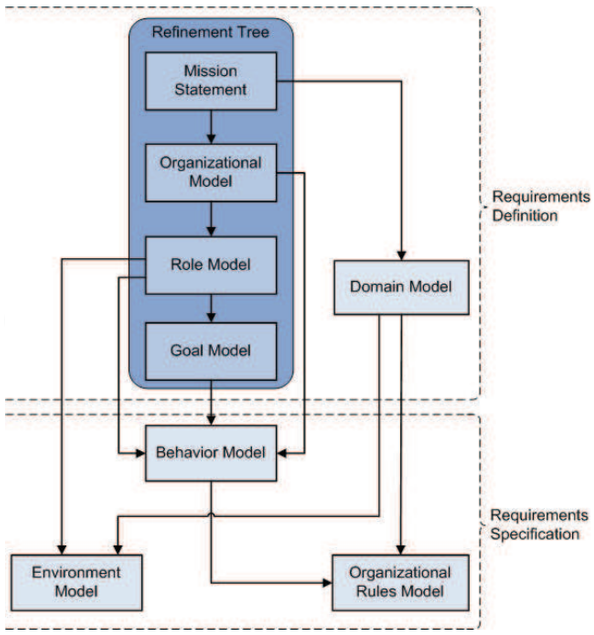


Fig. 1. Models of the proposal and its relationship

In order to obtain a clear view of the models used, each of them is presented as follows. The *Mission Statement* is defined in natural language, with a recommended extension of one or two paragraphs. Since the *Mission Statement* identifies the overall goal within the organization as a whole, it provides us with information about the organizational and functional perspectives. The *Mission Statement* is the root of the *Refinement Tree*. It is successively refined to identify the goals of the system to be represented as leaf nodes in the tree. It is possible to distinguish three general levels in this process: (i) we first define the decomposition of the system in a hierarchy of sub-organizations, thus representing the *Organizational Model*. A sub-organization is a part of the system that aims to achieve a goal in the system and weakly interacts with other parts of the system (low coupling); (ii) we then decompose the sub-organizations into roles that partially represent the *Role Model*. A role is the representation of an abstract entity that has (multiple) system goals; (iii) and finally, we identify the goals of the system and a hierarchy of them, thus representing the *Goal Model*. The goals are tasks which are carried out by a role in the sub-organization. The structure of the *Refinement Tree* allows us to identify elements of the organizational perspective through the decomposition of the system into sub-organizations; elements of the structural perspective by identifying the roles that make up the sub-organizations and finally aspects of the functional perspective by identifying the goals that each role has to perform. As was previously mentioned, the *Role Model* describes the roles that belong to the sub-organizations of the *Refinement Tree*. The purpose of this model is to represent the different roles found in each sub-organization and to reason about their special relationships. The special relationships between roles can serve to identify the common properties between the roles in order to create a hierarchy of roles using inheritance relationships and the identification of the social behavior relationships between roles in different sub-organizations. The resulting *Role Model* comprises the information represented in the *Refinement Tree*: one diagram for the inheritance relations between roles and one or more diagrams as needed for each sub-organization for the social behavior relations

between roles. The UML Use Case Diagram is used to represent this information and complement the *Refinement Tree* representation of roles. The roles are represented as actors which are labeled with the stereotype <<role>>.In addition, the inheritance relations are represented with the corresponding diagram relation, and the social behavior relations are represented as relations labeled with the stereotypes <<collaboration>>, <<disposition>> and <<veracity>>. We propose naming the relations with the corresponding property (i.e. for the social behavior relation *collaboration* the relation is named as "communicative", "non-communicative" or both, for the social behavior relation *disposition* the relation is named as "benevolent", "self-interested", "malevolent" or the combinations, and finally for the social behavior relation *veracity* the relation is named as "truthful", "untruthful" or both). A social behavior relation between two roles could be of one or more property, since the relation is dynamic, i.e. it may alter depending on the agent that will eventually play the role. This information allows us to express elements of the structural, organizational and social behavior perspectives.

The *Domain Model* represents the entities identified in the problem domain. The purpose of this is to identify key concepts and relationships, thus representing a first structural view. These entities are seen from the point of view of the application domain, and implementation details are therefore avoided at this level. Associations and inheritance relationships between domain entities are also represented. The identification of these domain entities and their relationships allows us to extract information for the structural perspective and to partially extract information for the organizational perspective. The UML Class Diagram will be used to represent this information.

The *Behavior Model* shows a sequence of steps that represent the flow of activities needed to achieve the goals identified in the system. A representation of the flow of tasks could be useful: to understand the logical flow of a role; to complement the information regarding social behavior identified in the *Role Model;* and to help to identify new information when one role needs to work with others in order to accomplish a task. The identification of the flow of activities allows us to extract information for the functional perspective. Furthermore, the identification of interactions between different roles allows us to identify information for the social behavior perspective. The UML Activity Diagram will be used to represent this information.

The *Environment Model* represents the permissions of the roles with regard to the entities identified in the *Domain Model*. For each role identified in the *Role Model*, resources are established for those who can legitimately access them. Finally the permissions (perceive or modify) are established. The identification of these permissions offers information of the structural and functional perspectives of the system. The UML Use Case Diagram is used to represent this information, and the roles are represented as actors which are labeled with the stereotype <<role>>, the resources are represented as classes and the permissions are represented as relations between the role and the entity, which are labeled with the stereotypes <<perceive>>, and <<modify>.

The *Organizational Rules Model* identifies and represents the general rules concerning the organization's behavior. These rules can be viewed as general rules, responsibilities, restrictions, the desired behavior, and the sequence or order in such conduct. These rules will be represented by building on GBRAM, in which two types of dependency relationships between goals are distinguished: precedence and restriction, which are represented by the symbols **<** and → respectively, and by adding a relationship to the

proposal to represent general rules of the system, which is represented with only natural language. This information contributes to extract information for organizational, structural and functional perspectives of the system. We suggest that the set of rules should be represented with a table schema in which each rule is defined by a natural language description of the relationship, the type and the corresponding formula if necessary.

Each of these models provides the information which is necessary to cover the different perspectives of a MAS: (i) structural (*Domain Model, Role Model,* and *Environment Model*); (ii) organizational (*Mission Statement*, *Organizational Model*, *Roles Model, Domain Model,* and *Organizational Rules Model*); (iii) functional (*Mission Statement, Goal Model, Behavior Model, Environment Model* and *organizational Rules Model*); and (iv) social behavior (*Role Model* and *Behavior Model*). Figure 1 shows an overview of these models and their respective relations.

The process for defining and specifying the requirements is described in the following subsection.

4.1 Requirements modeling process

As was mentioned earlier, the requirements modeling process proposed involves two phases: *Requirements Definition* and *Requirements Specification*. Figure 2 shows an overview of this process, using the SPEM graphical notation (OMG, 2010). Each activity of the process produces a document that is composed of the sum of all the models and documents of the working definition that is included in each activity. The *Requirements Definition* activity tasks are performed first, thus producing the requirements specification. The *Requirements Specification* activity tasks are then performed, using the requirements specification produced in the previous activity as input and resulting in the production of the refined requirements specification. At this point the *Requirements Definition* activity can again be performed in case some kind of inconsistency or incompleteness is encountered in the specification, or the process may end.

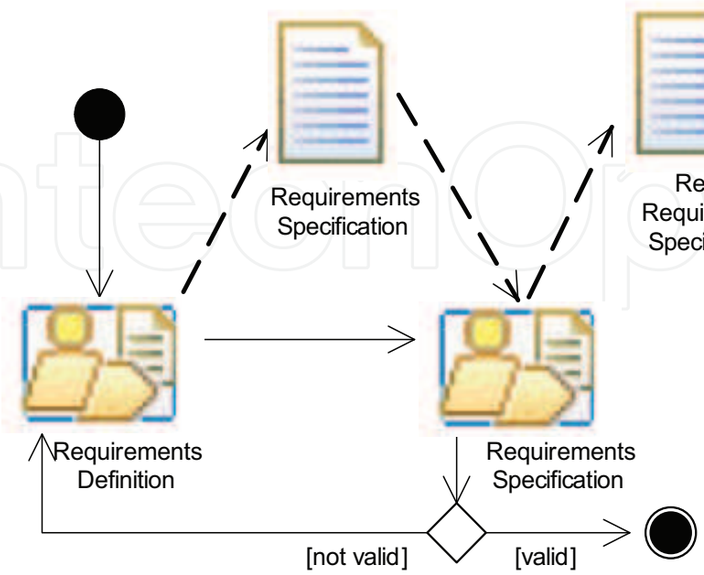


Fig. 2. Requirements modeling process overview

4.1.1 Requirements definition

The *Requirements Definition* activity consists of three tasks whose aim is to identify the models of the phase, as is shown in Figure 3. The first task is to *Create Refinement Tree*, beginning with the definition of the *Mission Statement* which is then broken into sub-organizations, roles and goals. This information is part of the *Mission Statement*, *Organizational Model*, *Role Model* and *Goal Model*. The list of roles identified in the previous task will be used as input for the next task: *Refine Roles*. Here we discuss possible structural similarities in order to identify inheritance relationships, and we analyze the goals to be attained by each role in each sub-organization in order to identify the social behavior relationships between them. If deemed appropriate, it is possible to return to the previous task in order to update the *Refinement Tree*, or the next task can be performed. In the last task, *Identified Entities*, the *Domain Model* is constructed from the identified entities, and association and inheritance relationships among them are defined.

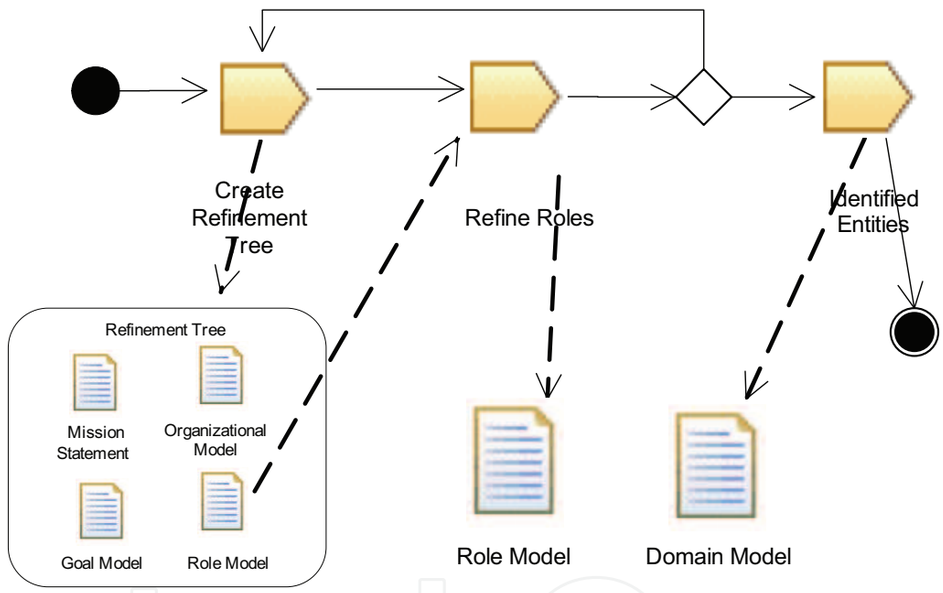


Fig. 3. Requirements Definition activity decomposed into tasks and artifacts

4.1.2 Requirements specification

The *Requirements Specification* activity involves the creation of three models: *Behavior Model*, *Environment Model* and *Organizational Rules Model*, and therefore consists of three tasks for the creation of the models, as is shown in Figure 4. The first task in this activity is *Create Activity Diagrams*. The *Organizational Model*, *Role Model*, *Goal Model* of the *Requirements Definition* activity are used as input. The necessary Activity diagrams are created as a result of this. When this has been completed, the next task is performed: *Develop Environment Model*. The *Role Model* and the *Domain Model* of the *Requirements Definition* phase are taken as input. Then, the *Define Organizational Rules* task is performed, taking as input the *Role Model* of the *Requirements Definition* activity and the *Environment Model* of the current activity. The *Organizational Rules Model* is produced as a result of this.

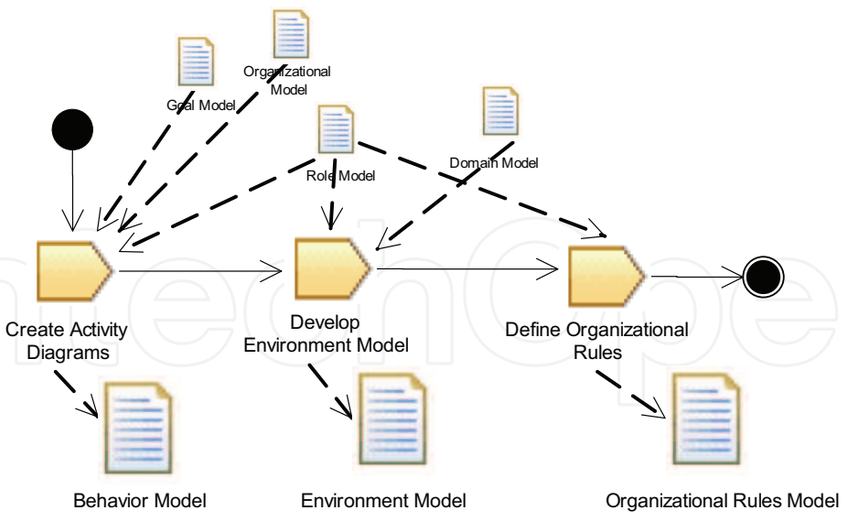


Fig. 4. Requirements Specification activity decomposed into tasks and artifacts

Finally, the artifacts generated during the process can relate to analysis and design artifacts from other methodologies by establishing a traceability framework. This will increase the overall quality of the system to be.

5. Case study: diplomacy game with agents

We have used the *Diplomacy Game* to verify the feasibility of our approach in areas such as negotiation, argumentation, trust and reputation (Fabregues et al., 2010) in the game development domain. Many interesting features make the *Diplomacy Game* compelling for the applying the agent technology: the absence of random movements, all players move their units simultaneously, all units are equally strong so when one attacks another the winner of the battle is decided by considering solely the number of units helping one another, etc. Accordingly, from a player's point of view, the most important feature of the game is the negotiation process: deciding allies, selecting who to ask for help, arguing with other players to obtain information about their objectives or to discover what they know, and so on. We have used the rulebook of the *Diplomacy Game* (Wizards, 2010) as a description of the system to be modeled with the process proposed in this work. The most relevant aspects of the game are provided as follows.

The *Diplomacy Game* is played by seven players and a Game Master. Each player represents one of the seven "Great Powers of Europe" in the years prior to World War I. These Great Powers consist of England, Germany, Russia, Turkey, Italy, France, and Austria. At the start of the game, the players randomly decide which Great Power each will represent. This is the only element of chance in the game. As soon as one Great Power controls 18 supply centers, it is considered to have gained control of Europe. The player representing that Great Power is the winner.

Diplomacy is a game of negotiations, alliances, promises kept, and promises broken. In order to survive, a player needs help from others. In order to win the game, a player must eventually stand alone. Knowing who to trust, when to trust them, what to promise, and when to promise it is the heart of the game.

At the beginning of each turn, the players meet together in small groups to discuss their plans and suggest strategies. Alliances between players are made openly or secretly, and orders are

coordinated. Immediately following this period of "diplomacy," each player secretly writes an order for each of his or her units on a slip of paper. When all the players have written their orders, the orders are simultaneously revealed, and are then all resolved. Some units are moved, some have to retreat, and some are removed. Resolving orders is the most challenging part of the rules and requires complete knowledge of the rules. Each turn represents six months of time. The first turn is called a Spring turn and the next a Fall turn. After each Fall turn, each Great Power must reconcile the number of units it controls with the number of supply centers it controls. At this time some units are removed and new ones are built. The purpose of the Game Master is to keep time for the negotiation sessions, collect and read orders, resolve issues, and make rulings when necessary. This role should be strictly neutral.
Each turn has a series of phases: (a) Spring four-phase turn: Diplomatic phase, Order Writing phase, Order Resolution phase, Retreat and Disbanding phase; (b) Fall five-phase turn: Diplomatic phase, Order Writing phase, Order Resolution phase, Retreat and Disbanding phase, Gaining and Losing Units phase. After a Fall turn, if one Great Power controls 18 or more supply centers, the game ends and that player is declared the winner.
Based on the tasks of the *Requirements Definition* and *Requirements Specification* activities proposed, and which were presented in the previous section, the development of the case study is presented below.

5.1 Requirements definition

The requirements modeling process starts with the *Requirements Definition* activity. This activity starts with the first task: *Create Refinement Tree*. First the *Mission Statement* of the system must be defined, which in this case is simple and is the *Management of the Diplomacy Game*. For the definition of the sub-organizations of the system we decided that the problem naturally leads to a conception of the whole system as a number of different MAS sub-organizations, one for each phase of the game, and one extra sub-organization representing the start of the game. The resulting sub-organizations are: *Initial phase*, *Diplomatic phase*, *Writing Order Phase*, *Order Resolution phase*, *Retreat and Disband phase* and *Gaining and Losing Units phase*. This concept of representing the sub-organizations of the system as phases was also used in (Zambonelli et al., 2003). The roles that are part of each sub-organization are then defined, resulting in three roles: *Great Power*, *Game Master* and *Unit* which, depending on which sub-organization they are, have different goals. Finally the roles are refined with the goals they need to attain in order to fulfill each sub-organization's objective. For example in the *Order Resolution Phase* sub-organization, the *Game Master* role has the goal of *Resolve Order Conflicts* and the *Unit* role has the goal of *Follow Orders*. Figure 5 shows the complete resulting R*efinement Tree*.
The second task, *Refine Role Model,* is performed to complete the *Role Model* based on the information defined in the *Refinement Tree*. Possible inheritance relationships between roles can be specified in this task. The goals of each role in each sub-organization are also reviewed in order to identify whether the role needs social behavior relationships in any sub-organization. Owing to space limitations we shall only illustrate the *Role Model* showing one of the most significant diagrams (see Figure 6). However, the same analysis should be performed for all of the sub-organizations, thus resulting in one or more Social behavior diagrams for each sub-organization. Upon analyzing the goals of the roles of the *Diplomatic Phase* sub-organization, we identified that the *Great Power* role needs to have the collaborative relation to attain all of its goals in the sub-organization analyzed, and more specifically, the role needs to be communicative with other instances of the *Great Power* role

and with the *Game Master* role. The same applies in the case of the *Game Master* role fulfilling its *Control Negotiation Session* goal: the collaborative relationship will be with the *Great Power* role. The collaborative relationship between *Great Power* and *Game Master* will therefore be on both sides, represented with a non-directional arrow. Moreover, if the *Great Power* role is to fulfill all of its goals in the sub-organization analyzed, it needs to be benevolent, self-interested or malevolent with regard to another instance of the *Great Power* role, depending on the agent's intentions. In this sub-organization, negotiation, persuasion and trust are keys to the *Great Power* role. On the other hand, the *Great Power* role in the sub-organization analyzed is in all cases benevolent with regard to the *Game Master* role, and vice versa. Finally, we believe that it is necessary for the veracity relation between the *Great Power* role and other instance of the same role to be truthful or untruthful, again depending on the intentions of the agent playing the role. We also believe that it is necessary for the veracity relation between the *Great Power* role and the *Game Master* role to be truthful in both directions.

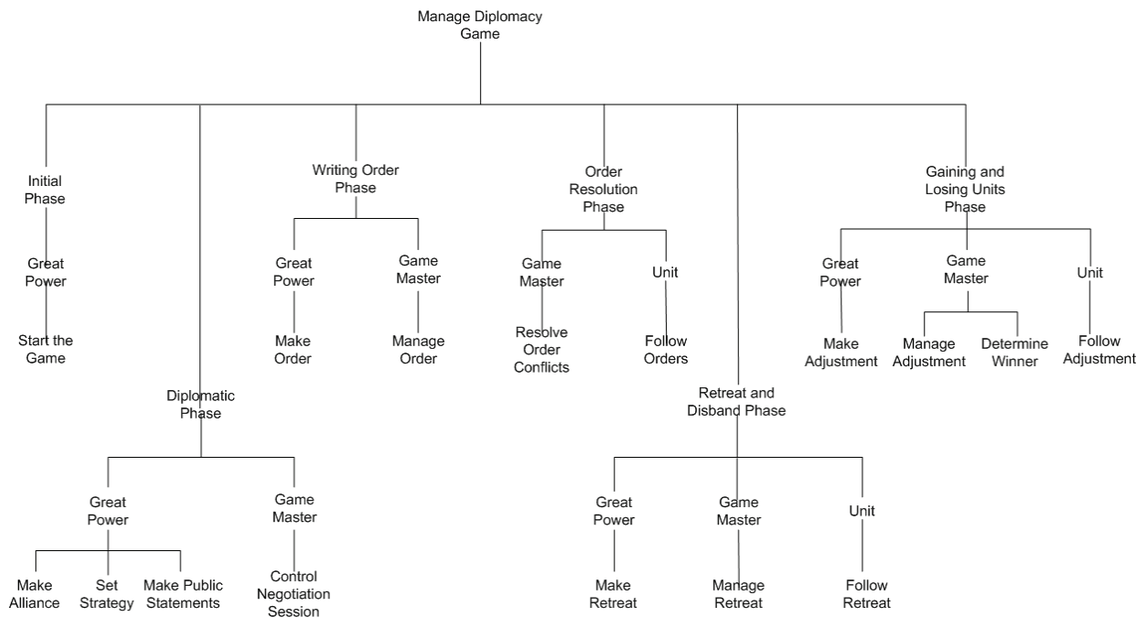


Fig. 5. Diplomacy game Refinement Tree

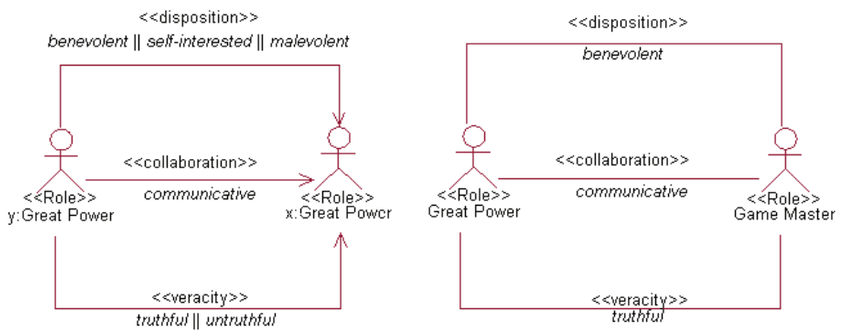


Fig. 6. Social behavior diagram showing relations between roles in the *Diplomatic phase* sub-organization (collaboration, veracity, and disposition)

The third task, *Identify entities,* is performed to define the *Domain Model.* Figure 7 shows the *Domain Model* generated. Briefly, the domain consists of a *Map* that is composed of many *Countries* which in turn have *Boundaries* and *Provinces.* A *Province* can be an *Inland*, *Coastal* or *Water* province. A *Supply Center* is in a *Province*, but a *Province* may or may not have a *Supply Center.* Furthermore, a *Unit* is in a *Province*, but a *Province* may or may not have a *Unit.* A *Unit* can be an *Army* or a *Fleet.* Both a *Province* and a *Unit* belong to a *Great Power* which in turn is a *Country*, but not all *Countries* are *Great Powers.* A *Great Power* has many *Documents*, *Orders*, *Retreats* and *Adjustments*, and they all belong to only one *Great Power*. *Orders*, *Retreats* and *Adjustments* are all for one *Unit* and a *Unit* follows many *Orders*, *Retreats* and *Adjustments.*

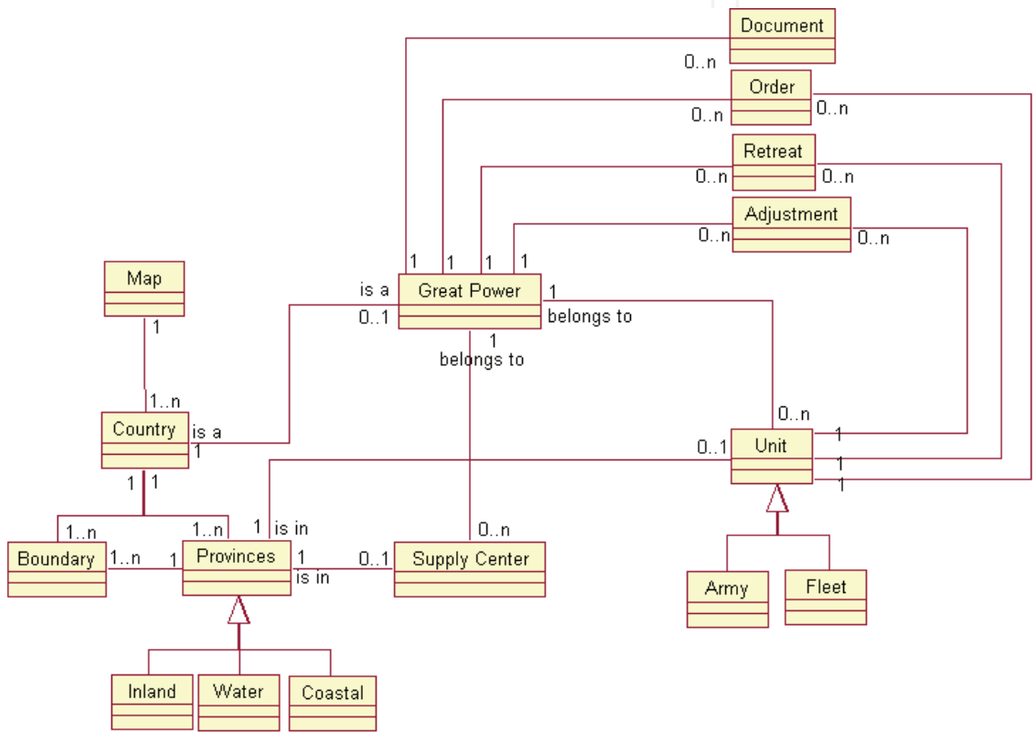


Fig. 7. Diplomacy game Domain Model

As a result of performing the *Requirements Definition* activity we obtain the *Refinements Tree* shown in Figure 5, which represents the *Mission Statement*, *Organizational Model*, partially the *Role Model* and *Goal Model*. One or more Social behavior diagrams is needed for each sub-organization, thus complementing the information of the roles in the *Refinement Tree* to complete the *Role Model*. An example of this is shown in Figure 6. Finally, the UML Class Diagram which relates the entities identified in the domain to represent the *Domain Model* are shown in Figure 7.

5.2 Requirements specification

The second activity is to perform the *Requirements Specification,* which starts with the first task, *Create Activity Diagrams*, in order to specify the *Behavior Model* using the information from the *Organizational Model*, *Role Model,* and *Goal Model* generated in the *Requirements Definition* activity as input. Once again, owing to space limitations we shall only illustrate

one Activity diagram from the *Behavior Model* (see Figure 8). However, the same analysis must be carried out for each goal identified in the *Goal Model,* resulting in one or more Activity diagrams for each goal. The presented activity diagram specifies the activities and protocols performed by the *Great Power* role to attain the *Make Alliance* goal and the activities and protocols performed by the *Game Master* role to attain the *Control Negotiation Sessions* goal, both of which are roles of the *Diplomatic phase* sub-organization. As the goals of these two roles are related, we decide to specify their activity diagrams in just one diagram with tree swim lines, two for the interaction between the two instances of the *Great Power* role (active and passive) to attain the *Make Alliance* goal, and the third for the interaction between the *Game Master* role and the instances of the *Great Power* role to attain the *Control Negotiation Sessions* goal.

As is shown in Figure 8, the flow of actions performed by the *Great Power* active role to attain the *Make Alliance* goal begins with a fork that gives the control to one initiator protocol: *Meet in private groups,* and to one reactive protocol: *Interrupt negotiation session (Game Master)*. The first protocol is initialized by the *Great Power* active role and result in the reactive protocol *Meet in private groups (Active:Great Power)* of the *Great Power* passive role, while the other is a reaction of the *Great Power* role to the *Interrupt negotiation session* protocol initialized by the *Game Master* role if the negotiation time has ended. If this protocol is performed, the *Great Power* active role must terminate the flow of action.

After the *Meet in private groups* protocol has been performed, the *Great Power* active role must perform the *Decide who to trust* activity in order to attain the *Make Alliance* goal. The *Great Power* passive role has the same flow of actions as the *Great Power* active role, with the difference that its *Meet in private groups (Active:Great Power)* protocol is a reaction to the *Meet in private groups* protocol initialized by the *Great Power* active role, and since this is a passive instance of the *Great Power* role, it does not end the flow of actions.

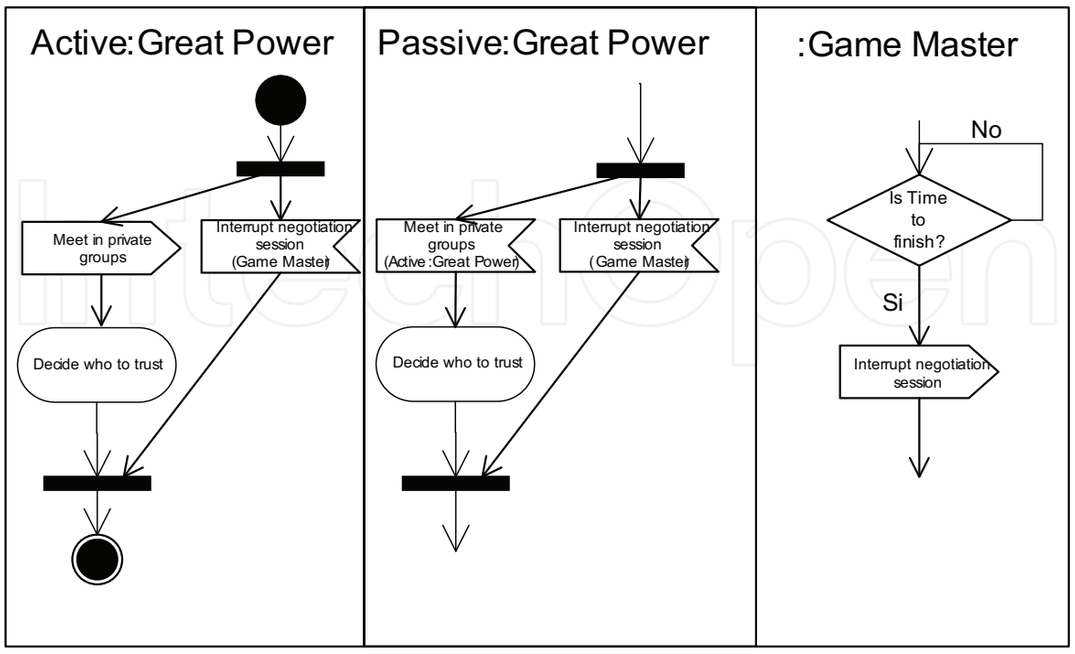


Fig. 8. Activity Diagram for the goals *Make Alliance* and *Control Negotiation Session*

The second task that must be performed in the *Requirements Specification* activity is *Develop Environment Model* using the information from the *Role Model* and *Domain Model* generated in the *Requirements Definition* activity as input. Figure 9 shows the permissions of the *Great Power*, *Game Master* and *Unit* roles with regard to the *Domain Model* resources that each role needs to perceive or modify in order to attain its goals. The *Great Power* role perceives the following entities in the system: other instances of *Great Power*, *Units*, *Map*, *Provinces*, B*oundary* and *Country*; and can modify: *Supply Center*, *Document*, *Order*, *Retreat* and *Adjustment*. The *Game Master* role perceives the following entities in the system: *Great Power*, *Units*, *Map*, *Provinces*, *Supply Center*, *Boundary* and *Country*; and can modify: *Order*, *Retreat* and *Adjustment*; but cannot perceive or modify the *Document* entity. Finally the *Unit* role perceives the following entities in the system: *Great Power*, *Map*, *Provinces*, *Boundary*, *Country*, *Document*, *Order*, *Retreat* and *Adjustment*; but cannot perceive or modify the following entities: other instances of *Unit* and *Supply Center.*

The third task that must be performed in the *Requirements Specification* activity is to *Define Organizational Rules,* using the information from the *Domain Model* generated in the *Requirements Definition* activity and the *Behavior Model* of the current activity as input. In the current domain, the important rules to identify are the general rules of the game, the number of players, the rules concerning the movement of the units depending on the type of unit and on the type of provinces the move take place in, etc. Table 1 shows an extract from the *Organizational Rules Model*.

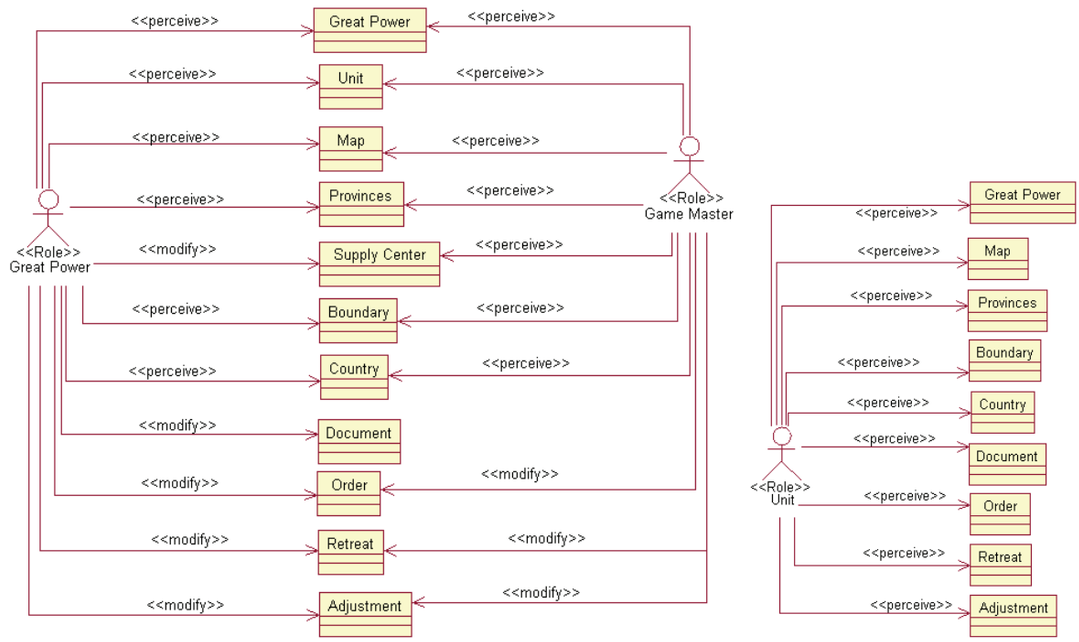


Fig. 9. Diplomacy game Environment Model

Finally, as a result of performing the *Requirements Specification* activity we obtain the *Behavior Model which* is composed with all the Activity diagrams (see example in Figure 8). We also obtain the *Environment Model* (see Figure 9). And finally, a table representing the *Organizational Rules Model* is obtained (see Table 1).

Description
The game is divided into a two year tour: Spring four-phase turn and Fall five-phase turn
Spring four-phase turn has phases: Diplomatic, Order Writing, Order Resolution and Retreat and Disbanding
Fall five-phase turn has phases: Diplomatic, Order Writing, Order Resolution and Retreat and Disbanding, Gaining an Losing
Only seven players may perform the role of "Great Power"
When 18 supply centers belongs to a "Great Power" the game ends and the winner is that "Great Power"
At the start of the game each “Great Power”, except Russia, controls 3 supply centers
At the start of the game, the “Great Power” Russia controls 4 supply centers
Maximum time in the first diplomatic phase is 30 minutes

Table 1. Extract of Organizational Rules Model

5.3 Discussion

With the definition of the *Diplomacy Game Refinement Tree* (see Figure 5), the requirements engineer is able to identify the overall goal of the system, the decomposition of the system in a hierarchy of sub-organizations, roles involved in each sub-organization, and the goals which are carried out by each role in the corresponding sub-organization. The *Diplomacy Game Refinement Tree* provides information for the *organizational*, *functional*, and *structural* perspectives of the case study system. In addition, the social behavior needed for each role to carry out their goals is specified by mean of one Social behavior diagram for each sub-organization (see Figure 6). The case study presents a variety of social characteristics that allow to fully evaluating the proposed Social behavior diagram. In particular, we identified relationships of collaboration, disposition and veracity. The Social behavior diagrams provide information for the *social behavior* perspective. Moreover, the relevant entities of the environment of the game are identified in the *Diplomacy Game Domain Model* (see Figure 7), providing information for the *organizational* and *structural* perspective.

With the construction of one Activity diagram for each goal, the requirements engineer is able to refine each goal in activities and protocols, and also to refine the social behavior identified in the previous activity. It is proper to mention that the collaboration relationships identified in the Social behavior diagrams is refined in the Activity diagrams. As an example, the initiator protocols and reactive protocols in Figure 8 show the specification of the collaboration relation identified in the Social behavior diagram of Figure 6. The Activity diagrams also provide information for the *functional* and *social behavior* perspectives.

Furthermore, the *Diplomacy Game Environment Model* (see Figure 9), identifies the resources of the system, and defines the permissions that roles have in those resources, providing information for the structural and functional perspectives. The organizational rules of the game are specified (see Table 1), providing information for the *organizational*, *structural* and *functional* perspectives.

Finally, due to its characteristics, the *Diplomacy Game* case study offers a good example to validate the feasibility of our approach to model the requirements of a MAS covering its *organizational*, *structural*, *functional*, and *social behavior* properties.

6. Conclusions and future work

In this work we proposed a requirements modeling process for MAS. The approach is organized into two main activities: *Requirements Definition* and *Requirements Specification*. In the *Requirements Definition* activity the following is modeled: (a) the organizational structure and structural properties of the system; (b) the functional behavior of the system; and (c) the domain entities and their relationships. In the *Requirements Specification* activity the requirements specifications are refined, identifying: (a) the interactions on which the social behavior of the system is based; (b) the mains activities which conform the functional behavior of each role; (c) the permissions of the roles in the domain entities; and (d) the structural and functional behavior. This process supports the four perspectives that characterize a MAS: *organizational*, *structural*, *functional* and *social behavior*. We believe that this proposal addresses the need for a requirements modeling process for MAS because it incorporates specific abstractions needed to capture and specify these four perspectives. In particular, the definition and specification of features of social behavior at the requirements level will increase the quality of specifications, thus providing the expressiveness needed by the MAS in an early stage of the software development process.

As a case study, we have presented the requirements modeling of the *Diplomacy Game*. The game development domain, given its characteristics, particularly allows us to observe and reason about different ways in which to identify, define, and specify requirements of social behavior, in addition to the organizational (because of the various phases of which a game is composed), the structural (owing to the different types of elements used), and the functional (because of the different actions to be performed). Moreover, according to Google Trends and the ESA annual report (ESA, 2010), game development is one of the business markets that has undergone most growth in the last few years. The social behavior characteristics (negotiation, cooperation, pro-activity, etc.) and complexity of games make them appropriate subjects for resolution with the agent-oriented paradigm.

Currently, we are working on the definition and specification of other social behavior characteristics, such as adaptability and mobility. In addition, plan to empirically validate our approach through a series of experiments using game development experts as subjects.

We are also working to extend this agent proposal to a model-driven development approach in the context of a project named Multi-modeling Approach for Quality-Aware Software Product Lines (MULTIPLE). MULTIPLE focuses on the definition and implementation of a technology framework for developing software product lines of high-quality software in the context of model-driven development. This extension would facilitate the integration and traceability among the artifacts generated during the requirements modeling process and the analysis and design artifacts used in the MAS development. This will increase the overall quality of the MAS to be developed. Finally, we plan to build a tool that will support the overall process defined, using the Eclipse Development Environment (The Eclipse Fundation, 2010).

7. Acknowledgments

This research is funded by the Ministry of Education and Science, under the National Research Program, Development and Innovation, MULTIPLE project (TIN2009-13838).

Lorena Rodriguez has a scholarship under the College Scholarship Program and Support of the Scientific and Technological Production, in the context of the Social Responsibility Program of the Itaipu Binacional/Parque Tecnológico Itaipu-Py.

8. References

Anton, A. (1996). Goal-based requirements analysis. *Proceedings of the 2nd International Conference on Requirements Engineering (ICRE '96)*, pp.136–144, ISBN: 0-8186-7252-8, Colorado Springs, April 1996, IEEE Computer Society, Colorado

Blanes, D.; Insfrán, E.; Abrahão, S. (2009) a. Requirements Engineering in the Development of Multi-Agent Systems: A Systematic Review. *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, pp. 510 – 517, ISBN: 0302-9743, Burgos, Spain, September 2009, Springer-Verlag, Berlin, Heidelberg

Blanes, D.; Insfrán, E.; Abrahão, S. (2009) b. RE4Gaia: A Requirements Modeling Approach for the Development of Multi-Agent Systems. *International Conference on Advanced Software Engineering and Its Applications (ASEA'09)*, pp. 245 – 252, ISBN: 978-3-642-10618-7, Jeju Island, Korea, December 2009, Springer, Berlin

Cernuzzi, L.; Cossentino, M.; Zambonelli, F. (2005). Process models for agent-based development. *Engineering Applications of Artificial Intelligence*, 18, 2, (March 2005) page numbers (205 – 222), ISSN: 0952-1976

ESA (2010). Entertainment Software Association, Industry Facts. Last accessed on September 14, 2010, en http://www.theesa.com/facts/index.asp

Fabregues, A.; Navarro D.; Serrano A.; Sierra C. (2010). dipGame: a Testbed for Multiagent Systems, *Proceeding of the ninth International Conference* of *Autonomous Agents and Multi-Agent Systems*, Toronto, Canada, May 2010

Falkenberg, E.D.; Hesse, W.; Lindgreen, P.; Nilsson, B. E.; Oei, J.L.H.; Rolland, C.; Stamper, R. K.; Van Assche, F.J.M.; Verrijn-Stuart, A. A.; Voss., K. (1998). *FRISCO - A framework of information system concepts — The FRISCO Report.* Task Group FRISCO, ISBN: 3-901882-01-4

Giorgini, P.; Kolp, M.; Mylopoulos, J.; Castro, J. (2005). Tropos: A Requirements-Driven Methodology for Agent-Oriented Software. In: *Agent-Oriented Methodologies*, Brian Henderson-Sellers; Paolo Giorgini, page numbers (20-45), Idea Group , ISBN: 1591405815, USA

Gómez-Sanz, J.; Pavón, J. (2003). Agent Oriented Software Engineering with INGENIAS, *Proceedings of the 3rd Central and Eastern Europe Conference on Multiagent Systems (CEEMAS '03)*, pp. 394 – 403, ISBN: 3-540-40450-3, Prague, june 2003, Springer-Verlag, Prague

Hector, A.; Lakshmi Narasimhan, V. (2005). A New Classification Scheme for Software Agents. *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, pp. 191 – 196, ISBN: 0-7695-2316-1, Sydney, Australia, July 2005, IEEE Computer Society, Washington, DC

Huzam S. F. Al-Subaie; Maibaum Tom S. E. (2006). Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method. *Proceedings of the Fourth International Workshop on Comparative Evaluation in Requirements Engineering*, pp. 8 - 19, ISBN: 0-7695-2712-4, Minneapolis/St. Paul, Minnesota, September 2006, IEEE Computer Society Washington, DC

Maiden, N. (1998). CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements. *Automated Software Engineering*, 5, 4, (October 1998) page numbers (419 - 446), ISSN: 0928-8910

Odell, J.; Parunak, H. V. D.; Bauer, B. (2000). Extending UML for agents. *Proceeding of the 2nd Int. Workshop on Agent-Oriented Information Systems*, pp. 3 – 17, Berlin, iCue Publishing

OMG (Object Management Group). Software Process Engineering Meta-Model (SPEM), version 1.1. Last accessed on March 11, 2010, en http://www.omg.org/cgi-bin/doc?formal/05-01-06.pdf

Rodriguez L.; Hume, A.; Cernuzzi, L.; Insfrán, E. (2009). Improving the Quality of Agent-Based Systems: Integration of Requirements Modeling into Gaia. *Proceedings of the Ninth International Conference on Quality Software (QSIC '09)*, pp. 278 – 283, ISBN: 978-0-7695-3828-0, Jeju, Korea, August 2009, IEEE Computer Society Washington, DC

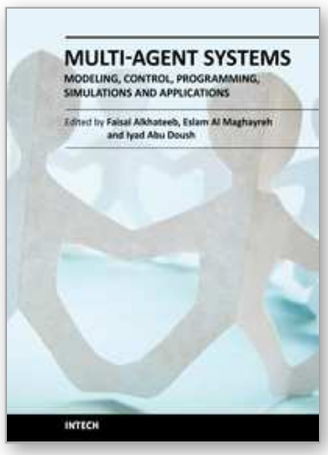
The Eclipse Foundation. Last accessed on September 2010, from http://www.eclipse.org

Van Lamsweerde, A.; Darimont, R.; Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. *IEEE Transactions on Software Engineering*, 24, 11, (November 1998) page numbers (908 – 926), ISSN: 0098-5589

Wizards (2010). The rules of Diplomacy. The game of international intrigue. . Last accessed on September 14, 2010 en http://www.wizards.com/avalonhill/rules/diplomacy_rulebook.pdf

Yu, E. (1997). Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE '97)*, pp. 226 – 235, ISBN: 0-8186-7740-6, Washington D.C., USA, January 1997

Zambonelli, F.; Jennings, N.; Wooldridge, M. (2003). Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12, 3, (July 2003) page numbers (317 – 370), ISSN: 1049-331X



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications

Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen