The ADELFE Methodology For an Intranet System Design

Carole Bernon, Marie-Pierre Gleizes, Gauthier Picard, Pierre Glize

Institut de Recherche en Informatique de Toulouse
118 route de Narbonne
31062 Toulouse Cedex, France

{bernon, gleizes, picard, glize}@irit.fr

Abstract. The paper presents the ADELFE methodology which is devoted to software engineering of adaptive multi-agent systems. Adaptive software is used in situations where either the environment is unpredictable or the system is open. ADELFE guarantees that the software is developed according to the AMAS theory[1]. We focus the presentation on analysis and design core workflows. In the analysis phase, the engineer is guided to decide to use adaptive multi-agent technology and to identify the agents through the system and the environment models. In the design phase, ADELFE provides the cooperative agent's model and helps the developer to define the local agents' behavior. These workflows are illustrated through the case study: "US West Global Village". Then the Adelfe methodology is compared with others methodologies.

1 Introduction

Progress realized in programming and more generally in computer science, allows us to use software to solve more and more complex problems. The agent-based and multi-agent based software gives solution for complex applications where the environment is generally constrained. The next challenge is to build software that is more robust than actual one in unpredictable environments like the Internet or mobile robots moving in the real world. It now seems necessary to develop new models, new methodologies and tools. In this area, we had proposed a theory for adaptive multi-agent system design [3]. Now we're developing a methodology named ADELFE which is an open and evolutionary systems software engineering. It is dedicated to the developers of adaptive multi-agent systems.

The objective of ADELFE is to cover all the phases of a classical software design from the requirements to the deployment. It is based on the RUP (*Rational Unified Process*) [9], as interpreted in the Neptune project [11], uses UML notations and adds some specific steps to adaptive system design. As far as possible, the ADELFE project will reuse both RUP and UML's agent-oriented extensions. The aim is not to add

[1] This theory has been developed and applied for the last 6 years at IRIT (Institut de Recherche en Informatique de Toulouse). See http://www.irit.fr//SMAC

another methodology but to work on some aspects not already considered by existing ones such as complex environment, dynamic, software adaptation. ADELFE is based on the view of a multi-agent system as a dynamic organization consisting of various cooperative agents.

Section 2 gives an overview of the main concepts used in ADELFE. The use of ADELFE is illustrated by means of a case study: an Intranet system design, described in section 3, which follows the specification of the US West Village given by Bhattacherjee in [1]. The different workflows illustrated in the case study: requirements, analysis and the design, are respectively presented in section 4, 5, and 6. Before concluding in section 8, related works are discussed in section 7, in insisting on the main commonalities and distinctions.

2 Main ADELFE Concepts

ADELFE[2] is a French acronym that signifies "Toolkit for Designing Software with Emergent Functionalities". The main goal of ADELFE is to help and guide any designer during the development of an adaptive multi-agent system. Thus, it will not be assumed that the software developer is familiar with adaptive multi-agent systems but the methodology uses a specific vocabulary which is first detailed in this section.

2.1 Focusing on Adaptive Systems

The DARPA defines self-adaptive software as follows: "self-adaptive software evaluates its own behavior and changes behavior when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible" [13]. This software is used in situations where the unpredictable nature of the environment poses a formidable challenge. Therefore, self-adaptive software must be more robust than classical one to continue to run in such an environment.

We are working on a certain type of self-adaptive software named adaptive multi-agent systems (AMAS). An AMAS is characterized by the following points: it is embedded in an environment and composed of several agents, each agent carries out a partial function and the agents' organization during run-time makes the system realizing an emergent function.

[2] ADELFE (Atelier pour le Développement de Logiciels à Fonctionnalité Emergente) is a national project that started in December 2000 and is still in progress, it is supported by the French Ministry of the Economy, Finance and Industry. ADELFE project partners are: IRIT (University of Toulouse III) and L3I (University of La Rochelle) from academia and are ARTAL and TNI from industry.

2.2 Concerning the System

The reason why a designer uses ADELFE is mainly that he wants to develop a multi-agent system (MAS). A “classical” definition given by [6] tells that a MAS is a system composed of autonomous entities interacting in a common environment. From our point of view, a MAS possesses also an environment and it can reach a behavioral (to reproduce the behavior of a simulated entity) or a functional (to perform the task for which the system had been built) adequacy. Furthermore, the important notion in a MAS is the fact that the function is collectively realized, is coherent and has a meaning.

Adaptive MAS specifically interest us; classically such a system is defined as a MAS which is able to change its behavior to react to the evolution of its environment. The specificity of our AMAS theory lies in the fact that we don't code the global function of the system within an agent. Each agent composing the system is able to change its interactions with others. The global function is realized by the collective and if the collective is able to change its interactions, the global function is then modified. Therefore this capacity of self-organization enables the system to adapt itself and to realize a function that it is not coded in the agent. The global function is then emerging.

2.3 Concerning the Non Cooperative Situations

When the environment is unpredictable or the system is open, traditional algorithms fail because the developer cannot find algorithms which consider every possibility. Our objective is to build systems that do the best they can when a difficulty is encountered. This difficulty can be viewed as an exception in classical programs. In the agent’s viewpoint, we call exceptions non cooperative situations such as misunderstanding, ambiguity, uselessness, conflict, and concurrency. We propose to the designer not only to describe what an agent has to do to achieve its local goal but also which local situations this agent must avoid and if these situations are detected how to remove them.

2.4 Concerning the Agents

During the analysis phase, the methodology will lead the designer to think about decomposing his system and it must therefore help him to decide if a component will be an agent or a simple object. In ADELFE, we are only concerned in a certain kind of agents which enable us to build AMAS. Therefore an agent must have some extra characteristics than those described in [6]. Firstly, an agent is also ignoring the global function of the system for this global function is emerging (from the agent level to the multi-agent level). One perspective would be to have a tool to judge if the emerging function is coherent with the expected one. Furthermore, an agent can detect non cooperative situations and it always acts to come back to a cooperative one. This doesn’t mean that it is altruistic or always helping others agents but it is just trying to have useful (from its point of view) interactions with others. What constraints each agent’s behavior is the behavior of the collective. All the considered agents are composed of five parts contributing to their behavior:

– *Skills*. An agent's skills represent what it is able to do or what abilities it may bring to the collective.
– *Representation of itself, of others or of its environment or beliefs*. These representations are what the agent knows about itself, the others and its environment.
– *Social attitude*. It is what enables the agent to change its interactions with others. This social attitude is based on what we call cooperation: if an agent detects a non cooperative situation, it acts to come back to a "cooperative" state.
– *Interaction language*. It is what the agent needs to communicate directly or not.
– *Aptitudes*. These aptitudes are the capacities an agent possesses to reason on its representations and on its knowledge.

2.5 Concerning the Environment

The AMAS theory focuses on complex applications design. In [17], Wooldridge writes that "one of the sources of such complexity is the nature of the environment". Thus the concepts used in ADELFE also concern the system environment. The classification proposed by [17] can be used to try to characterize the environment of the system to build using ADELFE. This environment can be accessible or not, deterministic or not, static or dynamic and/or discrete or continuous.

In applications developed with ADELFE the environment is rather inaccessible and dynamic for the system doesn't know all about its environment, it does not have a complete and accurate view of it, its actions can modify its environment but this latter can also evolve due to other phenomena external with the system. In an AMAS with a computational adequacy or linked to a human user, the environment is rather non-continuous and deterministic, the number of possible actions is not finite and an answer given by the system doesn't imply a definite reaction of the environment.

3 The "US West Global Village" Case Study

The following requirements are extracted from the "US West Global Village" project [1] in order to develop an internal company-owned computer networks or "intranet". The Global Village project would play a key role in achieving the goals of the reengineering initiative: improve business processes, increase employee productivity, and enhance customer satisfaction while reducing costs. These end-user requirements are derived from the analysis of the necessary dynamic of the society was slowed down by its current human organization and the internal information support (mainly paper). In order to by-pass this too static system (i.e. the US West company), a new communication mode must be proposed. To facilitate company-wide information processing and distribution, the Internet-based technologies has naturally emerged because of the internal skill of the company and its novelty as information support inside the company.

The early requirements in ADELFE correspond to a mutual agreement between the customer (including the users) and the designer on what the system should and should not do. Thus, the early requirements could be expressed as a continuous system adaptation (always the US West company) to a dynamic and constraining environment (i.e. the customer's needs) by the way of an "intelligent" communication network. In short, we assume that the early requirements can be supported by two technological approaches: the push technology and the pull technology.

– The requirements for the pull technology are the following: information sharing, communication and collaboration between employees using electronic mail, workflow software, and bulletin board system (BBS); transactions between employees which are interactive processes which allow employees to request specific information via on-line forms, and business computing.
– The requirements for the push technology concern: home page personalization, learning of emerging needs, targeted advertising, spontaneous communication and adaptive communities.

Thus, the Global Village can be seen as a dynamic organization reflecting -and also transforming- the real internal organization of the company. The following sections of the paper mainly focus on the application of the ADELFE methodology on the requirements presented in this push technology part.

4 Late Requirements

The late requirements provide the environment model that is composed of a system-environment interaction model and the characterization of the environment. The aims are to define a view of the system, to transform this view in a use-case model, and to organize and to manage the requirements (functional or not) and their priorities. From a multi-agent point of view, the environment model is a primary objective to design adaptive systems. The model is the result of a study of the different actors and their possible interactions with the system. Data flows between the system and passive actors are represented in collaboration diagrams. Interactions between the system and the active actors are summarized in use-case diagrams and detailed in sequence diagrams.

Step 1: Definition of the Studied System. This definition is the result of the analysis of the requirement set artifact. The output of this step is a set of the main terms that define the system. The system to design allows exchanges of information between several employees. The employees could have an active behavior, by sending e-mail for example, or an inactive one, by receiving e-mail. Shared information consists in web page contents or free text (mail, request …).

Step 2: Determination of the Actors. This step focuses on what may be in interaction with the studied system in terms of passive or active actors, or constraints. Active actors are employees and management operators. Employees are the main end-users of this system; they may send or request information to other employees. Management operators correspond to the system administrators[3]. Passive actors are the system databases: the employees database, the web pages database and the Bulletin Board System (BBS).

Step 3: Definition of the Context. This step requires a characterization of data flows and interactions between passive or active actors and the system. This step produces collaboration diagrams and sequence diagrams (actor/system or actor/actor). We identify five different patterns:

- Bilateral data flow between the system and the passive actors (employees database, web pages database and BBS). These exchanges correspond to database updating by the system;
- Bilateral interaction (request/response) between the system and an employee to retrieve information (pull technology);
- Unilateral interaction from an employee to the system. The employee provides new information to the system;
- Unilateral interaction from the system to an employee. The system communicates a relevant information to an employee (push technology);
- Database updating by the management operator via the system. The management operator can create, remove or update information about web pages or employees.

Step 4: Characterization of the Environment. Using terms which are presented in section 2.5, we characterize the environment of the system as being dynamic (the evolution of the active actors is not only subjected to actions of the system), accessible (the system can obtain all information about the environment's state), non-deterministic (the system is not able to know what are the effects of its action on an employee), and continuous (the number of interactions is infinite).

Step 5: Determination of the Use Cases. The main objective of this classical step is to clarify the different functionalities the system has to respond to. Only the active actors are implied in these use cases which are the results of a functional requirement set. The whole set of use cases is presented in figure 1 in which all possible interactions between the system and the active actors are described. Each use case is detailed in a specific sequence diagram, a textual description and Graphical User Interface (GUI) prototypes.

[3] In this article, we only focus on the information exchanges. We don't deal with management functions of intranet system such as database or remote machine managing.

5 Analysis

After the five steps of the requirements, the objective of the analysis workflow is to develop an understanding of the system, to structure it in terms of components and to know if the AMAS theory is required.

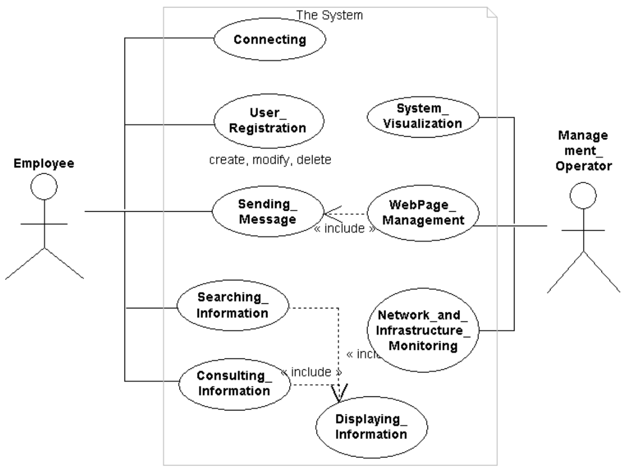

Figure 1. Use-case diagram for the US West Global Village.

Step 6: Domain Analysis and Study of the Architecture of the System. Domain analysis is a static view and an abstraction of the real world and the linked entities. Considering separately each use-case by defining scenarios, the designer has to divide the system into entities. These entities may be evident in distributed problems (for instance on-line brokerage) or may be more subtle in complex tasks solving (for instance flood prediction). The result of this step is a set of entities in preliminary class diagrams.

The US West Global Village application is composed of several sites which are geographically distributed and which concern a great number of employees and/or web pages. A site is represented by the `SiteRep` entity. In this site, information is supported by a set of `WebPageRep`, `EmployeeRep` and `BBSRep` entities which respectively encapsulate a web page representation, an employee representation or the BBS representation. The `ConnectionAssistant` helps the employee to connect or to register to the system; it maintains the employees' profiles (login and password for example) and verifies accesses to the system. The `SiteManager` is the intermediary between the management operator and the system. This medium enables the manager to update web pages for example.

Step 7: Adequacy of the AMAS Theory. As a primary goal, a designer wants to know if the AMAS theory is adequate to implement his application. This kind of programming is sometimes completely useless. For example, having a system which is able to adapt itself is completely useless if the algorithm required to resolve the task is already known, if the task is not complex or if the system is closed and nothing unexpected can occur. A certain number of criteria are then proposed to the designer, via a graphical tool in ADELFE, to help him to determine the adequacy of AMAS at two levels: at the global level "is an AMAS required to implement the system?" and at the local level "do some agents need to be implemented like AMAS i.e. do some decomposition needed".

Step 8: Agent Identification. In this step, we are only interested in agents which enable a designer to build AMAS. So the designer has to determine which entities fit in with this agent type. This identification is done considering the definition of the agents given in section 2 and their characteristics. Firstly we have to know where a lot of evolution or dynamic is required. Then, for each entity identified during step 6, we must examine if the entity has to be faced to unpredictable events and has to treat non cooperative situations (as they are defined in section 2.3).

For the `SiteRep` entity, employees and web pages can be removed, added to the site or their information can be modified. The `SiteRep` entity must therefore be able to learn the changes at these levels. Furthermore a site may exchange information with another distant site to gather information it doesn't possess locally, it is a means to cooperate and the site can misunderstand a request which is coming from another site, or it does not know to reply to the request because the request is out of the scope of the information it has. These situations are some non cooperative situations and the `SiteRep` entity must react to them. Consequently it evolves a lot and will be viewed as an agent.

The `ConnectionAssistant` entity is not an agent because its function doesn't evolve, it hasn't any autonomy, it doesn't need to interact (cooperate, negotiate) with other entities. In fact, it has not to cope with unpredictable events.

At the end of this step, by using similar reasoning, we can conclude that only `SiteRep`, `EmployeeRep`, `BBSRep`, `WebPageRep` entities are agents.

Step 9: Study of the Interactions between the Different Entities. The result of this step is a set of sequence diagrams and activity diagrams which explains the possible interactions between the different entities within the system. As the RUP is use-case guided, for each use case, which has been defined (during step 5) between the system and the environment, a sequence diagram has to be defined to show the internal view and the interactions within the system.

6 Design

In this section, we detail the design workflow in four steps. Because the complete design cannot be described in this paper, we only develop the steps which are not existing in others methodologies such as the agent architecture and the modeling of non cooperative situations.

Step 10: Detailed Architecture and Agent Model. The first step of the design requires to identify the software components and to describe them. The result provides the architecture of the system in terms of needed blocks, classes, agents and interactions. The agent model which represents the relationships between agents is included in this architecture. The previously defined architecture can be refined by determining if some design patterns and/or re-usable components can be used. For example, in object-oriented methodologies designers try to re-use models such as customer-server model… In ADELFE, we propose a specific design pattern named cooperative agent architecture.

Step 11: Agent Architecture. Following the description given in the section 2.4, an agent is composed of: two types of models, social attitudes, an interaction language, and aptitudes. These components are then detailed:

– *Skills model*. If the agent must learn new skills, ADELFE suggests to the designer to apply again the methodology to realize an adaptive multi-agent system to implement the skills of the considered agent. The skills of a `SiteRep` agent are the sum of the skills of its internal agents. Its skills may evolve a lot; this is why they compose an adaptive multi-agent system. The agents of this system are described in the step 8. The skills of `WebPageRep`, `BBSRep`, `EmployeeRep` agents are the information possessed by the web pages, the BBS or the employees.
– *Representation of itself, of others or of its environment models*. They can be represented by methods and/or attributes if this information is a static one. If they have to evolve and if the agent may learn new representations, ADELFE tells the designer to apply again the methodology to realize an adaptive multi-agent system for implementing the representations of the considered agent. The representations of a `SiteRep` agent may evolve during runtime and this agent has to adjust them. Therefore an adaptive multi-agent system is required to implement its representation of itself or of other `SiteRep` agents. The ADELFE methodology is started again to implement the software to manage these representations. In the same way, the representation possessed by the different agents within a same site (`WebPageRep`, `BBSRep`, `EmployeeRep`) may evolve during runtime and each agent has to adjust it. Therefore the ADELFE methodology is applied again to implement these representations as adaptive multi-agent systems.
– *The interaction language* between agents can reuse the mailbox concept to physically exchange information between several machines. Some acts of ACL such as request, inform… can be reused to communicate.
– *Aptitudes* allow an agent to learn (to add, to update and to remove) the representation of itself and of others, and to interpret a received message.

Step 12: Non Cooperative Situations Model. The designer must fill up a table for each Non Cooperative Situation (NCS). This table contains the agent's state, the textual description of the NCS, the conditions and actions linked to the NCS (as it is done for the `SiteRep` agent in the next section). The conditions describe the different elements that enable to locally detect the NCS. The actions describe what the agent has to do to remove this NCS.

In the case study, we found four NCS for the `SiteRep` agent:

1. *NCS1 - Total lack of understanding.* Condition: the agent compares the contents of the received request with the representation of itself and deduces if the request is understood. Action: the agent searches if it knows other sites which could be relevant for the received request. If it finds some it sends the request to them.

2. *NCS2 - Partial lack of understanding*. Condition: only one part of the received message has a meaning for the agent. Action: this agent brings to the sender the partial answer associated to the understood part of the message, it sends the other part of the request to a more relevant agent.
3. *NCS3 - Ambiguity*. Condition: a received message has several meanings for the agent. Action: it turns it up to the sender for clarification; if this latter cannot answer, it turns it up to its own sender.
4. *NCS4 – Conflicts*. Condition: two agents want to access to a service which provides a limited resource and their demand exceeds the offer. Action: the `SiteRep` agent has to try to relax constraints.

The Non Cooperative Situations for the `WebPageRep`, `BBSRep`, `EmployeeRep` Agents are the same as those encountered by a `SiteRep` agent. There is a difference in the actions realized in NCS1 and NCS2, the agent searches if it knows other agents within the site that could be relevant for the received request. If it finds some agents it sends the request to them. If it does not find such agents it sends the request to its `SiteRep` agent. The NCS will guide the engineer, during the construction in the implementation workflow.

Step 13: Class Diagrams. This step provides the different class diagrams for taking the GUI and database designs into account. These class diagrams are also refinements of the class diagram at step 6 and of the architecture at step 11.

7 Comparison to Other Approaches

In some recent states of the art about agent-oriented software engineering [8], [16], the authors divided globally methodologies into two groups: the object-oriented based methodologies and those that come from the knowledge engineering or other techniques. Because ADELFE belongs to the first category, we compare in this section only some representative methodologies of the first category like: AAII [10], AOR (Agent-Object Relationship) [15], AUML [12], GAIA [18], MESSAGE [5], TROPOS [4]. The comparison presented here isn't a detailed one that follows each step of a methodology. It is rather a comparison focusing on some main features, from general ones to more precise ones, of the considered methodologies.

The Extend of Coverage Over the Different Phases. The methodology designer should be interested in developing a detailed and complete methodology from existing analysis till software deployment and maintenance. It is now widely accepted that the core workflows of a methodology are the requirements[4], the analysis, the design, the development (also called implementation) and the deployment workflows [9].

Some methodologies cover the entire process of software engineering such as TROPOS, MESSAGE and ADELFE. For example, TROPOS insists a lot on the requirement phase which is divided into early requirements and late requirements.

[4] In some methodologies, requirements are included in the analysis phase.

MESSAGE covers at least analysis and design phases but it should also explain the relationships with implementation, testing and deployment.

Other methodologies essentially process the analysis and the design phases. AAII provides specialized set of models: the agent and the interaction models capture the notion of roles, responsibilities, services and control relationship between agents at the external level and the belief, goal and plan models to design BDI agents at the internal level. In GAIA, the analysis phase consists in filling up the role and the interactions models. The agent, service and acquaintance models are developed in the design phase. Odell, Parunak and Bauer's works propose a representation of the Agent Interaction Protocols (AIP). Then, they suggest further extension to UML in AUML in extending sequence diagrams to include richer role specification and in changing the deployment diagram definition to allow representation of mobile agents. AOR focuses on analysis and design of organizational information systems in which an information system is an agent composed of sub-agents and in interaction with external agents.

The Specialization of the Methodology for an Application. The challenge of agent or multi-agent methodologies development is the same: to help the designers for building complex systems such as multi-agent systems. Two categories of works could be found: those which are general high-level methodologies, like AAII, AUML, GAIA and those which focus on a specific design because the methodology is dedicated to a field of applications.

AOR concerns information systems design and MESSAGE deals with the development of agent-based applications in the telecommunications domain. ADELFE was developed for applications where:

– The system's environment is dynamic, making it ineffective to enumerate exhaustively all the situations the system may encounter.
– The system is open and therefore dynamic because it is constituted of a shifting number of components.
– The task the system has to achieve is so complex that we cannot guarantee a perfect design.
– The way by which the system may achieve the task it has been assigned is difficult or even impossible to apprehend globally by the designer.

So it is a suitable methodology for the development of adaptive multi-agent-systems.

MESSAGE and ADELFE give guidelines for the identification of the application areas where agent technology for MESSAGE and adaptive systems technology for ADELFE is better suited than other technologies e.g. object-oriented technologies. To guide the designers which are not specialized in agent or MAS technology, a tool in ADELFE helps them to decide if ADELFE is useful to design the software. This differs from most of the methodologies in which the question: ”Is the problem requires agent or MA technology?” is not asked because the initial hypothesis is that agent or multi-agent techniques are available and relevant for the application.

The Underlying Specific Architecture. Inside the AAII methodology, the internal model is concerned with the internal of an agent and based upon the Belief Desire

Intention agent architecture. Inside ADELFE, at the agent level in the design phase, the agent model proposed to the designer is a Cooperative Agent (CA) architecture [7]. In other methodologies such as AUML, GAIA, MESSAGE, TROPOS the architecture of the implemented agents is not defined and it is quite open. Note that TROPOS offers different architecture styles (flat structure, pyramid, joint venture…) for its architectural design phase.

From the Existing Object-Oriented Methodology or Notation. In the cited methodologies, all are derived from object-oriented ones except AOR which is inspired by the agent-oriented programming proposal of Shoham [14]. By building upon and adapting existing well understood methodology, the authors take advantage of the maturity and the well-known object-oriented methodology and aim to develop a methodology that will be easily used and understood by software analysts and engineers. MESSAGE and ADELFE are based on the RUP methodology and uses UML notation or extension of UML already done in AUML in particular the AIP notations [12]. This choice is justified because UML is a de facto standard for object-oriented modeling and that promoted its rapid take-up. TROPOS adopts the i* methodology [19] for requirements workflow and AUML for the detailed design workflow. AAII draws primarily upon object-oriented methodologies and extends them with some agent-based concepts. GAIA is based on the FUSION methodology.

Static Domain Vs Dynamic Domain. The actual challenge of methodologies is to support the software development which is open, have evolving structure, can react to unexpected events from its environment. This kind of software must become more robust than actual one.

The dynamic aspect is naturally taken into account at three levels of ADELFE: at the environment-system interactions level, at the level of the system composition i.e. agents could be added to or removed from the system and at the agent level i.e. agent's beliefs and skills can evolve. TROPOS expresses the dynamic and opening of the application in the requirements phases with the model of the environment and with particular soft goals. For example in the Media shop case study there is soft goal adaptability. But the methodology does not give guidelines for designing the right agents' behavior allowing the adaptability of the system. GAIA indicates that the domain covered by the methodology is static and that the methodology is dedicated to closed domain where agents' skills and beliefs are static at runtime. But Zambonelli, Jennings and al. [20] propose some improvements of GAIA to support applications in dynamic domains. AOR allows an integrated treatment of the static, dynamic and deontic aspects of information systems. Many other methodologies, like AAII, MESSAGE, do not focus on the dynamic aspect of the software environment and on the adaptation abilities of the software.

The Role Model. The role is an important concept in the recent methodologies. It represents an abstract level of agent description. Generally, roles are defined in the analysis phase of the methodologies like in AAII, AUML, GAIA, MESSAGE or TROPOS. The roles in AAII enable the elaboration of an agent class hierarchy. A role is described by a set of responsibilities and responsibilities are described as a set of services. In GAIA, a role is defined by four attributes: responsibilities, permissions, activities and protocols. The role model is a requirements guided identification of the key roles in the system. In MESSAGE, roles and agents are the main concepts involved in the agent model. The actors can be agents, positions or roles in TROPOS and they are produced at the early requirements analysis.

In some methodologies, the role concept is not explicitly present, because the description of the agents and relations between agents are made at a less abstract level. In AOR, the main concepts are agents, events, actions, commitments, claims and objects. In ADELFE the definition of the roles for each agent is not necessary to develop the system because the agent could change its role during run-time. If the agents in the system can be observed during run-time, the observer can give roles to them. The closest notion to the role one is the skill concept. But skills defined for each agent do not represent its role because they are elementary actions an agent is able to do.

The Environment Model. In TROPOS and ADELFE, the environment model is elaborated in the requirements phase or workflow. The environment is the environment within which the system will operate. The model of TROPOS is described in terms of actors, their goals and interdependencies. Because ADELFE is interested in adaptive systems, the environment is a key notion. The model of ADELFE is described in terms of passive and active actors and sequence diagrams to explicit the kind of exchanges between the system and its environment. In AOR the environment appears through the notion of external agents. In MESSAGE, the domain model captures some entities of the system environment and the interactions with the environment are described for each role in terms of sensory inputs and acquaintances, resources ownership and accesses, and finally tasks and actions. In AAII, the relation between the agent and the environment is taken into account in the interaction model. But in general, the environment modeling is not a central point in existing methodologies. In GAIA and AUML there is no particular model of the environment.

The Identification of the Agents. In some applications such as simulations (prey-predators systems, ant colony simulation, RoboCup…), designers have no difficulty to define what an agent is in the system. In some other applications such as problem solving applications (planning management, business process management or equation solving, …) it is difficult to a priori know what will be an agent in the system. It is the reason why it is necessary to help the designer to identify the agents in an agent-oriented methodology.

In AUML and GAIA, the agents are already identified and the methodology provides nothing to realize this identification. The AOR and TROPOS methodologies do not provide guidelines to identify agents in the system to develop. AOR distinguishes artificial agents such as software agents or robots, human agents and institutional agents and in TROPOS the agents are found inside the actors' set.

In AAII, the elaboration and refinement of the agent model and the interaction model help the designer to define and refine agent classes and to give the multiplicity and the lifetime of agent instances. A fine-grained model of agency is identified from role, responsibilities and services.

The agent definition, which is given in MESSAGE and in ADELFE, defines the features that will be ascribed to the entities that the developer will choose to consider as agents.

In ADELFE, after the adequacy phase, the developer knows his system must be adaptive. So the agents must be found in the part of the system where adaptability is required. Note that the concept of agent in ADELFE is a specific type of agents with specific properties (see section 2.4).

8 CONCLUSION

The aim of this paper was to present the ADELFE methodology. ADELFE is an agent-oriented methodology suited to adaptive multi-agent systems and it follows the different steps proposed by the RUP as interpreted by Neptune [11]. We had focused on the main contributions of ADELFE concerning the kind of MAS we are interested in. This had been done through the description of the first three core workflows provided by the RUP and adapted to ADELFE using a case study presented by Bhattacherjee in [1]. Further in this paper we had compared ADELFE with other methodologies like AAII, AOR, AUML, GAIA, MESSAGE and TROPOS.

The main contributions of ADELFE can be summed up in four points:

– Adelfe can be used by a designer who is not specialized in building AMAS;
– it can take into account the high dynamic of the system environment through the application of the AMAS theory;
– it is able to tell him if an AMAS is required to build his system using the adequacy step;
– it helps him to find what components of his system need to be treated like agents belonging to the AMAS theory (cooperative agents) and guide him to build them.

Furthermore, we have some perspectives for ADELFE which will be able to provide some tools and libraries to ease the design and development of systems. For example, we think that it is better for a designer to have the ability of rapid prototyping to judge the validity of his architecture. We would like also to assist the designer if another methodology is more adequate to the system he wants to build.

ADELFE is going to be implemented within the OpenTool© software provided by TNI firm (http://www.tni.fr). OpenTool is a graphical tool like Rational Rose© which supports UML notation.

The authors would like to thank Valérie Camps (L3I), Jean-Paul Bodeveix, Thierry Millan and Christian Percebois (IRIT) for their remarks and their help regarding especially UML.

References

[1] A. Bhattacherjee - US WEST Global Village - http://coba.usf.edu/departments/isds/faculty/-abhatt/cases/USWest.pdf

[2] G. Caire, F. Leal, P. Chainho, R. Evans, F. Garijo, J. Gomez, G. Pavon, P. Kearney, J. Stark & P. Massonet - Agent Oriented Analysis using MESSAGE/UML - *AOSE* 2001.

[3] V. Camps, M-P. Gleizes, S. Trouilhet - Properties Analysis of a Learning Algorithm for Adaptive Systems – In *Int. Journal of Computing Anticipatory Systems*, Editions Chaos, Liège, Belgium, 1998.

[4] J. Castro, M. Kolp & J. Mylopoulos – A Requirements-driven Development Methodology – In *Proc. of the 13th Int. Conf. on Advanced Information Systems Engineering (CAiSE'01)*, Stafford, UK – June, 2001.

[5] Project P907-GI - MESSAGE: Methodology for Engineering Systems of Software Agents, - http://www.eurescom.de/~pub-deliverables/P900-series/P907/D1/P907D1.zip

[6] J. Ferber - Les Sytèmes Multi-Agents. Vers une Intelligence Collective - Eds InterEditions 1995

[7] M-P. Gleizes, V. Camps, P. Glize - A Theory of Emergent Computation Based on Cooperative Self-Organization for Adaptive Artificial Systems - *4th European Congress of Systems Science*, Valencia, 1999

[8] C.M. Iglesias, M. Garijo & J. C. Gonzalez - A Survey of Agent-Oriented Methodologies - In *Intelligent Agents V, ATAL'98*, LNAI 1555, Springer Verlag 1999.

[9] Jacobson, G. Booch & J. Rumbaugh – The Unified Software Development Process – Addison-Wesley, 1999.

[10] D. Kinny, M. Georgeff, & A. Rao - A Methodology and Modeling Technique for Systems of BDI Agents - In *Proc. of the 7th European Workshop on MAAMAW* (LNAI 1038), pp 56-71. Springer Verlag, Germany, 1996.

[11] Neptune : Reference IST Project n° 1999-20017 - Guidelines of a Process for the Use of (27/07/01) http://neptune.irit.fr

[12] J. Odell, H.V. Parunak, & B. Bauer - Representing Agent Interaction Protocols in UML – In *Agent-Oriented Software Engineering (AOSE'01)*, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp. 121-140, 2001

[13] P. Robertson, R. Laddaga & H. Shrobe - Introduction: the First International Workshop on Self-Adaptive Software - In *Proceedings of the 1st IWSAS* LNCS 1936, pp 1-10, 2000.

[14] Y. Shoham – Agent-Oriented Programming – in *Artificial Intelligence*, 60(1):51-92, 1993.

[15] G. Wagner - Agent-Oriented Analysis and Design of Organizational Information System. In Proc. of the *4th IEEE Int. Baltic Workshop on Databases and Information Systems*, Vilnius, May 2000.

[16] M. Wooldridge & P. Ciancarini - Agent-Oriented Software Engineering: the State of the Art - In P. Ciancarini & M. Wooldridg, ed., *AOSE 01*, Springer Verlag LNAI 1957, 2001.

[17] M. Wooldridge - On the Sources of Complexity in Agent Design - In *Applied Artificial Intelligence*, 14(7), pp 623-644, 2000.

[18] M. Wooldridge, N. R. Jennings & D. Kinny - A Methodology for Agent-Oriented Analysis and Design - In Proc. of the *3rd Int. Conf. on Autonomous Agents (Agents 99)*, pp 69-76, Seattle, WA, May 1999.

[19] E. Yu - Agent Orientation as a Modeling Paradigm - *Wirtschaftsinformatik*. 43(2) April 2001. pp. 123-132.

[20] F. Zambonelli, N. R. Jennings, and M. Wooldridge - Organisational Abstractions for the Analysis and Design of Multi-Agent Systems - In *AOSE'00* LNCS, Springer-Verlag, 2000