4

Modelling Multi-Agent System using Different Methodologies

Vera Maria B. Werneck[1], Rosa Maria E. Moreira Costa[1] and Luiz Marcio Cysneiros[2]
[1]Universidade do Estado do Rio de Janeiro,
[2]York University,
[1]Brazil
[2]Canada

1. Introduction

The increasing use of multi-agent systems brings challenges that have not been studied yet, such as: how we should adapt requirements elicitation to cope with agent properties like autonomy, sociability and proactiveness. The agent-oriented modelling is proposed as a suitable software engineering approach for complex organizational application domains that deal with the need for new applications. These requirements are not broadly considered by current paradigms. Autonomy and sociability aspects such as the dependency of an agent on another, and how critical this condition should be, have to be analysed from the early stages of the software development process (Wooldridge & Jennings, 1997).
This research work is included in the Agent-oriented Project that has been developed by the Informatics and Computer Science Department of State University of Rio de Janeiro (UERJ) and the School of Information Technology of York University (Toronto). This project aims at studying and comparing agent-oriented software development methods and techniques based on attributes and norms and by the models construction based on an exemplar. These experiments enabled the development and construction of Multi-Agent Systems applied to Health and Education areas, providing research on Systems (MAS) especially on the agent proliferation of control, communication and availability of information and knowledge in different computing environments.
The construction of Multi-Agent Systems allows experiments on the agent-oriented technology in relation to development methodologies with regard to agent-oriented programming environments. It also allows us apply this technology in practical and real applications in Health and Education Domains. The Glycemic Monitor System based on the Guardian Angel for aiding the diabetes treatment (Tavares et al., 2010) and the Educ-MAS (Education Multi-Agent System) (Gago et al., 2009), (Dantas et al., 2007), a learning education environment with multi-agents helping the teaching process on a specific topic, are two examples of Multi-Agent Systems that have been developed in the project Oriented Agents.
Many methodologies applying agent-oriented concepts to software development have been proposed however, the evaluation of these methodologies is not an easy task specially to

choose the best method to be adopted in a MAS project. In this project, we have used an exemplar proposed by Yu & Cysneiros (2002) to evaluate some methodologies and language methods (Gaia, MESSAGE, Tropos, Adelfe, MAS-CommonKADS, MaSE, Ingenius, KAOS, AUML) (Souza et al., 2010), (Souza et al., 2009), (Werneck et al., 2008), (Werneck et al., 2007), (Werneck et al., 2006), (Coppieters et al., 2005), (Cysneiros et al., 2005), (Cysneiros et al., 2005a). This exemplar is rich and complex enough to guide us to investigate to understand them better. Now we are compiling the experiences we gathered from all the methodologies we evaluated to try and understand where most methodologies need to improve and where most of them are well developed. This knowledge will be modelled into an ontology and will be used to define an Agent-Oriented Methodology Approach based on the Situation Method Engineering (SME) that provides a flexible way of constructing a methodology based on a set of method fragments and the situation of the project requirements. This idea of using SME for constructing Agent-Oriented Methodology was also proposed by Henderson-Sellers & Ralyté (2010) that describes some experiences of using SME in MAS and object oriented methods.

This chapter provides a deep modelling overview of two different Multi-agents systems in two different Agent-Oriented Methodologies. Our objective is to demonstrate how modelling the problem with a methodology can improve quality and be a guide for further MAS development.

This chapter is organized into 5 sections. Section 2 gives an overview of Multi-Agent Systems methodologies describing the Adelfe (Bernon et al., 2003) (Henderson-Sellers & Giorgini, 2005), and Mase (Deloach, 2001), (O'Malley et al., 2001), (Dileo et al., 2002), (Henderson-Sellers & Giorgini, 2005) methodologies that will be shown in the next two sections. Section 3 describes the modelling of Guardian Angel System in Adelfe focusing on the mains aspects of agent oriented. Section 4 presents the modelling of the Educ-MAS using MaSE. Finally section 5 analyses the systems development with those methodologies concluding the work and also presents correlated and future works.

2. MAS methodologies

Many agent-oriented methodologies have been proposed based on a variety of concepts, notations, techniques and methodological guidelines. Some of these methodologies rely on standard methods or modelling languages as CommonKADS (Schreiber et al, 1999) and UML (Rumbaugh et al., 2004). The MAS-CommonKADS (Iglesias & González, 1998), (Henderson-Sellers & Giorgini, 2005) and AUML (2007), (Odell et al., 2001) extended CommonKADS (Schreiber et al., 1999) and UML (Rumbaugh et al., 2004) respectively to meet the multi-agent systems.

The agent-oriented methodologies have multiple roots (Figure 1). Some are based on the idea of artificial intelligence coming from the knowledge engineering (KE). Other methods originate from software engineering and they are extensions of object-oriented (OO) paradigm. There are still those that use a mix of concepts based on these two areas and some are derived from other agent-oriented methodologies.

Although we are going to present two methodologies based on Object Oriented in this chapter, both Adelfe and MaSE methodologies were chosen because they are based on common agent concepts and they are easy to understand having a good methodology guide and a tool support. A tool is a very important issue that can be a differential in the software development.

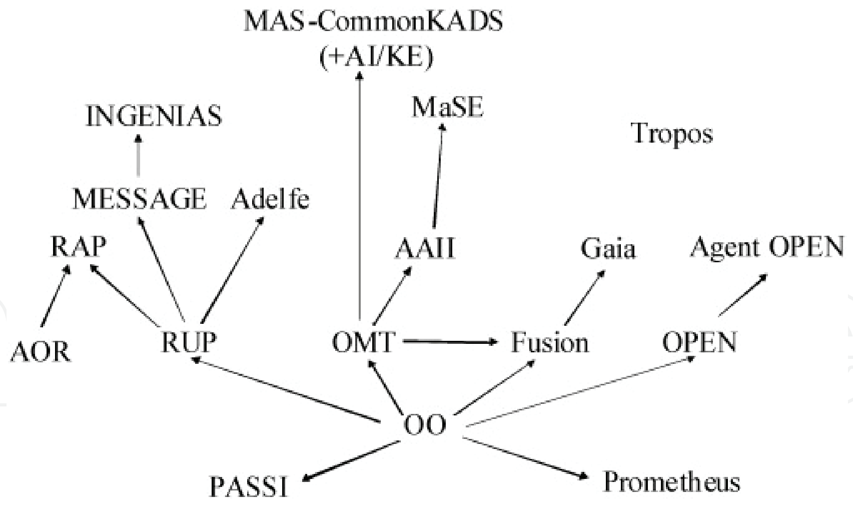


Fig. 1. Influences of Object-Oriented Methodologies on Agent-oriented Methodologies (Henderson-Sellers & Giorgini, 2005)

2.1 The Adelfe methodology

Adelfe is an acronym that translated from French means "framework to develop software with emergent functionality" (Adelfe, 2003), (Bernon et al., 2003), (Henderson-Sellers & Giorgini, 2005) and was developed to deal with open and complex agent problems. These systems work with composed agents that have cooperative interactions with each other and are called Adaptative Multi-Agent Systems (AMAS). .
Adelfe uses AUML principle (Odell et al., 2001), (AUML, 2007) together with UML (Rumbaugh et al., 2004) to express agent interaction protocols.
The development process of Adelfe is based on RUP (Rational Unified Process) (Krutchen, 2000) with some additions considering AMAS Theory specificities. For example, the environment characterization of the system and the identification of cooperation failures are some characteristics included in this process.
Adelfe provides some tools including one to estimate the AMAS technology adequacy. This can be a great support to inexperienced developers in the AMAS system field. The adequacy is studied at two levels: the global (the system) and the local (the components). Eight parameters are taken into consideration for the global level while for the components there are other three parameters.
Two other tools (Open Tool and Interactive Tool) are available to integrate the framework. The Open tool is a graphic modelling tool which supports Adelfe notation to construct the artefacts proposed in this method such as some UML diagrams and protocols of AUML interaction. The Interactive Tool provides the developer with a guide throughout the process application.
The Adelfe process covers all the phases of a classical software process from the requirements to the deployment based on the RUP process adapted to AMAS. Only the work definitions (WD) of requirements, analysis and design require modifications to be adapted for the AMAS. The rest of the RUP can be applied without modifications.

2.1.1 Preliminary and final requirements (WD1 e WD2)

The preliminary requirements work definition (WD1) of Adelfe (Adelfe, 2003), (Bernon et al., 2003), (Henderson-Sellers & Giorgini, 2005) is the same as proposed by the RUP (Table

1). The aim still consists of studying the stakeholders' needs to produce a document of the stakeholders and the developers´ agreement.
The activity of the environment characterization (A6) of the final requirements (WD2) (Table 1) was added to the RUP because the environment is a very important concept in AMAS theory. The environment has to be well comprehended and the A6 activity has the following tasks: determine the entities, define the context and characterize the environment. The characterization begins by the identification of the entities which interact with the system and the restrictions of these interactions (A6-S1). An entity in Adelfe is an actor classified as passive or active. An active entity can act in an autonomous and dynamic way with the system. A passive entity is considered a resource of the system that can be used or modified by active entities. The classification of the entities is essential in AMAS since the agents will be part of the system treated as active entities.
Define context (A6-S2) is an activity that analyses the environment through the interaction among entities and the system by defining UML sequence and collaboration diagrams. The information flow of passive entities and the system are expressed by collaboration diagrams, while interactions among active entities and the system are described by sequence diagrams. The Adelfe methodology defines these diagrams based on the result of the previous step (A6-S1) where the entities were pre-defined with the support of the set of keywords provided in (A4).

WD_1: Preliminary Requirements	**WD_2: Final Requirements**
• A_1: Define user requirements	**A6: Characterize environment**
• A_2: Validate user requirements	• S1: Determine entities
• A_3: Define consensual requirements	• S2: Define context
• A_4: Establish keywords-set	• S3: Characterize environment
• A_5: Extract limits constraints	A7: Determine use cases
	• S1: Draw inventory of use cases
	• **S2: Identify cooperation failures**
	• S3: Elaborate sequence diagrams
	A8: Elaborate UI (user interface) prototypes
	A9: Validate UI prototypes

Table 1. WD1 and WD2– Preliminary and Final Requirements in Adelfe (2003)

Completing the environment characterization, the developer performs the Step A6-S3 describing the environment in terms of being accessible (as opposed to "inaccessible"), continuous (as opposed to "discrete"), deterministic (as opposed to "non-deterministic"), or dynamic (as opposed to "static").
Cooperative agents are a central concept in Adelfe so the developer can be able to construct AMAS. The analysis of all the unexpected and harmful events is important to realize what the causes and consequences of non-cooperative situations are for the agents. These cooperation failures are exceptions. Taking this aspect into account, the determination of the use cases is modified by adding the step (A7-S2) in which cooperation failures must be identified using specific notation.
The elaboration of user interface (UI) prototypes activity (A8) models the graphic users interface (GUI) specifications used in the interactions defined in A6 and A7. GUIs are evaluated in A9 as functional or non-functional (ergonomics, design, ...) requirements. Sometimes in this phase it is necessary to go back to activity A8 to improve UI.

2.1.2 Analysis (WD3)

Adelfe Analysis phase (Table 2) is composed by three activities: (i) AMAS adequacy verification activity (A11) to identify agents and interaction among the entities, (ii) agents identification activity (A12) to analyse the entities defined in A6 that will be considered an agent in the system and (iii) the study of the interactions between entities activity (A13) to analyse all different types of interactions between active/passive entities, between active entities and between agents (Adelfe, 2003), (Bernon et al., 2003), (Henderson-Sellers & Giorgini, 2005).

The Adelfe AMAS technology adequacy verification of the system activity (A11) is performed using the adequacy tool which considers two levels of study: global (A11-S1) and components (A11-S2). The Global analysis answers the question: “Is an AMAS technology implementation to the system necessary?” For the local level the question is "Does any component need to be implemented as AMAS?" If the tool answers the first question positively, the developer can continue applying the process. If the second answer is also affirmative, the Adelfe methodology should be applied on the components considered as AMAS since they require evolution.

The developer identifies the components of the system studying use cases and scenarios previously elaborated in the domain analysis (Adelfe, 2003), (Bernon et al., 2003), (Henderson-Sellers & Giorgini, 2005).

A10: Analyse the Domain	**A12: Identify Agents**
• S1: Identify classes	• S1: Study entities in the domain context
• S2: Study interclass relationships	• S2: Identify potentially cooperative agents
• S3:Construct preliminary class diagrams	• S3: Determine agents
A11: Verify the AMAS adequacy	**A13: Study Interactions between Entities**
• S1: Verify it at the global level	• S1: Study active/passive entities relationships
• S2: verify it at the local level.	• S2: Study active entities relationships
	• **S3: Study agents relationships**

Table 2. WD3 – Analysis in Adelfe (2003)

The cooperative agents are a central concept of AMAS system. In Adelfe the agents are cooperative entities that satisfy at least the autonomy requirements, the local objective and the interaction with other entities. After assessing all the possible agents, the classes are marked with the cooperative agent stereotype.

In Adelfe, agents are not previously known thus the developer must identify them (A12). Entities which demonstrate properties such as autonomy, local objective to pursue, interaction with other entities, partial view of its environment and the ability to negotiate are the ones to be considered as potential agents. To effectively turn into a cooperative agent, the potential cooperative agent must be prone to cooperation failures. By studying its interactions with its environments and with other entities, the developer has to determine if this entity may encounter such situations that will be considered as non-cooperative situations at the agent level. The entities meeting all these criteria will be identified as agents and the classes related to them marked as agents.

The study of the interactions between entities (A13) analyses the interactions between entities and is represented by Collaboration and Sequence Diagrams. The agents' interactions are described by AUML Protocol Diagram.

2.1.3 Design (WD4)

The Adelfe design process (Table 3) starts by analysing the different possibilities of detailed architecture of the system, creating packages sub-systems, objects, agents and the relationships among them and producing the class diagrams with the new elements (Cooperative Agent Class and the Cooperative Agent stereotype) (Adelfe, 2003), (Bernon et al., 2003), (Henderson-Sellers & Giorgini, 2005).

A14: Study detailed architecture and multi-agent model • S1: Determine packages • S2: Determine classes • S3: Use design-patterns • S4: Elaborate component and class diagrams **A15: Study interaction languages**	**A16: Design Agents** • S1: Define skills • S2: Define aptitudes • S3: Define interaction languages • S4: Define representations • S5: Define Non-cooperative situations A17: FAST Prototyping A18: Complete design diagrams • S1: Enhance design diagrams • S2: Design dynamic behaviours

Table 3. WD4 – Design in Adelfe (2003)

In the activity A15 the developer studies the interaction languages to be able to define the protocols used by agents to communicate between themselves. This information exchange between agents has to be described. For each scenario defined in the A7 and A13 activities, these exchanges are described using AUML protocol diagrams. The protocols diagrams are attached to package (not classes) because they are generic. The language definition is not necessary when the agents' communications are via the environment.

The Design Agents (A16) activity is an Adelfe methodology specific activity and allows the developer to refine the CooperativeAgent stereotyped classes identified in the A12 and A14 activities. The different modules of an agent must be defined in these activities by describing its skills, aptitudes, interaction languages, design representations, design characteristics and design non-cooperative situations.

Methods and attributes can describe the skills of an agent with a stereotyped notation <<skill>>. Skills are the system knowledge that allows the agent to perform an action. The representation of aptitudes, interaction languages, design representations and design characteristics is defined similarly to skills with a stereotyped notation. Aptitudes are the agent´s capability to reason about a specific knowledge of the system or about a real situation.

The developer analyses protocols defined in A15 activity and those assigned to an agent are associated to a state-machine. The methods and attributes link with an interaction protocol must be stereotyped <<interaction>>. The methods and attributes related to perception and action phase are represented by <<perception>> and <<action>> respectively in (A16-S3).

The step Design Non-Cooperative Situations (NCS) (A16-S6) is the most important in the design agents' activity (A16), because this is a specific ability of cooperative agents. A model guides the developer in the definitions of all situations that seem to be "harmful" for cooperative social attitude of an agent. The table lists some types of situations like ambiguity, incompetence, uselessness and conflict. The developer should fill up the conditions described for each NCS. The table contains the state of this agent when detecting the NCS, a NCS textual description, conditions permitting local detection of NCS and actions linked to this NCS.

The Fast Prototyping activity (A17) uses OpenTool (Adelfe, 2003), (Henderson-Sellers & Giorgini, 2005) to test the agents´ behaviour previously defined. The customized version of OpenTool can automatically transform a protocol diagram into a state-chart that can be run to simulate the agents' behaviour. Some methods can be implemented using a OTscript language that is a set-based action language of OpenTool.

The last activity of design is to complete the detailed architecture enriching the class diagrams (A18-S1) and developing the state chart diagrams required to design the dynamic behaviours (A18-S2). The objective is to reflect the different changes of an entity state when it is interacting with others.

2.2 MaSE methodology

The Multi-agent System Engineering (MaSE) methodology aims at supporting the designer to catch a set of initial requirements, to analyse models and implement a multi-agent system (MAS). This methodology is independent of any agent's architecture, programming language, or communication framework. The MaSE's agents are considered object specializations that instead of simple objects, with methods that can be invoked by other objects, are agents that talk among themselves and act proactively in order to reach goals (MaSE, 2010), (Deloach, 2001).

MaSE is a traditional software engineering methodology specialization with two phases (Analysis and Design) and several activities which are shown in Figure 2 (Deloach, 2001). The MaSE Analysis phase has three steps: Capturing Goals, Applying Use Cases, and Refining Roles. The Design phase has four activities: Creating Agent Classes, Constructing Conversations, Assembling Agent Classes and System Design. The highlighted items represent the resulting models of each phase.

The first step in MaSE analysis is to capture goals that express what the system is trying to achieve. These goals generally remain stable throughout the rest of the Analysis and Design phases. A decomposition of goals in a hierarchy form is the MaSE goal representation.

After the goals were defined, the functional requirements are identified and represented into use cases. Use Cases describe the behaviour of agents for each situation in MAS. In the step Applying Use Cases, situations of the initial requirements are elicited and expressed into Use Cases Diagrams and Descriptions, and UML Sequence Diagrams. The Sequence Diagrams are applied to express the sequences of roles events and they represent the desired system behaviour and its sequences of events.

The last step of Analysis phase defines a set of roles (Role Diagram) that can be used to achieve the goals of the system level. A role is an expected abstract description behaviour of each agent that aids in reaching the system goals. These roles are detailed by a series of tasks, which are described by finite-state models (Concurrent Tasks Diagrams).

The Role Diagram associates at first the goals to a role by listing them below the role name. Often, these goals are represented by numbers used in the Goal Diagram. Then the Role Diagram is detailed by associating a set of tasks for each role, representing the expected role behaviour. Communications between roles are expressed by the roles´ association and their associate tasks.

The tasks definitions are built in Concurrent Tasks Diagrams based on finite automata states. By definition, each task must be executed concurrently, while communicating with other internal or external tasks. A concurrent task is a set of states and transitions. The states represent the internal agent mechanism, while the transitions define tasks communications. Every transition has an origin and a destination state, a trigger, a guard condition and a transmission (Deloach, 2001).

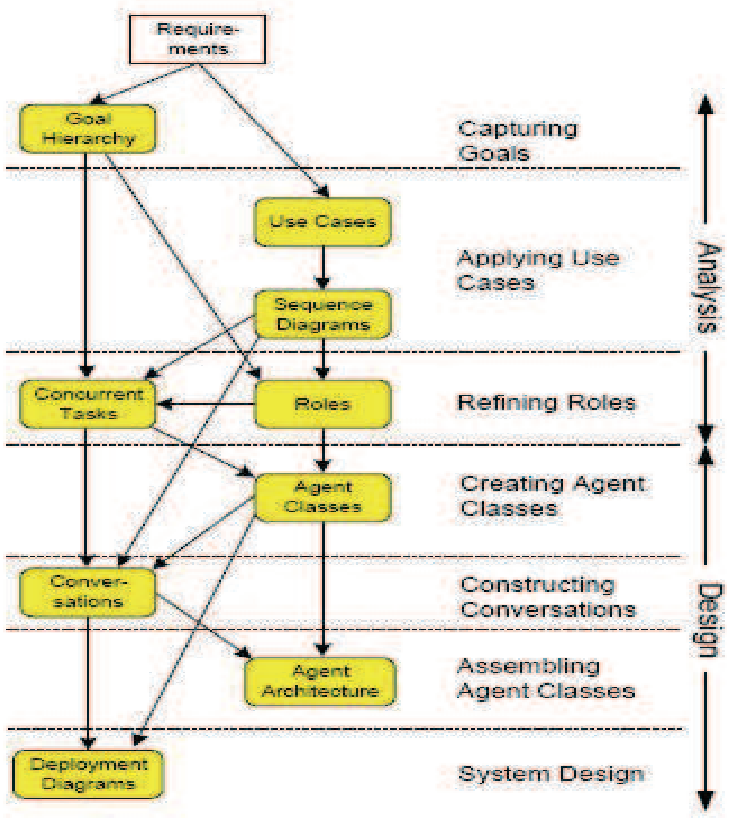


Fig. 2. MaSE Methodology Phases (Deloach, 2001)

In general, the events that are sent as broadcasts or triggers are associated with events sent to work in the same role instance, requiring an internal coordination of each task. The messages representation sent between agents uses two special events: (i) send event that represents the message sent to another agent and is denoted by send (message, agent) and (ii) receive event which defines the message received from another agent denoted by receive (message, agent).

The four diagrams proposed in the MaSE Design phase are Agent Classes, Conversations, Agent Architecture and Deployment Diagram.

The first step in the design process involves the definition of each agent class in an Agent Class Diagram. The system designer maps each role defined in the Roles Diagram to at least one Agent Class because this guarantees the goals will be implemented in the system and there is at least one agent class responsible for meeting this goal. The agent classes can be thought of as templates defined in terms of the roles they play and as the protocols they use to coordinate with other agents (O'Malley et al., 2001), (Gago, 2008).

The next step in the Design phase details the conversations between the agent classes and defines a coordination protocol between two agents. A conversation consists of two Communication Class Diagrams that represent the initiator and the responder. This diagram is a finite state automation defining the conversation states of the two agent classes using a similar syntax of the analysis phase:

rec-mess (args1) [cond] / action ^ trans-mess (args2)

This syntax defines: "if the message rec-mess is received with the arguments args1 and the condition cond holds, then the method action is called and the message trans-mess is sent with arguments args2. All elements of the transition are optional."

The third step in the Design phase is the definition of the agent architecture that is performed in two steps: (i) definition of the agent architecture and (ii) its components. The designer can choose the agent architecture, such as Belief-Desire-Intention (BDI), Reactive or Knowledge Base (Bryson & Stein, 2001).
The last activity in the Design Phase is defining the Deployment Diagram. In MaSE this diagram shows the agents´ number, types and location in the system. The diagram describes a system based on agent classes, and it is very similar to the UML Deployment Diagram. This diagram defines different agents' configurations and platforms to maximize the processing power and a network bandwidth.
MaSE can be developed using the AgenTool (2009) tool created by Air Force Institute of Technology (AFIT). AgenTool helps the system designer to create a series of models, from higher level goals definition to an automatic verification, a semi-automatic generation design and finally code generation.

3. Guardian Angel System Adelfe modelling

The Guardian Angel Project (Szolovits, 2004) was proposed as an information system centered on the patient, rather than the service provider. The software agents group explains the name "guardian angels" (GA). This "guardian angels" support functions for the patient's health, including the patient's medical considerations, legal and financial information.
Each GA is an active process which performs several important functions: (i) verification, interpretation and explanation of patient data collection, relevant facts or medical plans; (ii) recommendations with the acquired experience and patient's preferences; (iii) feasibility study, regarding the medical effectiveness, diagnostics cost and therapeutic planning; (iv) patient's health progress monitoring; (v) communications with other service providers software agents; (vi) education, information and support to the patient. All these facilities help to improve the medical diagnosis quality, increases the patient's commitment and reduces the disease effects and medical errors.
The Adelfe Guardian Angel (GA) modelling was developed using the Work Definitions for the early and final requirements, analysis and design, the AMAS Adequacy tool and OpenTool (Adelfe, 2003), (Henderson-Sellers & Giorgini, 2005). The Adelfe models presented in this section were developed by Kano (2007) and they were also improved and presented in Werneck et al. (2007).

3.1 Preliminary requirements

The following functional requirements were defined in the preliminary requirements phase: (i) allow the user to make different query to databases; (ii) allow to communicate with others sub-systems connected in the net; (iii) monitor the progress of the patient health conditions and the effect of the treatment; (iv) periodically verify the data integrity to find violations based on the user expectative and collateral effects; (v) expose the colleted data from auxiliary bases to user offering a maximal context comprehension to the user involved; (vi) customize services allowing the user objectivity, adequacy and efficiency; (vii) improve education functionalities to the user like access to encyclopaedias and universities researches to find knowledge from their diseases; (viii) provide alert and agenda functions remembering the patients their appointment, dosage and contraindications of medicines; (ix) offer to the patient the possibility to be in contact with support groups, forums and the

main medicines laboratories; (x) be able to organize of the illnesses and diseases in a hierarchal structure using decreasing levels of severity, in order to make possible to apply together different techniques to the patients.
In this phase, the following non-functional requirements were also defined: (i) to be able to store physical and logical information using an enormous data volume; (ii) to make use of visual, sonorous and touch communication capacity; (iii) the system should be available 24 hours along the 7 days of the week, 365 days per year; (iv) be multi-task and allow to answer to several data request simultaneous at a certain average time; (v) to be conceptually distributed (the small parts inside inhabit all the same environment, however they represent, separately, concepts and well distinct parts); (vi) to allow the sudden appearance and the abrupt disappearance of its components; (vii) to allow the adaptation and evolution of its components.
The key words defined in this phase are: Monitoring, GA, Patient, Communication, Health Professional, Insuring, and History Information.
One of the GA constraints relates to maintenance routines when the system will not be available. Another restriction is the subnets functionality with which the system interacts. The case of eventual problems in one of these subnets the user will be unable to access them until they become again in operation.

3.2 Final requirements

The environment characterization activity (A6) identified the following passive entities: World Wide Web, Library, Hospital Stay, Illness Organism Information, Idiopathic Cause and Therapy. The active entities list are: Patient, Family, Support the Patient Group, Government, Health Plan Insurance, Laboratory, Health Professional, Hospital, Clinic, Pharmaceutical Industry, Ambient Factors and the proper Guardian Angel.
The central entity of the Guardian Angel is the Patient that has the ability to activate any events in any circumstance that will be convenient, dynamically interacting with the system. The Family is another entity that can modify the patient treatment routine depending on the treatment results and satisfaction degree, being able to dynamically interact with the system. The Health Professional entity has the power to trace treatment plans, to request examinations and to prescribe medicines, dynamically interacting with the system.
The Guardian Angel can be seen as "processing cells" of the system that interact dynamically in accordance with the recurrently perceptions of the environment. This entity was divided in 4 specializations: (i) Analyser - GA directed towards the tasks which require analyses, interpretation and understanding of data in one determined context; (ii) Inspector - GA directed towards the monitoring/inspection of specific states in the system; (iii) Diplomat - GA directed towards the reduction and treatment of Non-Cooperative Situations. The GA Diplomat is responsible for using its "diplomacy" together with a GA Analyser that helps to determine the priorities of the GAs´ execution, and (iv) Worker - the GA worker is the basic processing cell with the physical operations required to modify data/state of the system.
The Collaboration Diagrams for passive entities and the Sequence Diagrams for the active entities were built (Kano, 2007) and figure 3 presents an example of Customize Setting to Adapt Treatment to Patient's Reality.
The Guardian Angel system activity of characterizing environment (A6-S3) was classified as: (i) inaccessible because several users can be logged and they can modify data at anytime; (ii) continuous because the users are free to make their own actions; (iii) non-deterministic

because the prescription of a treatment can be different for the same disease in different patients, and (iv) dynamic because the system depends on the environment and that can not be predicted by the system.

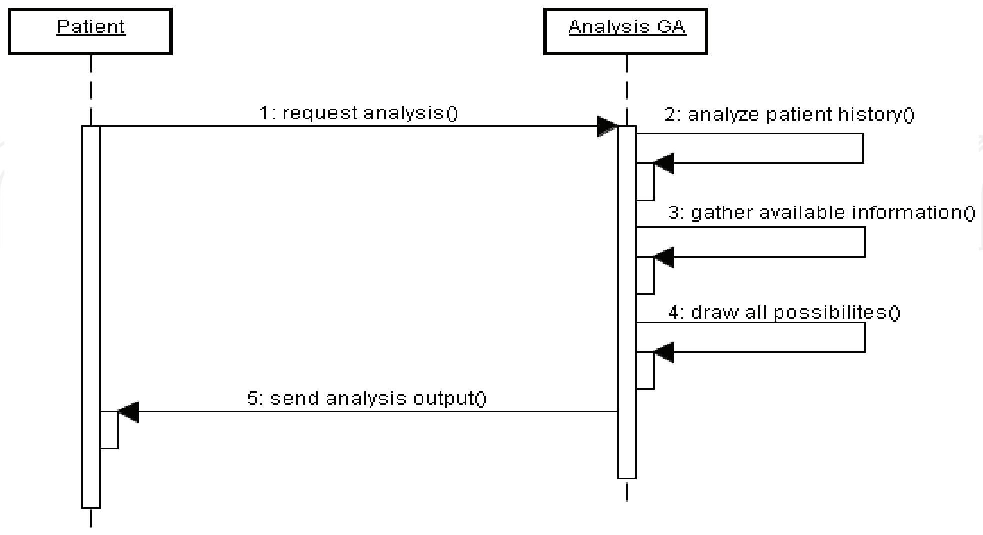

Patient
Analysis GA
1: request analysis()
2: analyze patient history()
3: gather available information()
4: draw all possibilites()
5: send analysis output()

Fig. 3. Sequence Diagram: Customize Setting to Adapt Treatment to Patient's Reality (Kano, 2007)

Then the use case diagrams were defined and divided in five groups: GA Domain, Patient, Institutions, Administrative and Service. For each group a Use Case Diagram was modeled involving several use cases and then for each Diagram some NCS were identified as shown in Figure 4.

3.3 Analysis

In the GA Domain Analysis four new passive entities (Idiopathic Cause, Therapy, Hospital Stay and Disease-Causing Organism) were found and some diagrams and documents developed during previous steps had to be modified.
The classes identified in this phase were: User, People, Patient, Family, Health Care Professional, Doctor, Guardian Angel (Analyser, Diplomat, Inspector and Worker), Data Source, Clinic, Insurer, World Wide Web, Library, Government, Laboratory, Pharmacy Industry, Hospital, Patient Support Group, Environmental Factor, Idiopathic Cause, Therapy and Hospital Stay.
In the AMAS technology adequacy activity, the GA got the following reply from the tool in relation to the global criterion; "Your application possesses, with a high degree, almost all the characteristics that can justify - without any ambiguity- using AMAS". In the components evaluation the tool reply was: "Even if your application needs using AMAS some of its components must also be designed using this technology. We recommend you to apply as many times as necessary the methodology to specify all those components".
The agents identify activity (A12) studied active entities and for each one a form was defined as shown in Table 4. Thus four cooperative agents have been identified.

3.4 Design

The Design phase defined the packages and classes by elaborating the classes and collaboration diagrams. No design pattern was applied and the activity A17 of Fast prototype was not realized because the JAVA version of the tool does not work in the project computer because of some incompatibility that we could not fix.

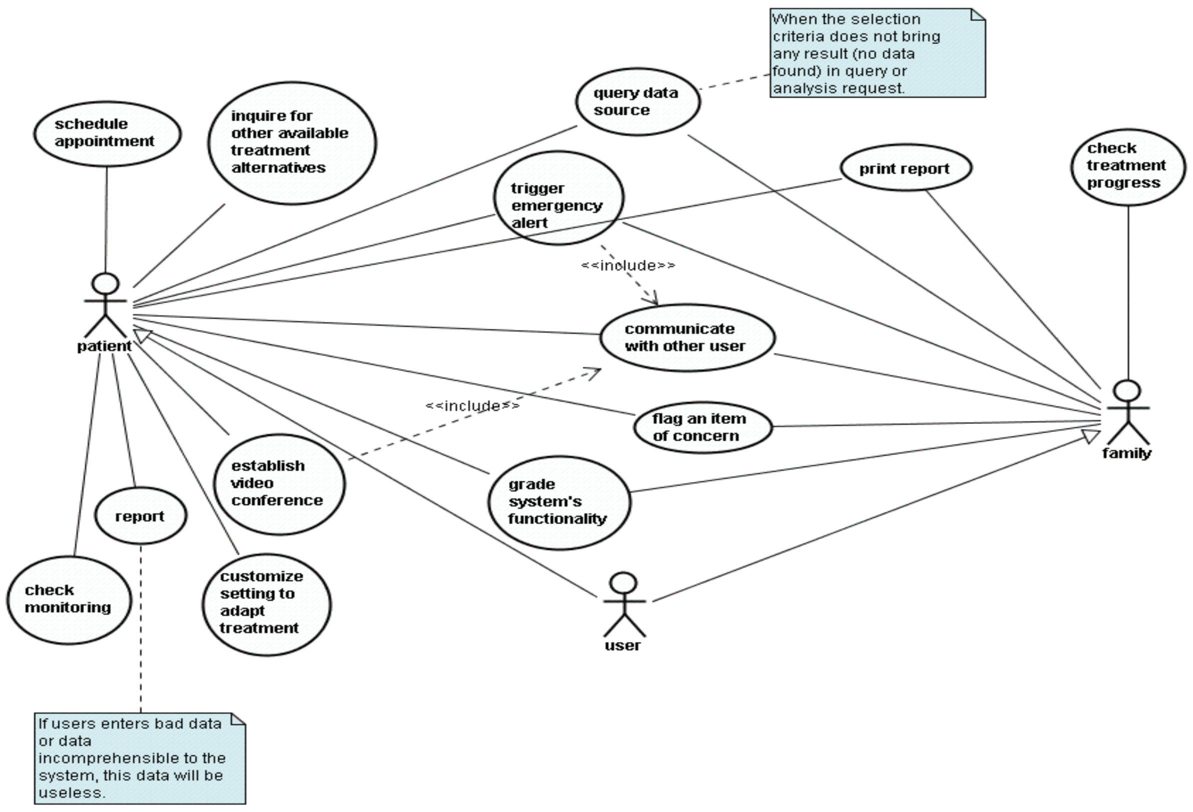

When the selection criteria does not bring any result (no data found) in query or analysis request.
query data source
schedule appointment
inquire for other available treatment alternatives
trigger emergency alert
print report
check treatment progress
<<include>>
patient
communicate with other user
<<include>>
flag an item of concern
family
establish video conference
grade system's functionality
report
check monitoring
customize setting to adapt treatment
user
If users enters bad data or data incomprehensible to the system, this data will be useless.

Fig. 4. Non-Cooperation Situations: User (patient) (Kano, 2007)

Guardian Angel	
Autonomy:	Has autonomy because can make decisions based only on its knowledge
Local Goal:	The local goal is to perform a task that was assigned to it.
Interactions with other Entities:	Interact with other Guardian Angels and Patient.
Environment Partial Overview:	Limited overview of the system
Negotiation Abilities:	Capable to Negotiate with other entities.
Potential agent:	An agent in potential according to Adelfe´s definition.
Dynamic environment:	Yes – it is not possible to prevent in which circumstances its actions are taken.
Face NCS	Yes - can request a service that is not available
Treat NCS	Yes- For example when a GA does not receive an answer to a feedback request.

Table 4. WD4 – Design in Adelfe (Werneck et al., 2007)

In the activity A15 the interactions between the agents were studied and for each an AUML Protocol Diagram was defined (an example is shown in figure 5). For each Guardian Angel the abilities, aptitudes, representations and characteristics were identified and also defined the protocols used in A15 activity which will be used by the agents. Finally the NCS in a form (Table 5) were defined.

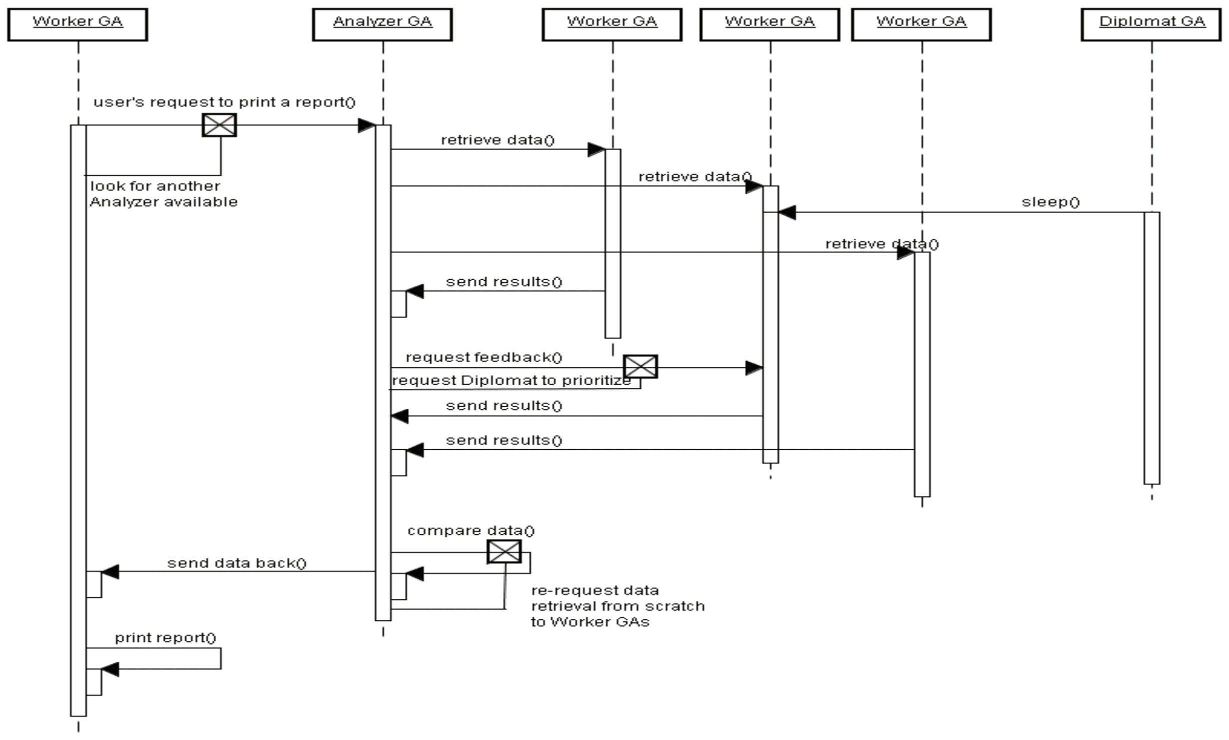


Fig. 5. Protocol Diagram of GA (Kano, 2007)

Name	Permission denied
State	Execute the activity
Description	An agent faces this situation when the activity that it intends to execute cannot be accomplished with the permissions of the user in question
Conditions	User with no knowledge about the system.
Actions	The agent must supply to the user a list of all the users who have connection with this and that they have permission to execute the task.

Table 5. The Identification of NCS Form (Kano, 2007)

The diagrams in the last activity (A18) were detailed and the dynamic behaviours were also completed by designing the State Chart Diagram where the attributes and methods were specified to express the agents' state, conditions and actions.

4. Educ-MAS MaSE modelling

The Educ-MAS (Educational Multi-Agent System) is a learning education environment with multi-agents that aims at helping the teaching process on a specific topic. The modelling presented in this chapter was improved from Gago (2008) and Gago et al. (2009).

The first step in the MaSE analysis requirements modelling is to define the Goal Hierarchy Diagram. Then the goals are decomposed in sub-goals until they can be expressed as functions as shown in Figure 6. The main goal Promote Individual Learning was structured based on the Intelligent Tutoring Systems classical architecture that considers four models: Pedagogic, Expert, Student and Interface. Each model reflects the ability and the characteristics of the Educational System (Viccari et al., 2003), (Wooldridge & Jennings, 1997): Explore Student, Plan Course, Manage Knowledge and Manage Teaching. The goals are also decomposed into other goals. For example the goal Plan Course was partitioned into two sub-goals (Consult Defined Goals and Define Course Plan) and the sub-goal Define Course Plan has two sub-goals named Define Content of the Modules and Define the Plan Presentation of the Module.

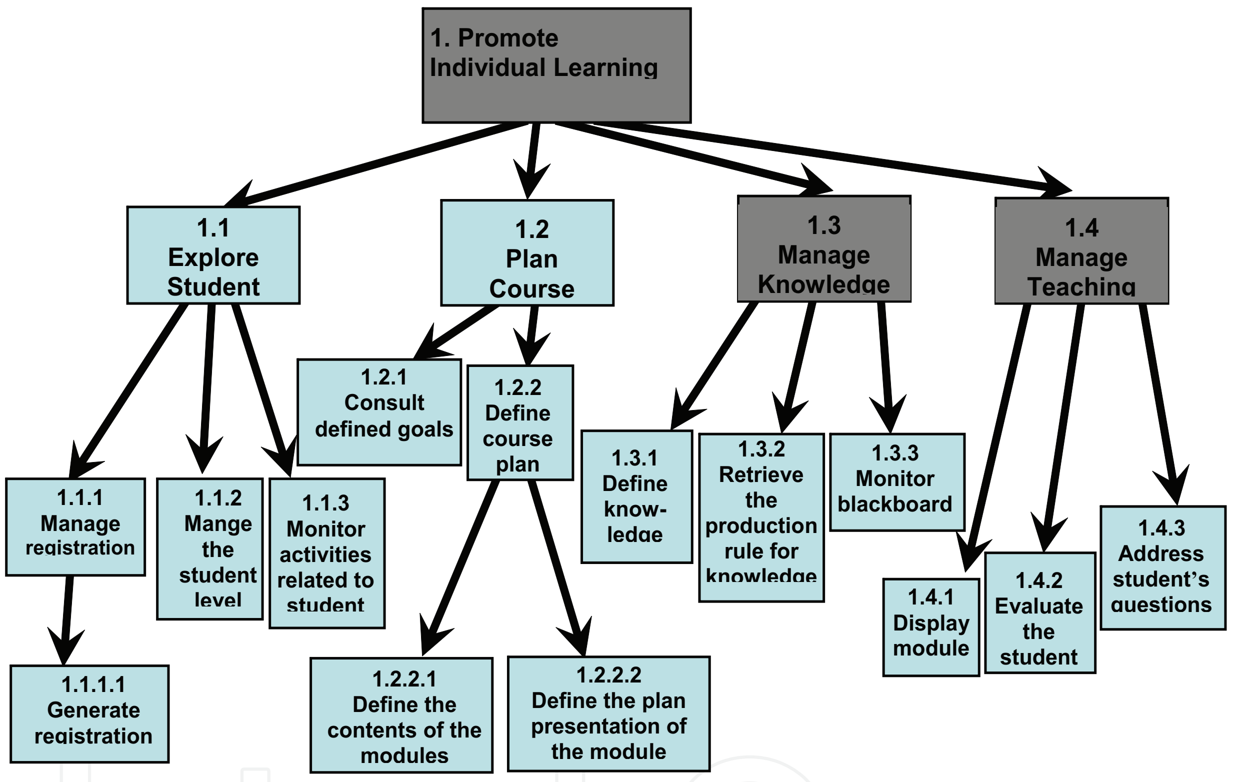
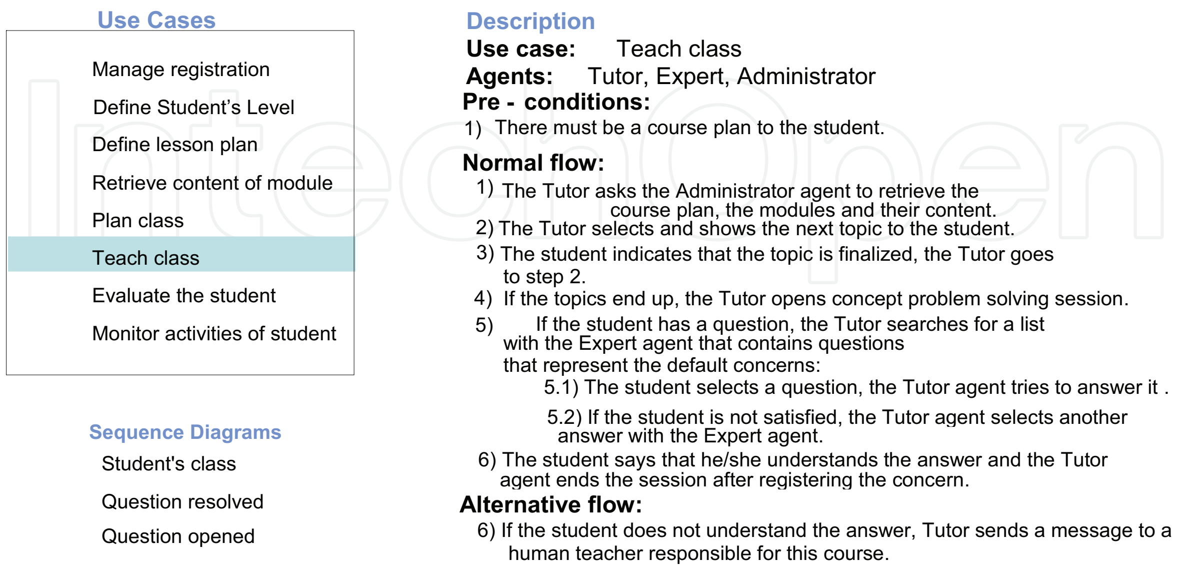

1. Promote Individual Learning
1.1 Explore Student
1.2 Plan Course
1.3 Manage Knowledge
1.4 Manage Teaching
1.1.1 Manage registration
1.1.2 Mange the student level
1.1.3 Monitor activities related to student
1.1.1.1 Generate registration
1.2.1 Consult defined goals
1.2.2 Define course plan
1.2.2.1 Define the contents of the modules
1.2.2.2 Define the plan presentation of the module
1.3.1 Define know-ledge
1.3.2 Retrieve the production rule for knowledge
1.3.3 Monitor blackboard
1.4.1 Display module
1.4.2 Evaluate the student
1.4.3 Address student's questions

Fig. 6. Educ-MAS Goal Diagram adapted from Gago et al (2009)

Then the goals and sub-goals were translated into use cases. Figure 7 presents an example of Educ-MAS use case and the respective sequence diagram for the functional requirement Teach Class, its description and also the name of Sequence Diagrams that retracts the scenarios of this use case. The scenarios are Student's Class, Questions Resolved and Questions Not Resolved. For each one a Sequence Diagram has to be built showing how the system behaves. Figure 8 shows the agent behaviour with the third scenario of the case Teach Class The whole specification of Educ-MAS can be found in Gago (2008).

The next activity is to develop a set of roles and tasks showing how the goals are reached based on the Goals, the Use Cases (diagrams and descriptions) and the Sequence Diagrams. Figure 9 represents the Preliminary Role Diagram where the goals were mapped to system roles. For example, the System Administrator role (Fig.9) achieves the goals Explore Student (goal 1.1 in the Goal Diagram), Manage Registration (goal 1.1.1 in the Goal Diagram), Generate Registration (goal 1.1.1.1 in the Goal Diagram), and Monitor (goal 1.1.3 in the Goal Diagram), activities related to student.


Use Cases

Manage registration
Define Student's Level
Define lesson plan
Retrieve content of module
Plan class
Teach class
Evaluate the student
Monitor activities of student

Sequence Diagrams

Student's class
Question resolved
Question opened

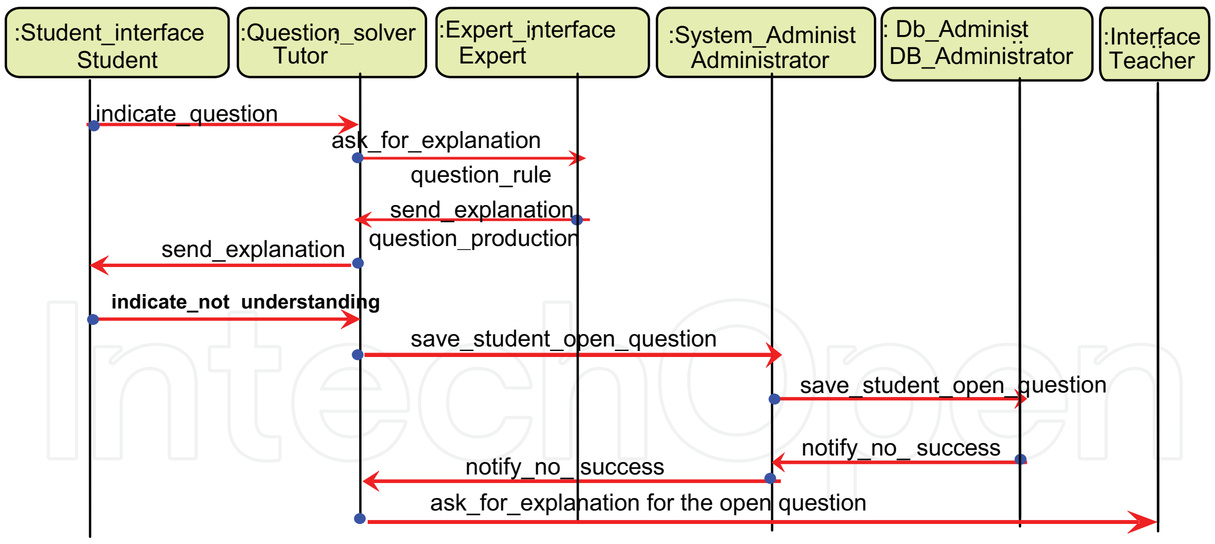
Description

Use case: Teach class
Agents: Tutor, Expert, Administrator
Pre - conditions:
1) There must be a course plan to the student.

Normal flow:
1) The Tutor asks the Administrator agent to retrieve the course plan, the modules and their content.
2) The Tutor selects and shows the next topic to the student.
3) The student indicates that the topic is finalized, the Tutor goes to step 2.
4) If the topics end up, the Tutor opens concept problem solving session.
5) If the student has a question, the Tutor searches for a list with the Expert agent that contains questions that represent the default concerns:
5.1) The student selects a question, the Tutor agent tries to answer it .
5.2) If the student is not satisfied, the Tutor agent selects another answer with the Expert agent.
6) The student says that he/she understands the answer and the Tutor agent ends the session after registering the concern.

Alternative flow:
6) If the student does not understand the answer, Tutor sends a message to a human teacher responsible for this course.

Fig. 7. Use Case Teach Class adapted from Gago et al (2009)



:Student_interface Student
:Question_solver Tutor
:Expert_interface Expert
:System_Administ Administrator
: Db_Administ DB_Administrator
:Interface Teacher
indicate_question
ask_for_explanation
question_rule
send_explanation
question_production
send_explanation
indicate_not understanding
save_student_open_question
save_student_open_question
notify_no_ success
notify_no_success
ask_for_explanation for the open question

Fig. 8. Question Opened Sequence Diagram adapted from Gago et al (2009)

In the complete Role Diagram the tasks responsible for the roles and the associations among themselves were introduced to reach the responsible goal roles. Continuing the Analysis phase the Concurrence Task diagrams have to be built for each task as shown in Figure 10 for the task Monitor Blackboard. This task is associated to the Expert interface role which is responsible for the goal with the same name. This task monitors the blackboard in order to interface the knowledge base introducing the questions and their contents.

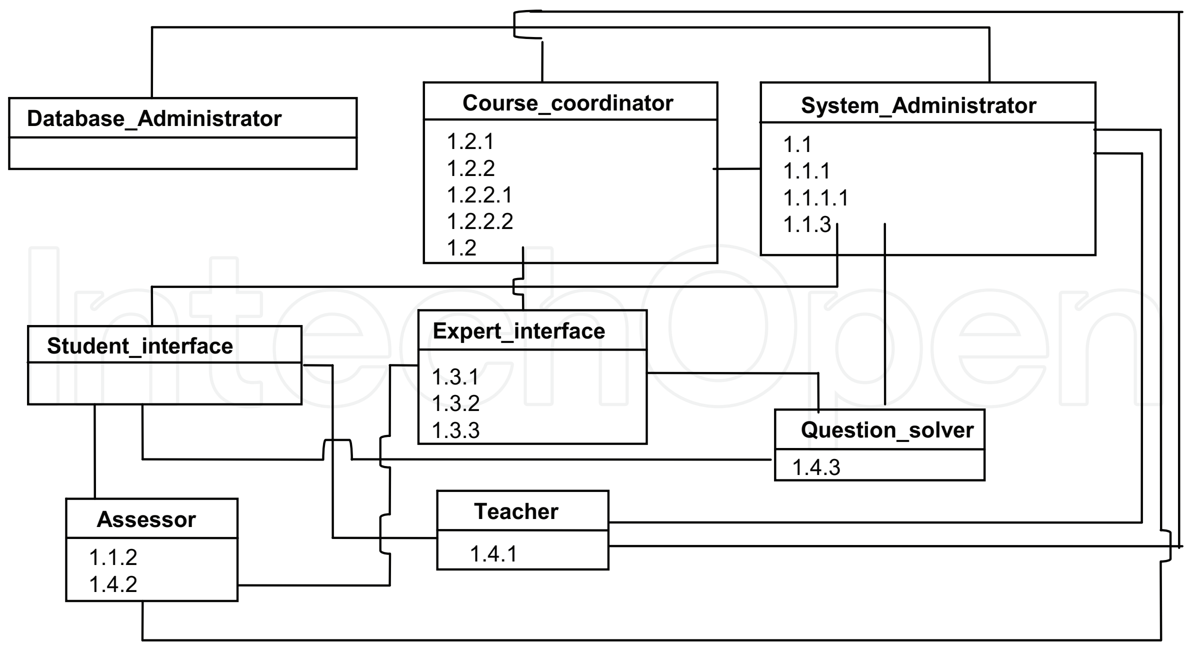


Fig. 9. The Educ-MAS Partial Role Diagram (Gago et al., 2009)

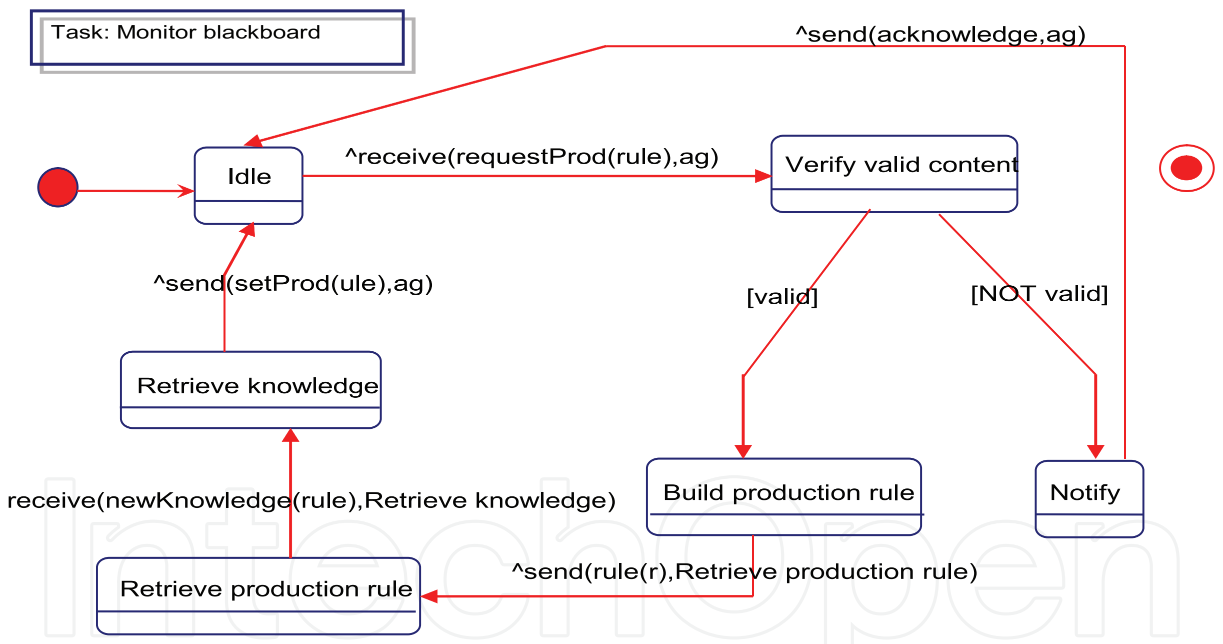


Fig. 10. Concurrent Task Diagram for the task Monitor blackboard (Gago et al., 2009)

In the Design phase the Role Diagram and the Concurrence Task diagrams have to be used to design the individual components of the agent classes as presented in Figure 11. The agent architecture chosen was a simple BDI (Belief-Desire-Intention) agent architecture and Figure 12 presents an example of a Tutor agent class partial structure components. The last step in the Design phase has to develop an overall operational design by designing the Deployment Diagram (Gago et al., 2009). The Tutor, Administrator, Coordinator and Expert Agents are defined in an environment as a system. The Interface (Student Model) starts at the student's computer while the Database Management and the other part of the system are in network computers.

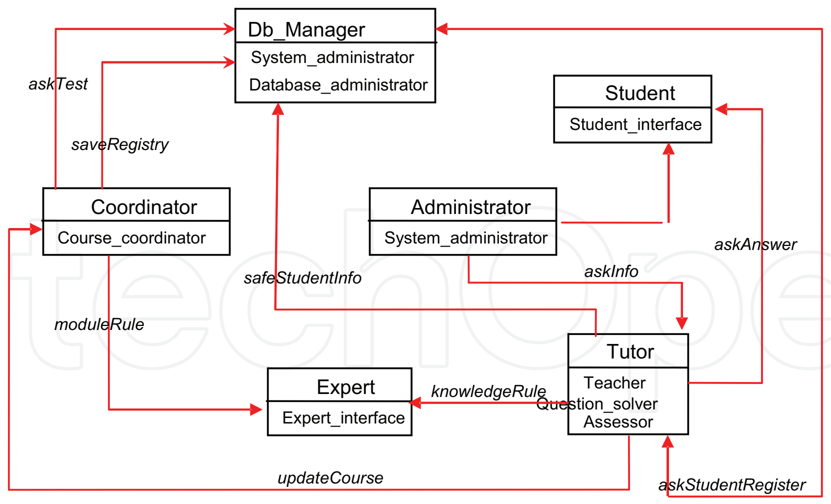


Fig. 11. The Educ-MAS Agent Class Diagram (Gago et al., 2009)

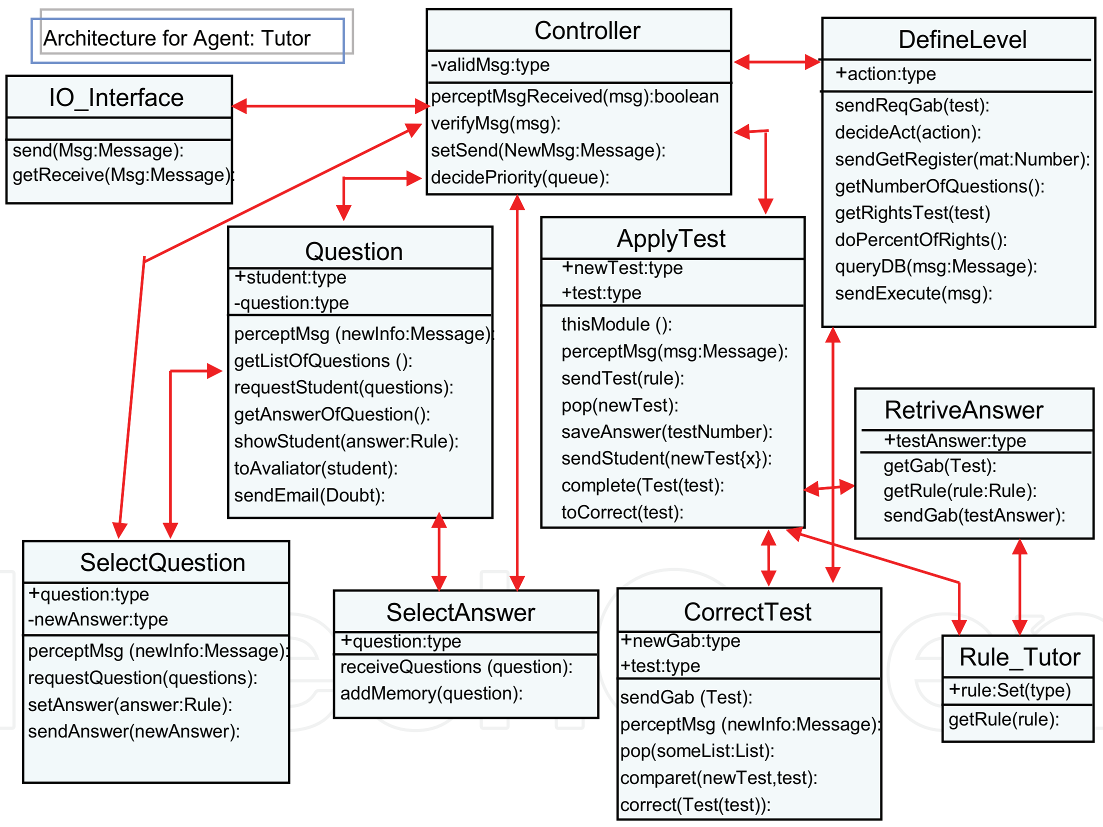


Fig. 12. Tutor Agent Class Partial Structure Component (Gago et al., 2009)

5. Conclusions

This work is part of a broader project which aims at analysing important aspects of modelling and developing different Multi-Agent Systems using several methodologies. The first system modelled presented was a classical Multi-Agent System case study in a Medical Domain using Adelfe methodology.

Adelfe is a methodology originated from object orientation based on UML (Rumbaugh et al., 2004) that incorporated AUML (2007) protocols diagram and the development process RUP (Krutchen, 2000). The Adelfe process covers the requirements, analysis and project phases with a well defined process. Adelfe can be a powerful methodology in terms of cooperative agents' concepts centred in Non-Cooperative Situations. This method allows the definition of important agent concepts as autonomy, proactivity and autonomy reason. However, the methodology needs to improve some aspects of characterized environment by adding new diagrams that can model goals, plan and organization.

The second Multi-Agent System modelled was a learning education environment in the MaSE that is an object-oriented methodology that supports analysis and design phases using agent-orientated techniques. MaSE can also be considered a powerful methodology in terms of cooperative agents' concepts (definition of autonomy, proactivity and autonomy reason and the agent concepts are centred in the Roles Diagram and in Goal orientation. However, this methodology is not completely defined, especially for the Early Requirements phase it lacks on capturing, understanding and registering terminology. In DiLeo et al. (2002) they propose to integrate ontology representation to MaSE that can solve this weakness. Another point to be improved is related to non-functional requirements that are not mentioned in the methodology and the MaSE protocols representation that is divided into two diagrams so both diagrams have to be seen to understand the agent communications.

In the future we are going to compile these experiences from all MAS development and define a knowledge base for an Agent-Oriented Methodology Approach based on the Situation Method Engineering (SME). This knowledge will provide a flexible way of developing a Multi-Agent System using a methodology based on strengths and examples of models of each method fragments and also situations when applying a respective method artefact.

6. Reference

Adelfe (2003). Atelier de Développement de Logiciels à Fonctionnalité Emergente) at *http://www.irit.fr/ADELFE*.

Agent Tool (2009). at *http://macr.cis.ksu.edu/projects/agentTool* , Last Updated October 2009.

AUML (2007). at *http://www.auml.org/auml/* Last updated on 17 June 2007.

Bernon, C., Camps, V., Gleizes, M. P. and Piscard, G. (2003). ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering; In: *Lecture Notes in Computer Science*, Volume 2577, Springer Berlin Heidelberg, ISSN: 0302-9743, pp. 156-169.

Bryson, J. & Stein, L. (2001). Modularity and design in reactive intelligence, In: *International Joint Conference on Artificial Intelligence*, IJCAI-2001, Seattle (USA), pp 1115-1120.

Coppieters, A. M., Marzulo, L.A.J., Kinder, E. And Werneck, V.M. (2005). Modelagem Orientada a Agentes utilizando MESSAGE, In : *Cadernos do IME-Série Informática*, Rio de Janeiro, v.18, ISSN 1413-9014, pp. 38-46 in portuguese.

Cysneiros, L. M., Werneck, V. M. B., Amaral, J., Yu, E. (2005). Agent/Goal Orientation versus Object Orientation for Requirements Engineering: A Practical Evaluation Using an Exemplar, In: *Proc. of VIII Workshop in Requirements Engineering*, Porto, ISBN 972-752-079-0, pp.123-134.

Cysneiros, L. M., Werneck, V. M. B.; Yu, Eric (2005a). Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar. In: *Journal of Computer Science & Technology*, ISSN: 1000-9000, USA, v. 5, n. 2, pp. 71-79.

Dantas, T. C.; Soares, G. E. ; Costa, Rosa Maria Esteves Moreira Da Werneck, Vera M. B. ; Castro, M. C. S. (2007). AprendEAD: Ambiente para Educação à Distância Apoiado em Agentes; In: *Cadernos do IME. Série Informática*, v. 23, p. 16-23, ISSN 1413-9014, in portuguese.

Deloach, Scott A. (2001). Analysis and Design using MaSE and agentTool. In:*12th Midwest Artificial Intelligence and Cognitive Science Conference* (MAICS 2001), Miami University, Oxford, Ohio.

Dileo, Jonathan; Jacobs,Timothy; Deloach, Scott. (2002). Integrating Ontologies into Multiagent Systems Engineering. In*: Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002*).

Gago, I. S. B. ; Werneck, Vera M. B. & Costa, Rosa Maria Esteves Moreira da . (2009). Modelling an Educational Multi-Agent System in MaSE. In : *Lecture Notes in Computer Science*, v. 5820, ISSN: 0302-9743, p. 335-346.

Gago, I., Utilização da metodologia MaSE na modelagem de Sistema Tutor Inteligente Dissertation on Informatics Technology, UnderGraduation, UERJ, Rio de Janeiro, 2008 pp 114 in portuguese.

Henderson-Sellers, Brian & Giorgini, Paolo (ed). (2005). *Agent-oriented Methodologies*. 1ed: Idea Group Inc, London, UK, ISBN 1-59140-581-5, p412.

Henderson-Sellers, Brian & Ralyté, Jolita. (2010). Situational Method Engineering: State-of-the-Art Review, In: *Journal of Universal Computer Science*, vol. 16, no. 3, DOI: 10.3217/jucs-016-01, 424-478.

Iglesias, C.A. & González, J.C. (1998). A Survey of Agent-Oriented Methodologies In *Proceedings of the 5th International Workshop on Agent Theories, Architectures and Languages (ATAL'98),* LNAI n1555, Springer Verlag, Paris, France, 317-330.

Kano, Abrahão Yehoshua Kano, Modelagem orientada a agentes do Sistema Guardian Angel: Sistema de Informação de Saúde centrado no Paciente. *Dissertation on Informatics Technology UnderGraduation*, UERJ, Rio de Janeiro, 2007 pp 247 in portuguese.

Krutchen, P. (2000) *The Rational Unified Process: An Introduction*, Reading, MA, Addison Wesley.

MaSE (2010} at *http://macr.cis.ksu.edu/projects/mase.htm* access at August, 2010

O'Malley , Scott A.; Deloach, Scott A. (2001). Determining When to Use an Agent-Oriented Software Engineering Paradigm. In : *Proceedings of the Second International Workshop On Agent-Oriented Software Engineering (AOSE-2001*), Montreal, Canada.

Odell, J., Parunak, H., Bauer, B. (2001). Representing agent interaction protocols in UML, In: *First International Workshop on Agent-Oriented Software Engineering*, (AOSE 2000), Ciancarini, P., Wooldridge, M., Eds., LNCS 1957 Springer, Limerick, Ireland, 121-140.

Rumbaugh, J., Jacobson, I. & Booch, G. (2004). The Unified Modelling Language Reference Manual, Second edition, Addison,-Wesley.

Schreiber, G., Akkermans, H., Anjewierden, A. Hoog, R. de, Shadbolt, N., Velde, W.Van de, Wielinga, B. (1999). *Knowledge Engineering and Management: The CommonKADS Methodology*, Cambridge, MA, ISBN: 0262193009 .

Sousa, R.; Araújo, Alex L. de, Gomes Junior, Carlos A.; Costa, Rosa Maria Moreira da, Werneck, V. M. B. (2009). Modelagem de Requisitos Orientada a Agentes utilizando MaSE; Cadernos do IME. Série Informática, v. 28, ISSN 1413-9014, in portuguese.

Sousa, R.; da Cunha, A. L. F.; Martins, R. F. A. Cysneiros, L.M., Werneck, V. M. B. (2010). Evaluating MaSE Methodology in the Requirements Identification; In : *Proceedings of Proceedings of the 33nd Annual IEEE Software Engineering Workshop*.: IEEE Computer Society Press, Skovde.

Szolovits, P., Doyle, J., Long, W.J., Kohane, I. e Pauker, S. G. (2004). *Guardian Angel: Patient-Centered Health Information Systems*, Technical Report MIT/LCS/TR-604, at http://groups.csail.mit.edu/medg/projects/ga/manifesto/GAtr.html

Tavares, D. G.; Eichler, J.; Pereira, L. F.; Silva, T. S.; Da Cunha, A. L. F.; Martins, R. F. A. Cysneiros, L.M., Werneck, V. M. B. (2009). Processo de Desenvolvimento do Sistema Multi-Agentes Monitor Glicêmico; Cadernos do IME. Série Informática, v. 28, ISSN 1413-9014, in portuguese.

Viccari, Rosa M., Floresa, Cecilia D., Silvestrea, Andre´ M., Seixasb, Louise J., Ladeirac, Marcelo, Coelho, Helder (2003). A multi-agent intelligent environment for medical knowledge, In: *Artificial Intelligence in Medicine,* 27, ISSN: 0933-3657, 335–366.

Werneck, V. M.; Pereira, L. F.; Silva, T. S.; Almentero, E. K.; Cysneiros, L. M. (2006). Uma Avaliação da Metodologia MAS-CommonKADS, In: *Proceedings of the Second Workshop on Software Engineering for Agent-oriented Systems*, (SEAS´06), Florianópolis, Brazil, ISBN: 0164-1212; 13-24, in portuguese.

Werneck, Vera M. B. ; Cysneiros, Luiz Marcio ; Kano, A. Y. (2007) Evaluating Adelfe Methodology in the Requirements Identification. In: *Proceedings of 10TH Workshop on Requirements Engineering*. Toronto , York University Printing Services, v. 1. 13-24.

Werneck, Vera M. B. ; Cysneiros, Luiz Marcio ; Kano, A. Y. ; Coppieters, A. M. ; Fasano, A. L. ; Marzulo, L. A. J. ; Furtado, L. O. ; Pereira, L. F. ; Lopez, M. A. C. ; Pereira, Ricardo Augusto Gralhoz, Rafael Barros; Silva, T. S. ; Santos, T. R. M. (2008). Metodologias Orientadas a Agentes. Cadernos do IME. *Série Informática*, v. 26, , ISSN 1413-9014. 7-16, in Portuguese.

Wooldridge, M. & Jennings, N. (1997). *Intelligent Agents: Theory and Practice*, at http://www.doc.mmu.ac.uk/STAFF/mike/ker95/ker95-html.html.

Yu, E. & Cysneiros, L.M. (2002). Agent-Oriented Methodologies-Towards a Challenge Exemplar, in *Proc of the 4th Intl. Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002)*, Toronto, pp.47-63.