ARTÍCULO

Approaches for the Analysis and Design of Multi-Agent Systems

Pablo Villarreal[1], Maximiliano Alesso[1], Sebastián Rocco[1], María Rosa Galli[1,2] and Omar Chiotti[1,2]

[1]GIDSATD - Universidad Tecnológica Nacional - Facultad Regional Santa Fe
Lavaisse 610 - (3000) Santa Fe - Argentina
{pvillarr,malesso,srocco}@frsf.utn.edu.ar

[2]INGAR-CONICET
Avellaneda 3657 – (3000) Santa Fe - Argentina
{mrgalli,chiotti}@ceride.gov.ar

Keywords: Organizational Concepts, Business Process, Analysis and Design, Multi-Agent Systems.

This work firstly presents an approach based on the organizational concept of business processes to identify roles and protocols as part of the analysis stage of a methodology for agent-oriented analysis and design. Secondly, the work presents an approach to help to decide when using a mobile agent in a multi-agent architecture is better than using a static one. This approach is based on both the quality attributes specified for the multi-agent architecture and the execution environments of the multi-agent system. A real case is used to exemplify these approaches.

Approaches for the Analysis and Design of Multi-Agent Systems

Pablo Villarreal[1], Maximiliano Alesso[1], Sebastián Rocco[1], María Rosa Galli[1,2], Omar Chiotti[1,2]

[1]GIDSATD - Universidad Tecnológica Nacional - Facultad Regional Santa Fe
Lavaisse 610 - (3000) Santa Fe - Argentina
{pvillarr,malesso,srocco}@frsf.utn.edu.ar

[2]INGAR-CONICET
Avellaneda 3657 – (3000) Santa Fe - Argentina
{mrgalli,chiotti}@ceride.gov.ar

Abstract

This work firstly presents an approach based on the organizational concept of business processes to identify roles and protocols as part of the analysis stage of a methodology for agent-oriented analysis and design. Secondly, the work presents an approach to help to decide when using a mobile agent in a multi-agent architecture is better than using a static one. This approach is based on both the quality attributes specified for the multi-agent architecture and the execution environments of the multi-agent system. A real case is used to exemplify these approaches.

Keywords: Organizational Concepts, Business Process, Analysis and Design, Multi-Agent Systems.

1. Introduction

Agents constitute a new and innovating technology for developing information systems. They bring about the study and development of methodologies that compete in introducing concepts to guide the multi-agent systems (MASs) analysis and design. Some proposals extend the object oriented methodologies and techniques to the design of MASs [Kendall99][Odel00]. However, these proposals fail to capture the autonomous and proactive behavior of agents, as well as the richness of the interactions [Zambonelli01b]. Other approaches attempt to model and implement MASs from an organization-oriented point of view. Gaia and MaSE methodologies [Butte02] can be mentioned as examples. These methodologies derive mainly from the autonomous nature and the proactive behavior of agents. This principle exhibits the convenience of thinking a system as an organization in which agents play *roles* and participate of *interactions* among roles.

One methodology for analysis and design of MASs is Gaia [Woooldridge00]. The key concepts of the analysis in Gaia are roles and protocols. Roles can interact with one another in certain institutionalized ways, which are defined in the protocols. Although Gaia claims to allow an analyst to go systematically from a statement of requirements to a sufficiently detailed design, it does not present a procedure to guide roles and protocols identification. A recent paper [Zambonelli01a] introduces three organizational concepts (organizational rules, organizational structures and organizational patterns) and discusses why they believe they are necessary for the complete specification of MASs. They use these organizational abstractions to extend

the Gaia methodology. Another methodology for developing heterogeneous MASs is MaSE [DeLoach01]. It uses graphically based models to describe systems goals, behaviors, agent types, and communication interfaces. We believe that the same organizational principle is the source of other concepts that can help in the analysis and design of systems as multi-agents organizations.

Furthermore, besides autonomy, reactivity, pro-activeness and social ability [Woooldridge95], another software agent's property is mobility. Mobility refers to the capability of an agent for dynamically transferring its execution onto different sites [White97]. Although mobile agents technology present several advantages when compared to middleware technologies used in developing distributed applications, such as RPC or CORBA [Chess97], the advantage of using mobile agents versus using static agents is not quite clear. Both mobile and static agents can be used to solve the same problems. The difference lies on how each alternative solves the problem. In spite of the wide diffusion that agent mobile technology is having currently, there are few proposes that settle directions for a designer to determine when mobile agents are convenient to be used or not. The proposed methodologies for developing MASs do not provide methods to determine in which cases mobile agents should be used.

Firstly, this work is aimed at presenting an approach to guide the roles and protocols identification in the analysis stage of a MAS, as part of the Gaia methodology. This approach is based on organizational concepts. Secondly, the work presents an approach to identify mobile agents in a multi-agent architecture. A real case is used to exemplify these approaches.

2. Gaia Methodology

Gaia methodology [Woooldridge00] proposes a developer to think of building agent-based systems as a process of organizational design. The main concepts in Gaia are divided into two categories: abstract and concrete. The abstract concepts involve the *Roles*. A role is defined by four attributes: *Protocols*, *Permissions*, *Activities* and *Responsibilities*. Responsibilities are divided into two types: *Liveness Properties* and *Safety Properties*. The concrete concepts involve: *Agent types*, *Services* and *Acquaintances*. A role can be viewed as an abstract description of an entity's expected function. In other words, a role is more or less identical to the notion of an office. A protocol defines the way that a role can interact with another role. The analysis stage implies defining the role model and the interaction (protocols) model. In this work, the terms of Gaia are defined in the different sections where they are necessary. More details of Gaia methodology can be obtained in [Woooldridge00].

3. An Approach to Identify Roles and Protocols

This section presents an approach based on the concept of business processes [Hammer90] to identify roles and protocols as part of the analysis phase in Gaia Methodology. Following, we describe each stage of the approach.

3.1. Stage 1: Define the system's goal

Every information system pursuits a certain purpose for which it was conceived and put into operation. Starting from an organizational abstraction of the MAS, it is necessary to take into account that every organization must define a goal and direct all efforts of every agent of the organization towards that goal. Then, the first step is to define the goal of the MAS.

3.2. Stage 2: Define the processes

Current management techniques point out that it is convenient to coordinate organization activities according to a set of business processes [Hammer90]. Once the goal is defined, it is possible to define one or more processes whose primary objectives are to reach that goal. In this way, processes are useful to describe what must be done to reach the goal of the organization. Then, by considering the process, its activities, inputs and outputs, its clients, and its environment, we obtain the knowledge needed for the specification of the roles to be played in the organization and the protocols among those roles.

A process can be defined by means of the following:

- Their inputs and outputs, determining which entities (systems or humans) of the organization provide inputs and which ones, receive their outputs.
- The set of activities and their respective precedence relationships.

The concept of process activity is also quite important for our methodological proposal. Therefore, each activity of the process is defined in an activity template and characterized by the following attributes: activity objective, inputs,

outputs, tasks involved in the activity, constrains and resources (information required by the activity).

Once processes are defined, we obtain all activities that the organization has to carry out, their precedence relationships and the possible activity sequences or control flows of the processes. Fulfillment of a process activity allows reaching its objective, and the achievement of successive objectives involved in each activity allows the organization (the system) to reach the goal. In addition, the activity sequences or control flows of the processes determine how the activities (system functions) have to be coordinated in the organization (the system), describing the dynamic and operative aspects of the organization. In this way, the organization processes establish the form in which the roles played by agents in an organization, which carry out the process activities, have to be coordinated to reach the organization's goal.

3.3. Stage 3: Identify Roles and Protocols

Following the processes definition, each activity in the aforementioned processes should be modeled, defining the role to be played in the organization to achieve the objective of that activity. The role should accept the constraints and use the available resources of that activity. The interactions involved in that role are also defined. Thus, the roles and protocols of the whole system are obtained. In this definition, some points should be taken into account:

- Process activities define the roles to be played to carry out these activities.
- Performing an activity in the context of a process imply different interactions with other process activities, through precedence relationships. Such interactions are expressed by the activity inputs and outputs, and their precedence relationship. These interactions can define protocols.

3.4. Stage 4: Define the role's schemas and the protocol's attributes

In this stage roles model and interactions model are completed, defining the role's schemas and the protocol's attributes. Roles model and interactions model define the roles that the agents of an organization should perform and their interactions to reach their goal. A role is defined by four attributes: responsibilities (liveness and safety properties), permissions, activities and protocols. These attributes are defined in the role's schema. We can use some elements defined in the processes, which have a mapping with some elements of the role's attributes:

- A process activity can be represented by an atomic task sequence that has to be carried out as a transaction. The task concept is the same as the role activity concept in Gaia. Thus, tasks define the activities of a role. An activity of a role correspond to a unit of action that a role performs without interact with other role.
- Process activities constraints generally define safety properties of the role that carry out those activities.
- The resources identified in a process activity define the role permissions. Role permissions describe the information resources that can be used to perform the role and state the rights to access these resources, with which the role has to operate.
- The activities sequences and their precedence relationships in the processes define the liveness property. It defines the potential execution trajectories through the activities and protocols associated with the roles. The livenness property describes those states that a role must bring about, given certain environmental conditions.

These four stages define the analysis phase. Afterwards, the agent model is defined, and so are the services model and the acquaintance model. For that purpose, we follow the steps proposed in the design phase of the Gaia methodology [Woooldridge00].

3.5. Summary of the Analysis and Design Procedure

Applying our approach to identify roles and protocols, with the Gaia methodology, the resulting complete procedure for the analysis and design of a MAS can be summarized as follows:

Analysis phase:

Stage 1: define the system's goal.

Stage 2: define the necessary processes to achieve that goal.

Stage 3: identify roles and protocols in order to define the roles model and the interactions model.

Stage 4: define the role's schemas and the protocol's attributes in order to complete roles model and interactions model (Analysis phase in Gaia).

Design phase:

Stage 5: define the agent, the services and the acquaintance models (following the steps proposed in the design phase of Gaia).

3.6. Application of the Analysis and Design Procedure

This procedure has been used to the analysis and design of a system to support ad-hoc information requirements by different agents that conform a real organization. This system is known as Dynamic Decision Support System. Therefore, we will use such system as a real case to explain the aforementioned procedure.

3.6.1. Stage 1: Define the system's goal.

The defined goal for the Dynamic Decision Support System is: "Allows geographically distributed users of an organization to obtain answers to their information requirements so that the decision making processes they must carry out can improve their efficacy by using a flexible and dynamic means of information exchange".

3.6.2. Stage 2: Detailed definition and description of the processes needed to achieve that goal.

Definition of inputs and outputs of the processes are represented in Figure 1.

From the analysis of inputs and outputs of processes, it can be said that:

- There are people (User A) in the company that would like to ask for some information. It has not been foreseen that this information could be required. It already exists or can be generated in the company but it is not known who has it or can generate it.
- There are people (User B) that could receive the query and generate an answer since they have the requested information.
- The user that originated the query would like to consult the state of his query.
- The person that originated the query will then receive one or more answers from the different sources of information.
- The person that originated the query could to cancel it after the search for information has started.

It is clear that the users supported by the system provide the inputs and receive the system outputs, so these users are the entities of the system. Such users perform activities in geographically distributed places in the company, which are called *Domains*. It should be stressed that domains are autonomous and each of them has information systems to support the decision processes that are carried out inside them.

From this brief description of the system requirements, we gather that there are three main processes: (I) get the required information, (II) get the states of an issued query for information and (III) cancel a query. Processes have been designed following the process design methodology proposed in [Klein94] and have been represented using UML diagram activity.

For the purpose of showing our approach, we only show the main process I (Figure 2) which have to be carried out by the system in its abstraction as an organization to achieve the system's goal. This process consists of six activities: the first activity has the object of obtaining the user's query. The second activity has the object of determining the possible sources for providing the required information. The third activity has the object of visiting the domains that the previous activity classified as possible providers of the required information. The forth activity has the object of obtaining the answer to the query, the fifth activity has the object of delivering the answer to the user that issued the query, and the sixth activity has the object of learning from this search, thus improving the knowledge about domains as regards the information they can provide.

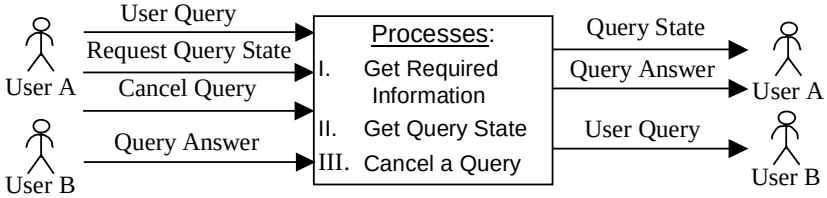

Figure 1. Intputs and outputs of the proceses

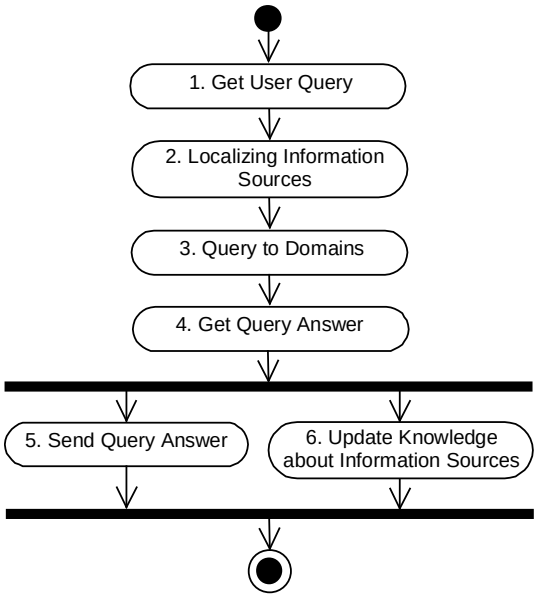

Figure 2. Process I: Get Required Information

Activity	*Localizing Information Sources*
Object	Determine the domains that are potentially capable of providing the required information.
Inputs	User query.
Outputs	List of domains to be queried.
Tasks	1. Receive a user query for identify possible information sources. 2. Relate a user query in natural language with the information kept in domains. 3. Generate a list of the domains that are potentially capable of answering the user query.
Constraints	Maximum number of queries to be received Maximum number of queries to answer simultaneously.
Resources	Knowledge Base about information managed by Domains.

Figure 3. Attributes of Localizing Information Sources activity

To complete the processes definition, the activity attributes are defined. Each of the defined process activity is described in detail. As an example, Figure 3 presents the *Localizing Information Sources* activity, which correspond to the main process.

3.6.3. Stage 3: Identify Roles and Protocols.

Figure 4 shows the defined roles and protocols derived from the main process. Process activities assigned to each identified role are shown by a tuple (Process.Number; Activity.Number). Protocols that settle interactions among roles are shown with arcs among roles. Activities of the roles are showed with a loop.

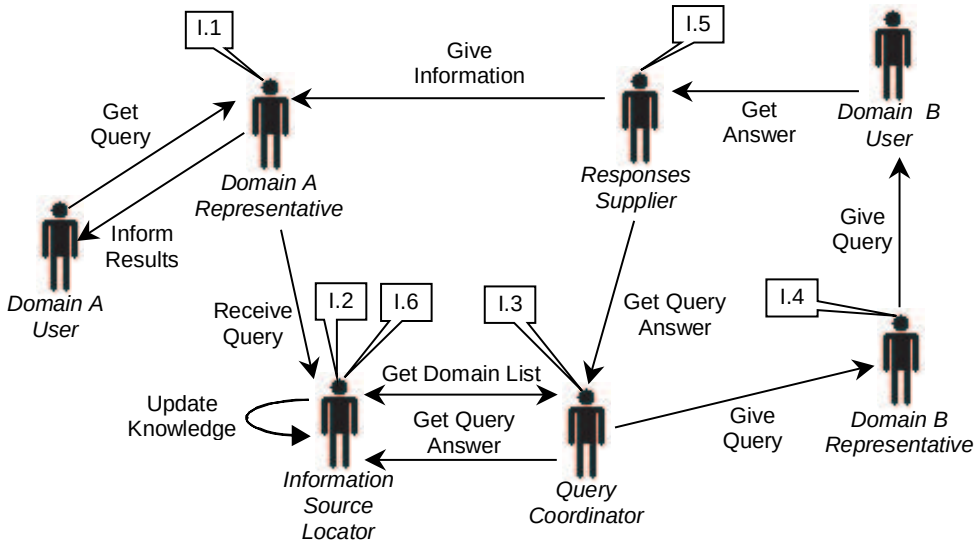

Figure 4. The Roles and Protocols from the Process I.

3.6.4. Stage 4. Define the role's schemas and the protocol's attributes.

Once the roles and protocols are identified, we detailed each role and each protocol. We have made this following the Gaia methodology proposal. As an example, Figure 5 shows a schema of the *InformationSourceLocator* role.

Role Schema: *InformationSourceLocator*
Protocols and Activities: ReceiveQuery, MatchQuery, GenerateDomainsList, GiveDomainsList, GetQueryAnswer, UpdateKnowledge
Permissions: reads *UserQueries* *QueryAnswer* changes *BaseKnowledge*
Responsabilities Liveness: InformationSourceLocator = (ReceiveQuery. MatchQuery.GenerateDomainsList. GiveDomainsList) || (GetQueryAnswer.UpdateKnowledge) Safety: UserQueries < n // number of user queries attended simultaneously QueryAnswers < m // number of query answers attended simultaneously

Figure 5. Schema of the role *InformationSourceLocator.*

3.6.5. Stage 5. Define the agents, the services and the acquaintance models.

Following the design steps of Gaia methodology, in this stage the agents, the services and the acquaintance models are defined. The agent model identifies the agent types that will make up the system, the agent instances that will carry out these agent types at run-time, and the mapping between roles and agent types. The services model identifies the services that are required to carry out by each agent and specifies the main properties of these services. In Gaia, a service means a function of the agent. The services are derived from the protocols, activities, responsibilities and the liveness properties of a role. The acquaintance model documents the communication lines between the different agents.

Figure 6 shows three identified agents types, the Domain Representative Agent (DRA), the Information Source Locator Agent (ISLA) and the Query Coordinator Agent (QCA). In this model it can see the mapping among roles and agents, which defines the roles that each agent perform. The number of DRA instances will depend of the number of domains to be supported by the system, due to will exist a DRA to each domain. Once the agent model was defined we must define the services model. We won't show this model because of space limitations.

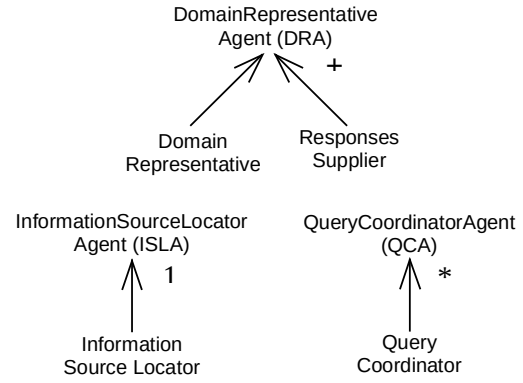

Figure 6. The agent model

Figure 7 shows the acquaintance model. This is the last model to be generated. It shows the communication links between agent types. The DRA has a loop that point out a communication link between agents of the same type.

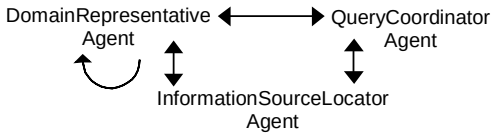

Figure 7. The Acquaintance Model

4. An Approach to Identify Mobile Agents

An agent should meet the following properties: autonomy, reactivity, pro-activeness and social ability [Wooldridge95]. These properties are present in roles performed by an agent and thus they are taken into consideration when modelling the agent. Internal activities of an agent, and the interactions (protocols) among agents, can be always carried out

locally in the same environment (using mobile agents) or in a distributed way (using static agents). The difference lies on the advantage a solution may have over another. These advantages can be identified if we take into account the software quality attributes the developing MAS should possess. Then, it is possible to settle some criteria to decide whether an agent must meet the mobility property according to certain quality attributes the MAS is intended to have. The following section describes some quality attributes a mobile agent may meet. Then, we discuss the need of identifying execution environments, and finally we present an approach to identify and decide when it is convenient that an agent meets the mobility property.

4.1. Software Quality Attributes Related to Mobile Agents

Software architectures design [Bass98] has been the focus of considerable research, which has resulted in a collection of well-understood architectural styles and a methodology for evaluating their effectiveness with respect to particular software qualities. However, there exist few evaluations of MASs in terms of software qualities. Some software quality attributes for multi-agent architectures were identified from a perspective of organizational styles as architectural styles: predictability, security, adaptability, coordinability, and so on [Kolp01]. Quality attributes are not all related with choice between mobile agents or static agents. For example, security is a quality attribute that can be accomplished using a solution with an architecture where there are some static agents that support the security requirements or another architecture where some mobile agents exist. Using mobile agents does not contribute to accomplish the security requirements. In addition, there are different approaches to support security in applying mobile agents because security is a concern in these applications. However, we think that there are some quality attributes of MASs that mobile agents can contribute to reach. This quality attributes are:

Performance. If an agent must interact with a great number of agents that are located in distributed execution environments, the use of mobile agents may allow for an improved system performance, considering both performance perceived by the user and the network performance. This can be achieved because one of the mobile agents' characteristics is that they can perform the same activity in different sites in a parallel way. This parallelism is possible due to the mobile agents' ability to clone, which enables the same activity type to be performed by the same type of agent in different sites and at the same time. This parallelism is possible when the tasks to be executed are independent. A comparison between mobile and static MASs from the performance point of view was carried out [O'Malley00]. In that work, solutions to a problem of distributed text search in a network files system were developed, using different approaches to the MAS (static versus mobile) and these alternatives were tested. They conclude that the mobile agents system yields a better performance than the static agents system when cloning is used to carry out the search.

Reliability. Using mobile agents may increase reliability and availability measures related to certain aspects. For example, an agent may have to interact with several agents in different execution environments. If this agent is static and the environment where it is executed is down, the agent is down too. In contrast, if this agent is mobile, once it has moved to another execution environment, it can go on with its execution and reach its objective, even when its execution environment of origin is down. In addition, a mobile agent can choose the activities to carry out according to its knowledge about the state (up or down) of others agents and its execution environment. So, mobile agents contribute to improve the ability of a MAS to keep on operating over time to reach its goals.

4.2. Execution Environments

Agents are situated entities of a MAS considered as an organization. As situated entities, agents have an environment in which they are executed and interact with other agents. Generally, agents are executed in environments that may be dynamic, open, distributed and heterogeneous. Therefore, we think that it is not appropriate to model a MAS without modeling and defining the execution environments of it and the environment in which each agent will be executed. Different environments in a MAS can affect the way in which system goals are reached so they can affect the design of activities and interactions of each agent. In this paper, we only discuss identification of execution environments and we do not discuss how modeling this execution environment. Identification of execution environments of MASs will help us to decide when using a mobile agent is appropriate.

4.3. Guides to Identify Mobile Agents

Our aim is to define a guide to help us to decide when is better to use a mobile agent than a static one. To identify mobile agents we define the following steps:

a) Identify the execution environments in which each system agent has to carry out its tasks and interactions.

b) Identify communication pathways between agents of different execution environment. This may be carried out considering the acquaintance model and the defined execution environments. In this way, we can identify a priori the agents that may be defined as mobile ones.

c) If some quality attributes for the multi-agent architecture related with mobile agents are specified, it is possible to define the mobility property for the agents identified in the previous step.

In this way, analyzing execution environments and looking in the acquaintance model, we can decide use mobile agents to reach some quality attributes of a MAS.

4.4. Identifying Mobile Agents in the Studied Case

In the studied case, we have identified two types of execution environments. Let's consider the DRA associated to a domain; anytime, a user may want to make a query or queries from other domains may arrive asking for information. In this way, the DRA, which interacts with the user, must be executed in the domain environment. Since domains are geographically distributed, there will be an execution environment for each domain. Furthermore, due to reliability reasons, it is convenient to have another distributed environment where the ISLA and the QCAs will be executed.

If we observe communication links from the acquaintance model in Figure 7, it can be identified that interactions among DRAs, ISLA and QCAs are performed through distributed environments just as interactions among DRAs. Then, it is convenient to develop the DRA as a static agent since it must be always available to receive queries or answers from users of its domain or queries or answers of other domains. Due to its main role in the architecture and to the fact that there will be a sole instance of it, the ISLA should be developed as a static agent. Moreover, the ISLA makes it possible that DRAs need to know only one agent to try to obtain an answer to any kind of query. As regards QCAs, there is no requirement that compel them to be static. In this way, the mobility property could be considered for QCAs.

In order to decide whether it is necessary for QCAs to be mobile, we resorted to the quality attributes specified for the MAS architecture. Among the attributes defined for the system, we mention performance and reliability. As we have previously discussed, these attributes may be met by using mobile agents.

Performance was defined from the reduction in the network load and the time to response the user. To reduce the network load, it is necessary to reduce messages in the network. This can be achieved by using mobile agents, due to it is cheaper to bring the interactions to give the user queries and to obtain the answers to this queries where the interactions can take place locally. Thus, the number of messages necessary to maintain these conversations between agents is eliminated [Butte02]. Response time can be reduced by using mobile agents, with the cloning technique, as show in [O'Malley00].

Reliability was defined as the need for the system to be able to achieve its goal, no matter whether the DRA that sent the query and the ISLA are in execution or not. Maybe a query can be solved in hours because a domain that must answer is not available. Therefore, it is necessary to settle a reliable mechanism that allows the system to keep on operating to satisfy the search. This can be fulfilled by adding mobility to QCAs. In this way, QCAs can visit domains and stay there waiting for a DRA answer for a certain period of time, no matter whether the ISLA and its execution environment are active or not. Moreover, as it is mobile, the QCA can decide which domain it will visit taking into account the ranking of domains, the DRA and its execution environment availability, to which it must deliver the answer.

Therefore, given the performance and reliability quality attributes to be met and according to what has been previously mentioned, we have added the mobility property to the QCA agent type. Thus, the MAS architecture is now conformed by two types of static agents (DRAs and ISLAs) and one type of mobile agents (QCAs).

5. Conclusions

In this work, we have presented an approach to identify roles and protocols as part of the analysis phase of Gaia methodology, which is based on the organizational concept of business process. The business processes of an organization define all the activities that the organization has to perform to reach its goal. From a point of view of the information system as an organization, the process activities define the system functionalities. In addition, the activities precedence relationships and the activities sequences of processes define the way

in which the activities have to be coordinated in the organization. These processes elements define the way in which the system functions have to be coordinated. Therefore, we propose to start modeling processes to define the functional and coordination aspects of the organization and then, based on processes, to derive the structure of the organization most suitable to reach the organization goal. In this way, the structure of roles is derived according to the activities and their coordination requirements stated by the processes.

Sometimes, when the real-world organization that the MAS is designed to support exists, may be direct to derive the roles of MAS based on real-world organization roles. However, when the real-world organization does not exist or it is not well structured or the MAS is thought to change the real-world organization, it is difficult to identify roles and their interactions that the organization requires. In those cases, we think it is appropriate to model the organization processes and then, taking into account the processes, to identify the organization roles and protocols. However, our approach can also be used when real-world organization exists, due to it helps to define the roles and protocols attributes and the way in which the roles must interact among them.

Finally, we have described other approach in which the quality attributes specified for a multi-agent architecture and the execution environment of each agent type of a MAS allow to decide when using a mobile agent is better than using a static one.

References

[Bass98] Bass,L. , Clements, P. , Kazman, R. "Software Architecture in Practice". Adisson-Wesley . (1998).

[Butte02] Butte, T. "Technologies for the Development of Agent-based Distributed Applications". ACM Crossroads. (Spring 2002).

[Chess97] Chess, D., Harrison, C.G., Kershenbaum, A. "Mobile agents: Are they a good idea?". In Mobile Object Systems: Towards the Programmable Internetwork. Ed. by J. Vitek and C. Tschudin. Berlin, Germany: Lecture Notes in Computer Science pp. 25-47. (1997).

[DeLoach01] DeLoach, S., Wood, M., Sparkman, C. "Multiagent Systems Engineering". International Journal of Software Engineering and Knowledge Engineering. Vol. 11, no.3. pp. 231-258. (2001).

[Hammer90] Hammer, M. "Reengineering Wok: Don't Automate, Obliterate". Harvard Business Review. pp. 104. (1990)

[Kendall99] Kendall, E. "Role Modeling for Agent System Analysis, Design, and Implementation". First International Symposium on Agent Systems and Applications (ASA'99), Third International Symposium on Mobile Agents (MA'99), Palm Springs. (1999).

[Klein94] Klein, M. "Reengineering Methodologies and Tools." Information Systems Management. pp. 30-35. (1994).

[Kolp01] Kolp, M., Mylopoulos, J. "Software Architectures as Organizational Structures". In Proceedings ASERC Workshop on "The Role of Software Architectures in the Construction, Evolution, and Reuse of Software Systems", Ed. monton, Canada. (2001).

[O'Malley00] O'Malley, S.A., Self, A.L., DeLoach, S.A. "Comparing performance of static versus mobile multiagent systems". Proceedings of the National Aerospace and Electronics Conference. (2000).

[Odell00] Odell, J., Parunak, H.V.D, Bauer, B. "Extending UML for Agents". Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence, Gerd Wagner, Yves Lesperance, and Eric Yu eds, Austin, TX, pp 3-17 (2000).

[White97] White, J. "Mobile Agents. In Software Agents". Ed. Bradshaw, AAAI Press. pp. 437-472. (1997).

[Wooldridge95] Wooldridge, M., Jennings, N. "Intelligent agents: Theory and Practice". The Knowledge Engineering Review, Vol. 10, no 2. pp. 115-152. (1995).

[Woooldridge00] Woooldridge, M., Jennings, N., Kinny, D. "The Gaia Methodology for Agent-Oriented Analysis and Design". Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3, no 3, 2000.

[Zambonelli01a] Zambonelli, F., Jennings, N. R., Wooldridge, M. "Organizational rules as an abstraction for the analysis and design of multi-agent systems." International Journal of Software Engineering and Knowledge Engineering. Vol. 11, no 3. pp. 303-308 (2001).

[Zambonelli01b] Zambonelli, F., Jennings, N., Omicini, A, Wooldridge, M. "Agent-Oriented Software Engineering for Internet Applications". In Coordination of Internet Agents. Eds. A. Omicini, F. Zambonelli, M. Klusch and R. Tolksdorf. Springer Verlag, pp. 326-346 (2001).