Evaluating ADELFE Methodology in the Requirements Identification

V.M. B. Werneck[1]; A. Y. Kano[1], L. M. Cysneiros[2]
[1] *Instituto de Matemática e Estatística*
UERJ - Universidade do Estado do Rio de Janeiro - Brasil
vera@ime.uerj.br,
[2] *School of Information Technology*
York University – Toronto – Canada
cysneiro@yorku.ca

Abstract

The increasing use of multi-agent systems brings challenges that have not been studied yet, such as how should we adapt requirements elicitation to cope with agent properties like autonomy, sociability and pro-activeness. Many methodologies were proposed adopting this new paradigm. However, most of them are still in their early phases and therefore need to be adapted. In this work ADELFE, an agent-oriented methodology is evaluated. We use an exemplar proposed in 2001 by Yu and Cysneiros [1] to evaluate both agent/goal orientation and object orientation. This evaluation aims at analysing the strengths and weaknesses of ADELFE through the methodologies questions proposed in the exemplar.

1. Introduction

The agent-oriented modeling is proposed as a suitable requirement engineering approach for complex organizational application domains which have to deals with the proliferation of software components distributed by a great number of users in a global scale. Moreover, this new paradigm comes up as an approach to deal with the need for new applications such as autonomy and sociability. 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 should this condition be, has to be analyzed since the early stages of the software development process.

This research work is included in a project for the evaluation of agent-oriented methodologies [2], [3], [4], [5], based on the evaluation framework proposed by Yu e Cysneiros [1]. Our aim in this paper is to present an analysis of ADELFE methodology using the experience in the definition and design of the requirements of the Guardian Angel System [6]. The experiment allowed us to evaluate the ADELFE methodology, considering its potential in identifying requirements.

This paper is organized in five sections and aims to present the experience on the agent-oriented methodologies evaluation framework based in an exemplar described in the section 2. An overview of ADELFE methodology is presented in the section 3 and discusses the difference between this methodology and the object oriented approach of UML/RUP. Section 4 describes the modeling of Guardian Angel System in ADELFE focusing in the mains aspects of agent oriented. The ADELFE methodology evaluation is discussed in section 5, comparing it with related works. The section 6 presents the conclusion and the future works.

2. Evaluation Framework

The evaluation framework [1] used in this research is based on the Guardian Angel (GA) project [6], which gives automatic support for chronic disease patients suffering for example from insulin-dependent diabetes, hypertension, or undergoing anticoagulation therapy. This system covers all the aspects from the health system including prescriptions, gathering a comprehensive life-long record of the individual's health-related information. The GA project proposes developing a group of agents representing the hospital (GA hospital), home members (GA HOME) and the patient (GA PDA). The system stores, handles and interprets personal medical history as well as daily treatment routine of the patient, suggesting better options based on his/her preferences. It should be accessible all the time and for all the health care institutions.

The Guardian Angel project [6] was chosen as a base for the exemplar, because of its complexity and its requirements for all the current systems. By having such rich and complex example we expect to be able to deeply evaluate each methodology on complex properties such as distribution, privacy autonomy, pro-activity or sociability problems that could be otherwise not properly judged. Since the GA is an easily comprehended and open system, it is optimum to analyze methodologies, identifying its strengths and weaknesses.

The exemplar is expressed in terms of a set of numbered scenarios (EA0.0 until EA9.0) such as the one below: EA2.0- Once the patient and the doctor establish a treatment plan GA-PDA must help the patient to keep and monitor the routine.

The process also provides together with the scenarios, a set of evaluation questions aimed to help evaluating how well the methodology supported modeling the set of scenarios. In this work, we used these methodological questions presented in the exemplar to evaluate ADELFE methodology that will be better explained in section 5.

3. The ADELFE Methodology

The ADELFE methodology [7], [8], [9] was developed to work on some aspects not already considered by existing methodologies and to consolidate the AMAS (Adaptative Multi-Agent Systems) Theory. ADELFE is an acronym that translated from French means "framework to develop software with emergent functionality"

ADELFE provides a specific process adapted from an interpretation of RUP (Rational Unified Process) [10]. Some additions have been made to consider AMAS Theory specificities, for example the environment characterization of the system the identification of cooperation failures. Each user and service provider has an individual objective which is to execute for a required request and does not comprehend the whole system functionality.

In ADELFE, the developer is not obliged to be specialized in the AMAS system field. Some additional notations are provided together with some tools to guide the developer throughout the process application. ADELFE provides the analyst a tool to estimate the AMAS technology adequacy. 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.

ADELFE also uses AUML principle [11] together with UML [12] and RUP [10] to express agent interaction protocols. Two other tools (Open Tool and Interactive Tool) are integrated to the framework. The Open tool is a graphic modeling tool which gives support to UML notation and ADELFE notation which introduces new stereotypes and protocols of AUML interaction. The Interactive Tool describes the modeling process and helps guiding the developing and its application.

The main objective of the ADELFE method is to cover 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 is applied without modifications.

3.1 Preliminary and Final Requirements (WD1 e WD2)

ADELFE [7], [8], [9] does not add anything to the preliminary requirements work definition (WD1) as described by the RUP (Table 1). The aim still consists of studying the customer's needs to produce a document on which both the customer and the developer agree.

The characterization of the environment (A6) is the first activity of the process of the final requirements (WD2), aiming at defining the environment. This activity was added to the RUP because the environment is very important concept in AMAS theory thus, the environment has to be studied and comprehended before the use cases definition. 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 is an actor just as in UML, but it may be classified in ADELFE as passive or active. An active entity can act in an autonomous way and dynamically with the system. A passive entity is considered as 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.

The context definition (A6-S2) analyses the environment through the interaction among entities and the system defining UML sequence and collaboration diagrams. The information flow of passive entities and the system are expressed by collaboration diagrams, while interaction 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 (A4).

Table 1. WD1 & WD2 – Preliminary and Final Requirements in ADELFE

WD$_1$: Preliminary Requirements	**WD$_2$: Final Requirements**
▪ A$_1$: Define user requirements ▪ A$_2$: Validate user requirements ▪ A$_3$: Define consensual requirements ▪ A$_4$: Establish keywords-set ▪ A$_5$: Extract limits constraints	**A6: Characterize environment** ➢ S1: Determine entities ➢ S2: Define context ➢ S3: Characterize environment 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

In the Step A6-S3, the developer has to describe the environment in terms developed of accessible (as opposed to "inaccessible"), continuous (as opposed to "discrete"), deterministic (as opposed to "non-deterministic"), or dynamic (as opposed to "static").

ADELFE is interested in cooperative agents to be able to construct AMAS. The developer must think about all the unexpected and harmful events that could happen causing non cooperative situations 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. GUI are evaluated in A9 as functional or non functional (ergonomics, design, etc). Sometimes in this phase is necessary to go back to activity A8 to improve UI.

3.2 Analysis (WD3)

In the Analysis phase (Table 2), AMAS adequacy verification activity (A11) is included to identify agents and interaction among the entities. The developer identifies the components of the system studying use cases and scenarios previously elaborated in the domain analysis [7], [8], [9].

Table 2. WD3 – Analysis in ADELFE

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

The ADELFE AMAS technology adequacy verification of the system activity (A11) is to be performed using an adequacy tool that studies the system adequacy considering two levels: global (A11-S1) and components (A11-S2). Globally the study must answer 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 positive the developer can continue applying the process. If the second question's answer is also affirmative the ADELFE methodology should be applied on the components considered as AMAS since they require evolution.

The activity of identifying agents (A12) was also introduced to RUP and has to analyse the entities defined in A6 that will be considered an agent in the system. In ADELFE, agents are not previously known thus the developer must identify them. 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 meet all these criteria will be identified as agents and the classes related to them marked as agents.

The focus of AMAS system development is on cooperative agents who come from the previously defined entities (A6-S1) and the classes elaborated (A10-S1). The cooperative agents are 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.

The activity of study the interactions between entities (A13) was also incorporated to the process. This activity studies the interactions between active/passive entities, between active entities and between agents. The interaction between entities is represented by Collaboration and Sequence Diagrams. The agents' interactions are described by AUML Protocol Diagram.

3.3 Design (WD4)

The first activity of the design process (Table 3) identifies the detailed architecture of the system, creating packages sub-systems, objects, agents and the relationships among them, aiming at improving the class diagrams with the new elements occurring after the new agents were accepted [7], [8], [9].

Table 3: WD4 – Design in ADELFE

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 behaviors

The developer in the new activity A15 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 we describe these exchanges using AUML protocol diagrams. The protocols diagrams are attached to package (not classes) because they are generic. The language definition is not necessary when agents' communications is via the environment.

The activity Design Agents (A16) is an ADELFE methodology specific activity and allows the developer to refine the CooperativeAgent stereotyped classes identified in the A12 and A15 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. Similar as skills, aptitudes, interaction languages, design representations and design characteristics are defined with a stereotyped notation. Aptitudes are the capability of the agent to reason about a specific knowledge of the system or about a real situation.

The developer analyses protocols defined in A15 activity as well as 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 activity of defining agents (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. A table can be defined listing some type of situations like ambiguity, incompetence, uselessness, conflict. The developer should fills up the conditions described for each NCS. This 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 [7], [8] 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.

4. The Guardian Angel ADELFE Modeling

This first version of the system permits that final users and service providers establish contact when they can share a common interest in a dynamic and distributed context. The main requirement of this scenario is: (i) to give relevant information to final users for a certain query; (ii) to ensure service providers can expose their information to relevant users.

The system has to provide: notification and guidance to personal users; propagation for request or inquiry between the system actors; distribute information for potential users interested on receiving this information; obtain information from providers that reflect real users desires.

This first version of the Guardian Angel (GA) model was developed using the Work Definitions for the early and final requirements, analysis and design, the AMAS Adequacy tool and OpenTool [7], [8]. The development process was lasted three months and the doubts we had were answered by e-mail by Carole Bernon, one of ADELFE authors, from IRIT (Institut de Recherche en Informatique de Toulouse).

4.1 Preliminary Requirements

During the preliminary requirements phase we identified the following functional requirements: (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) customizable services allowing the user objectivity, adequacy and efficiency; (vii) improve education functionalities to the user such as access to encyclopaedias and universities researches to find knowledge from their diseases; (viii) provide alert and agenda functions remembering the patients of their appointment, medicines, dosage and contraindications; (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 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.

The following non-functional requirements were defined in this phase: (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 during the 7 days of the week, 365 days per year; (iv) be multitask and allow to answer to several data request simultaneous in a certain average time; (v) to be conceptually distributed (the small parts are all inside of the same environment, however they represent, separately, concepts and well distinct parts); (vi) to allow suddenly appearance and the abrupt disappearance of its components; (vii) to allow the adaptation and evolution of its components.

The stage of requirements validation and the definition of agreement requirements were carried out by the developer with the supervision of an adviser professor. We have also defined the main key words: Monitoring, GA, Patient, Communication, Health Professional, Insuring, and History Information.

One limitation relates to maintenance routines. A pre-defined agenda has to be followed. During these maintenance routines the system will not be available. Another restriction is that the system is not responsible for operating subnets with which it interacts, which implies that in case of eventual problems the user will be unable to access them until they become available again.

4.2 Final Requirements

The final requirements definition phase started with the environment characterization activity (A6) where the following passive entities had been identified: World Wide Web, Library, Hospital Stay, Illness Organism Information, Idiopathic Cause and Therapy. The considered active entities were: 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 Patient is the central entity of the Guardian Angel, having the ability to (i) activate any events in any circumstance at his convenience, dynamically interacting with the system; (ii) the Family can modify patient's treatment routine depending on the treatment satisfaction degree and its results; (iii) to dynamically interact with the system.

The Health Professional 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 interacts dynamically in accordance with the recurrently perceptions of the environment. This was divided into 4 distinct categories, searching specialization: (i) Analyzer - GA directed towards the tasks which require analyses, interpretation and understanding of data under one determined context; (ii) Inspector - GA directed towards the monitoring/inspection of specific states in the system; (iii) Diplomat - GA geared towards the reduction and treatment of Non Cooperative Situations. The GA Diplomat is responsible to use its "diplomacy" together with a GA Analyzer that helps it to determine the priorities of the GAs in execution, and (IV) Worker - the GA worker is the basic processing cell with the physical operations required to modify data/state of the system.

From the entities identification and classification (A6-S1) we built the Collaboration Diagrams for passive entities and the Sequence Diagrams for the active entities (A6-S2). In this context definition we modeled 12 Sequence Diagrams related the patient entity and figure 1 presents an example.

During the characterize environment (A6-S3) step the Guardian Angel system environment 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 in 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.

After the environment definition we define use case diagrams (A7-S1) divided in five groups: GA Domain, Patient, Institutions, Administrative and Service. For each group we designed one Use Case Diagram involving several use cases for each group and then for each Diagram we identified some NCS (A7-S2) as we show in Figure 2. We defined 16 User Interface prototypes and an example is presented in Figure 3.

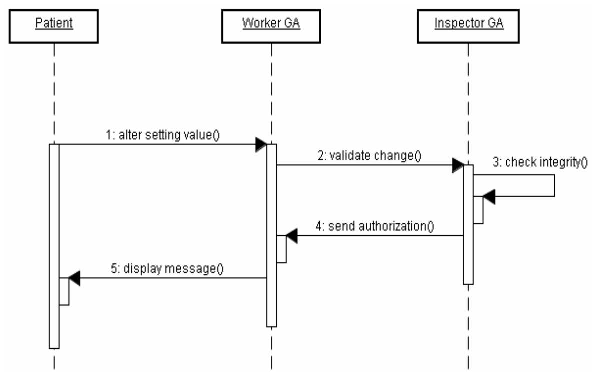

Figure 1 – Sequence Diagram: Customize Setting to Adapt Treatment to Patient's Reality

4.3 Analysis

In the Analysis phase we developed the Class Diagram, identified agents and refined interactions through the Collaboration and Sequence diagrams. We also developed the Agents Protocols Diagrams.

In the Domain Analysis we found four new passive entities (Idiopathic Cause, Therapy, Hospital Stay e Disease-Causing Organism), and we had to modify some diagrams and documents developed during previous steps.

The classes identified were: User, People, Patient, Family, Health Care Professional, Doctor, Guardian Angel (Analyzer, 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 system got the following reply from the tool in relation to the first 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".

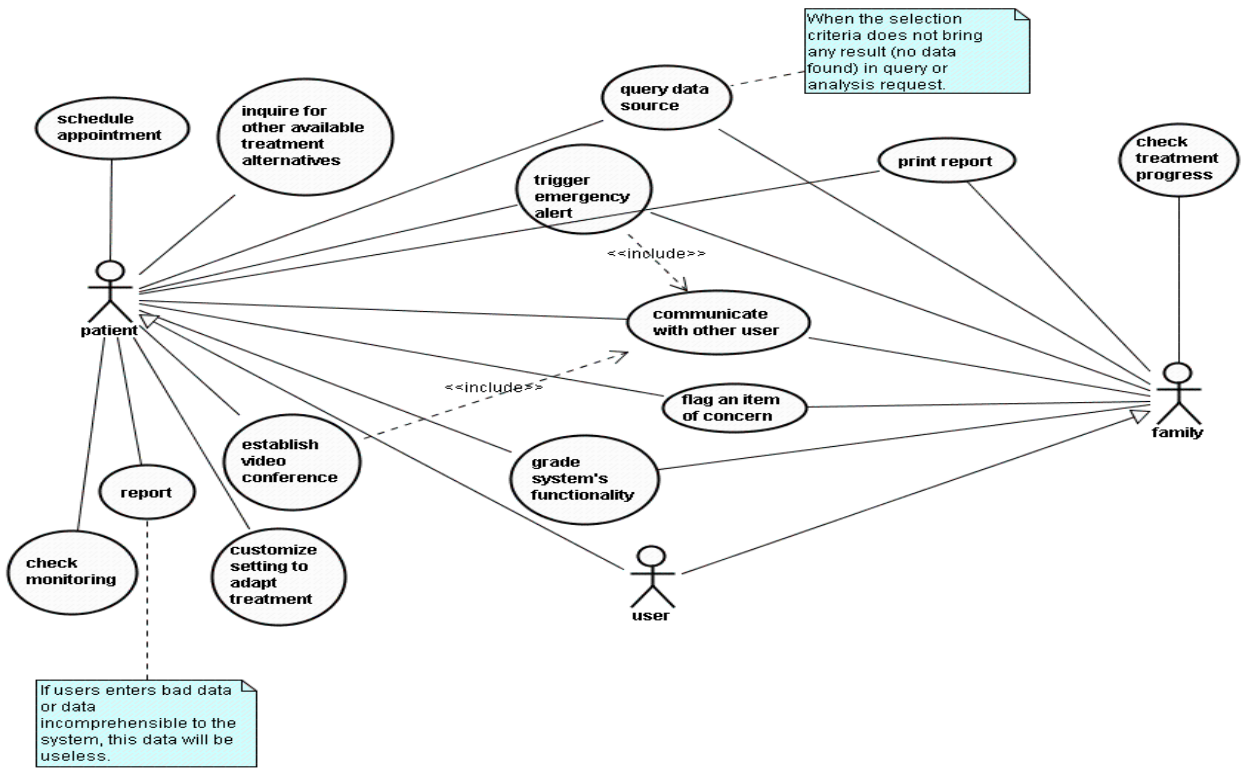
When the selection criteria does not bring any result (no data found) in query or analysis request.
query data source
schedule appointment
inquire for other available treatment alternatives
print report
check treatment progress
trigger emergency alert
<<include>>
communicate with other user
patient
<<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.

Figure 2 - Non Cooperation Situations: User (patient)

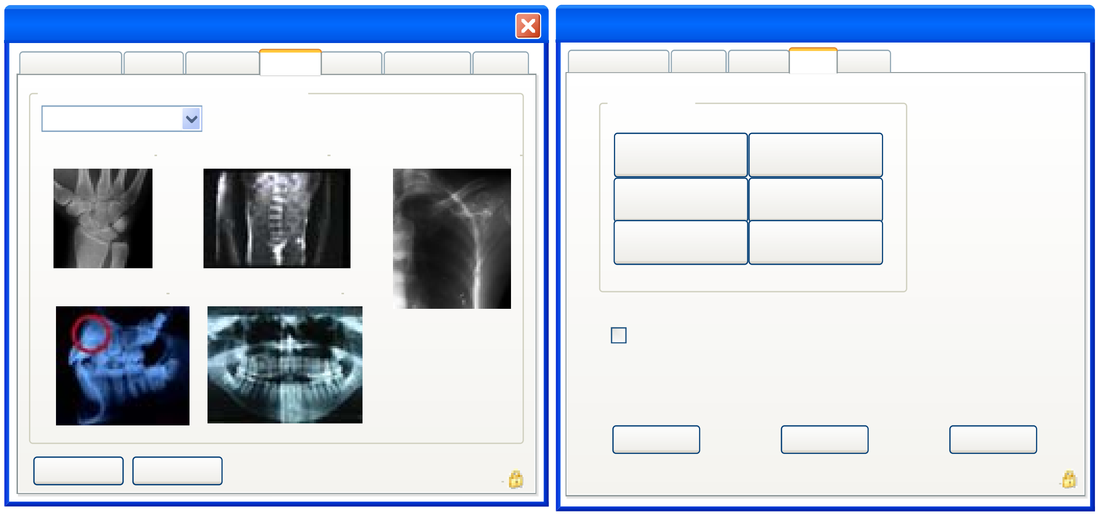

Figure 3 - UI Examples of Query Exams Repository" and "Adds new record to patient's history

The identify agents activity (A12) studied active entities and for each entity a form was defined as shown in Table 4 (without the two last lines). Thus four cooperative agents had been identified.

4.4 Design

The Design phase defined the packages and classes by elaborating the class and collaboration diagrams. No design pattern was applied and the activity A17 of Fast prototype was not carried out because the JAVA version do not work in the project computer due to some incompatibility that we could not fix with a new JAVA version.

Table 4 - Form for identification of agents in potential

Guardian Angel	
Autonomy:	Has autonomy because can make decisions base only in his knowledge
Local Goal:	The local goal is to perform a task that was assigned to him.
Interactions with others Entities:	Interact with other entities like others Guardian Angels and Patient.
Environment Partial Overview:	Limited overview of the system
Negotiation Abilities:	Capable to Negotiation with others 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 action are taken.
Face NCS	Yes - can request a service that is not available
Treat NCS	Yes- For example when a GA do not receive an answer for a feedback request.

During activity A15 we studied the interactions between the agents and we defined some AUML Protocol Diagram as the one shown in the figure 4. For each Guardian Angel we identified the abilities, aptitudes, representations and characteristics. We also defined the protocols used in A15 activity which will be used by the agents. Finally we defined the NCS in a form (Table 6).

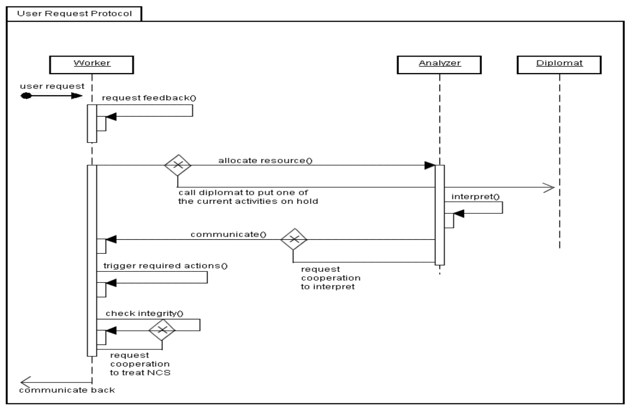

Figure 4. Protocol Diagram 2.

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

5. The Evaluation of *ADELFE*

The Evaluation Framework proposed in Yu and Cysneiros [1] was fundamental for ADELFE evaluation because of the practical, real and complex example to test and to verify the methodology simulating real situations for modeling. Some of these scenarios were incomplete, intentionally omitted to test how the methodology directs and supports the requirements findings. The evaluation questions complete these scenarios, identifying situations where the methodology must also guide in the requirements elicitation and analysis and specific cases of modeling that the methodology should attend. Previous works of methodologies evaluation of Tropos [22], Gaia [23], MESSAGE [24] and UML/RUP [2], [3], [4], [5], using this framework showed its efficiency for analysing the strengths, the weakness and the potentialities of each methodology.

Table 5 - The Identification of NCS Form

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 with 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 6 presents a general analysis of ADELFE methodology, showing answers for the main aspects addressed by the evaluation questions. Table 6 shows the concepts addressed by these questions and the answers rated as "S", "N" or "W". "S" means that the question is strongly supported by the approach, i.e., the methodology provide enough constructs, mechanisms and guidance. "N" means that, although the approach supports the aspect addressed by this question, it does so only up to some extent and therefore, only partially provides ways to model this aspect. "W" means that the approach either does not provide any constructs, mechanisms and guidance or provide very few, hence the requirements engineer would hardly be able to deal with the aspect raised by this question.

We also show others evaluations approaches found in literature and compare the results of these approaches with ours.

5.1 General Evaluation

ADELFE is a methodology originated from object orientation based on UML and RUP which are sufficiently well known in software development. In this methodology, the AUML protocols diagram was also incorporated. Although the process is well defined in [8], the documentation many times was superficial. In general, the concepts are shown in very simple case studies. Thus we attribute the neutral degree for learning curve (QA32) and strong for the support of tools (QA31). Support was received by email (Carole

Bernon <carole.bernon@irit.fr> which were fundamental for understanding the methodology.

Table 6. Exemplar Evaluation Questions Graded for ADELFE

	ADELFE
QA1 - Proactiveness	S
QA2- Human Autonomy vs software autonomy	S
QA3 - Autonomy reasoning	S
QA4 - Different levels of Abstraction	N
QA5 - Identifying participants in the domain	N
QA6 - Capturing, understanding and registering terminology	N
QA7 - Domain analysis	N
QA8 - Finding requirements	S
QA9 -Human-machine cooperation	S
QA10 - Database design	N
Q11 - Database evolution	N
QA12 - Database design and legacy	N
QA13 - Reasoning about different non-functional aspects	S
QA14 - Mobility	W
QA15 - User interface design	N
QA16 - Generating test cases	N
QA17 - User interface design (usability)	N
QA18 - Architectural design and reasoning	N
QA19 - Eliciting and reasoning about Non-Functional aspects	N
QA20 - Architectural design and reasoning (flexibility)	N
QA21 - Architectural design and reasoning (cost and confidence)	N
QA22 - Validating specification over the life cycle	S
QA23 -Tracing changes in the requirements into design	W
QA24 - Tracing changes from design to code	N
QA25 - Concurrency	S
QA26 - Tracing back to requirements	W
QA27 - Software Modularity	N
QA28 - Formal Verification and Validation	N
QA29 - Project Management	W
QA30 - Working in distributed teams	N
QA31 - Tool support	S
QA32 - Learning curve	N
QA33 - Integration with other methodologies	N
QB7 - Lightweight versions of methodology(simpler problems)	N

ADELFE allows to easily modelling cooperative agents who have autonomy (QA1) to take decisions that perceive the environment. Moreover, it is possible to define situations involving autonomy in both the knowledge level, (in execution time) and the modeling/implementation level (time of design) where we can analyze the human autonomy versus the software autonomy (QA2). QA3 the question addresses the agent autonomy reasoning and this situation can be modeled in ADELFE as a stereotyped cooperative agent class that uses the methodology concepts of cooperation. The GA_home_computer can execute a perception method to verify if some appointment in a next period can be scheduled without urgency (a routine examination appointment) for example. After that, a method of priority decision analysis of the commitments would be executed, and finally it would promote the change in the patient schedule, if there was an evidenced possibility of agenda reorganization (called the methods action to update patient calendar, to update medical calendar, etc.).

ADELFE methodology works with the identification of Non Cooperative Situations or fails in several occasions, where the cooperative work is weak. Therefore, the analyst has the opportunity to anticipate the analysis of machine-user cooperation problems (QA9) by identifying and analysing possible points of ambiguity and lack of understanding leading to create more efficient models.

Although ADELFE allows modeling in different levels with 18 activities where the software engineer can model from the abstract level to the design level, we rated neutral as for the support for navigating through different abstraction levels (QA4) because the methodology is not complete. For the same reason we also rated neutral for design traceability to the code (QA24) since this aspect is not explicitly defined. Our experience suggests that there should not be any major problem regarding packages and class identification for incorporating new behaviors However, in the requirements traceability (QA23 and QA26) ADELFE is weak in situations requiring different steps not explicitly defined and "traces-back" is not very trustworthy.

The domain participants' identification (QA5) was considered neutral. ADELFE only adopts diagrams based on UML or AUML. In steps A4 (identify key-words), A6 (Characterize environment) and A7 (identify use-you marry) we can identify the participants but we have a mixed list with actors and resources. We consider that ADELFE lacks a diagram with a general organization overview as we can have in the Tropos (actor diagrams) or MESSAGE (organization diagram) methodologies.

ADELFE identifies the set of keywords (QA6), but does not consider any domain glossary or ontology.

In the domain analysis (QA7) the modeling and the reasoning of the social relations can be only partially done. We rated this characteristic as neutral because we can only represent the communication between parts.

ADELFE adopts a process with preliminary and final requirements definition that introduces a requirements revision activity and consensual requirements definition. Moreover, an elaboration activity of keywords exists that improves the emphasis in the system, although for preliminary requirements ADELFE do not propose any diagram.

The reasoning of the different non functional aspects (QA13) can easily be modeled in ADELFE as a typical situation of Non Cooperative Situation. Thus, once identified the possibility of one determined scenario to happen, the involved agents in the communication must have the intelligence to identify such circumstances and to look alternative routes for this communication when possible, or to try to attend to a request based on the most recent transactions with this data. However in the non functional requirements elicitation and reasoning (QA19) we consider the methodology neutral. ADELFE considers a revision of the non functional requirements elicitation in activity 9, when it validates the user interfaces elaborated in terms of functional and non functional requirements however, it does not allow us to analyze the influence of the different non functional requirements as it is possible in TROPOS [22].

ADELFE proposes to frequently carry out specifications validation (QA22) during the software development life cycle, as for example through the diagramming of the system interactions and messages exchange. However, the methodology is only developed up to the design phase thus this validation is incomplete. ADELFE also adopts revisions of immediately previous steps during the process activities, such as in the requirements elicitation and modeling. The OpenTool tool at the end of the process can verify and validate some artefacts. However the methodology does not define any formal verification so neutral degree was attributed in the QA28.

Another interesting observation is related to answering QA33 "Integration with other methodologies". We couldn't find any guidance to integrate ADELFE with other methodologies although the notations are all based on UML and AUML diagrams. For example, it is not clear how we could implement part of the exemplar using object-oriented approach since there is no guideline on how to integrate ADELFE and UML models. The same issues we found answering QB7 - Lightweight versions of methodology for simpler problems. We presume that this is probably possible but ADELFE does not mention these situations.

5.2 Comparing with Correlated Works

In the agents orientation literature we can found works comparing methodologies, as Sturm and Shehory [13] that defines one framework of quantitative and qualitative evaluation and uses it in the evaluation of GAIA, Adept and Desire methodologies using a auction case study. Another interesting evaluation work was the one developed by Dam and Winikoff [14] that uses a framework based on attributes and a questionnaire sent and answered for the researchers, for methodologies authors and students who had modeled the Planning of Personal Itinerary case study. This evaluation tackled GAIA, MaSE, Message, Prometheus and Tropos methodologies. Iglesias and González [15] describe a survey of methodologies analysing its extensions in the objects and knowledge paradigms

Ceruzzi and Rossi [16] consider another evaluation that uses metric and quantitative methods, showing the framework utility by comparing the MAS-CommonKADS and Agent Modelling Technique for Systems of BDI Agents methodologies. This evaluation was based on previously defined criteria and through the results quantification and arithmetic average application.

ADELFE evaluation was proposed by Tran, Low and Williams [17]. In their proposal there were 50 criteria for evaluation grouped in process-related, technique-related, model-related and supportive-feature criteria. In Tran and Low's [18] evaluation table they consider ADELFE medium for ease of understanding and usability for defining interactions protocols but ADELFE uses the same AUML protocol as TROPOS and MAS-CommonKADS that is consider high. However some of these criteria are not analyzed, mainly the techniques relative ones. The article mentions that a deeper analysis must be carried through in a future.

In the designing of Gaia, PASSI and ADELFE meta-models, Bernon Bernon, Conssentino, Gleizes, Turci and Zambonelli [19] compare the methodologies regarding agent structure, agent interactions, agent society, organization structure, and agent implementation. One weakness pointed out in the paper is the concept of goal and plan in the agent structure because ADELFE considers that in a complex and open systems a plan can be built during system execution. Although we agree with it, ADELFE uses this goal notion when defining the skills of an agent but only in the Design phase when you can define skills and attitudes. During requirements and analysis phases these goals and plan identified could help agent identification. We also agree that one of ADELFE strengths is the definition of agent interaction abilities. They explain the lack of organization structures by the open societies that are modeled in ADELFE. The organization is only showed by the interactions between the agents.

Based on previous works [2], [3], [4], [5] we can conclude that ADELFE is a powerful methodology in terms of cooperative agents' concepts allowing the definition of autonomy, proactivity and autonomy reason, centered in a Non Cooperative Situations. However this methodology is not completely defined, needing to incorporate new models or to modify some diagrams to give more general overview of the system with its environment and most of all to support goal and plan definition during the requirements phase. It also needs to better define some steps as well as to improve the implementation phase as well as to improve traceability mechanisms.

6. Conclusion

This work is part of a broader project which aims at analyzing important aspects of requirements identification and modeling in multi-agents systems. In this particular work the goal was to establish a qualitatively evaluation of ADELFE methodology using the exemplar framework proposed Yu and Cysneiros [1].

ADELFE has a strong set of concepts to model cooperative agents and Non cooperative Situations covering the requirements, analysis and project phases with a well defined process. However, the methodology needs to improve some aspects of requirements traceability and participations identification. The characterize environment can be improved by adding new diagrams that can model goals, plan and organization aspects during early requirements.

Future work will be twofold. At one hand we will continue to evaluate the methodologies implementing the models we have obtained so far using an agent platform as JADE [20] or OpenCybelle [21]. On the other hand we intend to build a specialized version of the Guardian Angel for Diabetes Disease implementing in different platforms (PDA, cellular telephone...).

Finally we intend to compile 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

7. References

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

[2] Cysneiros, L. M., Werneck, V. M. B.; Yu, Eric, "Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar". *Journal of Computer Science & Technology*, USA, v. 5, n. 2, 205, pp. 71-79.

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

[4] Coppieters, A. M., Marzulo, L.A.J., Kinder, E. and Werneck, V.M.. "Modelagem Orientada a Agentes utilizando MESSAGE", *Cadernos do IME-Série Informática*, Rio de Janeiro, v.18, 2005, pp. 38-46 in portuguese.

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

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

[7] Bernon, C., Camps, V., Gleizes, M. P. and Piscard, G., "Engineering Adaptive Multi-Agent Systems: The ADELFE Methodology"; IN: *Agent-Oriented Methodologies*, Brian Henderson-Sellers e Paolo Giorgini (ed.), IDEA Group Publishing, 2005, pp. 172-202.

[8] ADELFE (Atelier de Développement de Logiciels à Fonctionnalité Emergente) at http://www.irit.fr/ADELFE.

[9] Bernon, C., Camps, V., Gleizes, M. P. and Piscard, G., "ADELFE: A Methodology for Adaptive Multi-agent Systems Engineering"; In: *Third International Workshop Engineering Societies in the Agent (ESAW 2002, Lecture Notes in Computer Science*, Volume 2577/2003, Springer Berlin Heidelberg, 2003, pp. 156-169.

[10] Krutchen, P., *The Rational Unified Process: An Introduction*, Reading, MA, Addison Wesley, 2000.

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

[12] Rumbaugh, J., Jacobson, I. e Booch, G., "*The Unified Modeling Language Reference Manual*"; Addison-Wesley, 1999.

[13] Sturm, A. e Shehory, O., "A Framework for Evaluating Agent-Oriented Methodologies", In: *Proc of 5th International Workshop on Agent-Oriented Information Systems* (AOIS'03), 2003, pp. 60-67.

[14] Iglesias, C.A. e González, J.C., "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, 1998, pp.317-330.

[15] Dam K. H. e Winikoff, M.; “Comparing Agent-Oriented Methodologies”, In: *Proc. of 5th International Workshop on Agent-Oriented Information Systems* (AOIS’03), 2003, pp.52-59.

[16] Ceruzzi, L. and Rossi, G. “On the Evaluation of Agent-Oriented Methods”, *Agent Oriented Methodology Workshop*, November 2002.

[17] Tran, Q.N., Low, G. and Williams, M., “A Preliminary Comparative Feature Analysis of Multi-agent Systems Developments Methodologies”, In: *Agent-Oriented Information Systems (AOIS-2004) Lecture Notes in Computer Science*, Volume 3508, 2004, pp. 157-168.

[18] Tran, Q.N. and Low, “Comparison of Ten Agent-Oriented Methodologies”, IN: *Agent-Oriented Methodologies*, Brian Henderson-Sellers e Paolo Giorgini (ed.), IDEA Group Publishing, 2005, pp. 341-367.

[19] Bernon, C., Massimo C., Gleizes M.P., Turci P., and Zambonelli, F. "A Study of Some Multi-agent Meta-models", In: *Agent-Oriented Software Engineering (AOSE-2004) Lecture Notes in Computer Science*. Volume 3382, Odell et al. (Eds,), Springer-Verlag Berlin Heidelberg 2005, pp. 93–108.

[20] JADE Java Agent Development Framework, at http://jade.tilab.com/.

[21] Open Cybelle Platform, at http://www.opencybele.org.

[22] Bresciani, P. Giorgini, P., Giunchiglia, F. Mylopoulos J. and Perini. A., TROPOS: An Agent-Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems. Kluwer Academic Publishers, 2003.

[23] Zambonelli, F., Jennings, N. R. and Wooldridge, M., Developing Multiagent Systems: The Gaia Methodology, In ACM Transaction on Software Engineering and Methodology, Vol. 12, No. 3, July 2003, pp 317-370.

[24] Caire, G., Coulier, W., Garijo, F., Gómez-Sanz, J., Pavón, J., Kearney, P.and Massonet. P., Message: A Methodology for Development of Agent-Based Applications, To appear at Methodologies And Software Engineering For Agent Systems, edited by Federico Bergenti, Marie-Pierre Gleizes and Franco Zambonelli, to be published by Kluwer Academic Publishing (New York), 2004.