ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering

Carole Bernon*, Marie-Pierre Gleizes*, Sylvain Peyruqueou**, Gauthier Picard*

*Institut de Recherche en Informatique de Toulouse
118 route de Narbonne, 31062 Toulouse Cedex 4, France
{bernon, gleizes, picard}@irit.fr

**Artal Technologies
Parc Technologique du Canal, 31520 Ramonville Saint-Agne, France
peyruqueou@artal.fr

Abstract. Adaptive software is used in situations where either the environment is unpredictable or the system is open. This paper presents a methodology named ADELFE, which is led by the Rational Unified Process (RUP) but is devoted to software engineering of adaptive multi-agent systems. ADELFE guarantees that the software is developed according to the AMAS theory[1]. We focus this presentation on the additions of ADELFE regarding the three first core workflows of the RUP. Therefore, during the requirements phase, the environment of the studied system must be defined and characterized. Then, 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. Finally, the design workflow of ADELFE must provide the cooperative agent's model and helps the developer to define the local agents' behavior. We illustrate the methodology by applying it to a case study: a timetable design.

1. Introduction

Today applications are more and more complex and a possible solution to deal with this complexity is the use of agent-based or multi-agent based software. Usually this kind of software considers an environment where all is predictable. However, applications are more and more dealing with environments that are unpredictable and submitted to changes like the Internet or the real world in which robots can evolve. They require more robustness, more autonomy, more complexity; they require adaptive software. In [13], according to the DARPA definition, it is said "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". The multi-agent community generally considers the Multi-Agent Systems (MAS) as being adaptive because agents are autonomous, situated, pro-active, social… The condition for the system to adapt is to be composed of autonomous agents. In our point of view, this kind of adaptation is weak adaptation and all existing multi-agent systems are adaptive in this sense.

[1] The AMAS theory is developed and applied for the last 8 years at the Research Institute in Computer Science in Toulouse (IRIT). See http://www.irit.fr/SMAC

We call "strong adaptation" of a multi-agent system, the ability this system possesses to take into account unpredictable events in order to react to evolutionary environments in order to realize its "right" task. This new generation of MAS represents the next challenge in programming. To theoretically study MAS with strong adaptation capability, we developed an Adaptive MAS (AMAS) theory [5][8].

The aim of this paper is then to present a methodology, named ADELFE, able to guide and help a designer when he wants to build an adaptive multi-agent system based on the AMAS theory, presented in section 2. An important point is that the agents considered in this theory are not adaptive, only the whole system is. In the third section of this article, a brief overview of ADELFE is given. In the section 4, ADELFE is applied to a timetabling case study to present in a more detailed way the different steps of the methodology. Before concluding this document, the main strengths and limits of ADELFE and its contributions, with regard to other methodologies, are given in the section 5.

2. The AMAS Theory

The first aim of this theory is to realize MAS having the "classical" characteristics given in [7] to build a society of situated agents. But, from our point of view, such a MAS is also plunged into an environment and must reach a behavioral adequacy (by reproducing the behavior of a simulated society) or a functional one (by performing the right task, the task for which the system has been built). In our vision, the important notion is the collective; the AMAS theory must then lead to a coherent collective activity that realizes the right task.

We then proved the following theorem [8]: "*For any functionally adequate system, there is at least a cooperative interior medium system which fulfills an equivalent function in the same environment"*. Therefore, we focused on the design of cooperative interior medium systems in which agents are in cooperative interactions.

The specificity of the theory resides in the fact that we do not code the global function of the system within the agents. The global function of this system emerges from the collective behavior of the different agents composing it. Each agent possesses the ability of self-organization i.e. the capacity to locally rearrange its interactions with others depending on the individual task it has to solve. Changing the interactions between agents can indeed lead to a change at the global level and this induces the modification of the global function. This capacity of self-organization at the lowest level enables to change the global function without coding this modification at the upper level of the system. Self-organization is founded on the capacity an agent possesses to be locally "cooperative", this does not mean that it is always helping the other ones or that it is altruistic but only that it is able to recognize cooperation failures called "Non Cooperative Situations" (NCS, which could be related to exceptions in classical programs) and to treat them. The local treatment of NCS is a means to build a system that does the best it can when a difficulty is encountered. Such a difficulty is primarily due to the dynamical nature of the environment of the system, as well as the dynamics of the interactions between agents. More precisely an agent can detect three kinds of NCS:

- when a signal perceived from its environment is not understood and not read without ambiguity;

- when the information perceived does not induce the agent to an activity process;
- when concluding results lead to act in a useless way in the environment.

The agents, called cooperative agents, we are considering to build AMAS are composed of five parts contributing to their behavior: skills (what the agent is able to do), representations of the world (the knowledge it has about itself, about the others or about its environment), an interaction language (to communicate with others or with its environment), aptitudes (the capacities it possesses to reason on its knowledge) and a social attitude led by what we call cooperation.

Any designer needs a methodology to guide him when building a complex application. In the multi-agent domain, some methodologies exist [9][16] but few of them are able to deal with notions such as dynamics or evolving environment. This is the main reason why, from our point of view, a methodology suited for AMAS design is required. Furthermore, we want this methodology used by any designer and not only by those specialized in adaptive multi-agent systems because this would be a means to popularize this kind of programming.

3. A Short Overview of the ADELFE Methodology

This section highlights how ADELFE differs from object-oriented and other agent-oriented existing methodologies and where and how it handles the strong adaptation of the system to an evolutionary environment.

ADELFE[2] [1] is based on object-oriented methodologies, follows the Rational Unified Process (RUP) [10] and uses UML and AUML [12] notations. Some steps had been added in the classical workflows to be specific to adaptive MAS. It is not a general methodology such as GAIA [17] or TROPOS [6] but it has a niche which concerns applications that require adaptive multi-agent system design using the AMAS theory. ADELFE covers the entire process of software engineering like MESSAGE [4], PASSI [3] and TROPOS. In this paper, we only focus on the three first workflows, as shown in figure 1. At each workflow, the designer could backtrack to previous results to update or to complete them.

The **requirements workflow**, which is also taken into account in TROPOS, is a fundamental step in software engineering. In the AMAS theory, the adaptation process starts from the interactions between the system and its environment. Therefore, it is important to give a model of the environment during this workflow. The **environment model** consists in three steps: determining the actors, defining the context and characterising the environment.

In the **analysis workflow**, we added two steps to the RUP: the identification of the agents and the adequacy of the AMAS theory. ADELFE focuses on the **identification of the agents** like in AAII [11], MESSAGE and PASSI. In the previous workflow, the designer has identified the entities of the system, now ADELFE must help him to identify what entities will be agents. In ADELFE, the notion of agent is restrictive; we are only

[2] ADELFE is a national project started in December 2000 and supported by the French Ministry of the Economy, Finance and Industry. The different partners of this project are IRIT (University of Toulouse III) and L3I (University of La Rochelle) from academia and are ARTAL and TNI from industry.

interested in finding cooperative agents as described in the previous section. The designer has some guidelines: an agent is an entity previously defined, this entity may be faced with unexpected events and it may have evolutionary representations about itself, other agents or about its environment and/or evolutionary skills.

Because an adaptive MAS is not a technical solution for every application, ADELFE is the only methodology providing a tool to help a designer to decide if **the AMAS theory is adequate** to implement his application. For example, if the algorithm required to resolve the task is already known, if the task is not complex, if the system is closed or if no unexpected events can occur, this kind of programming is useless.

In the **design workflow**, the **agent model** and the **NCS model** are added to the RUP. The AAII methodology is dedicated to BDI agents, ADELFE is also dedicated to a specific architecture of agent: cooperative ones. Using the agent model, the designer must then describe the architecture of a cooperative agent by giving the components realising its behaviour. A MAS which is not in a cooperative interaction with its environment needs to adapt itself to it. But, according to the AMAS theory, the global function of the system is not coded, only the local behaviour of the agents composing the society is coded. The adaptation will then be managed by these agents through the NCS model. An agent which locally detects cooperative failures acts to change its interaction with others to remove this state. The NCS of an agent must be described by the designer, they depend on the application. To help him, ADELFE provides the designer with generic cooperative failures such as incomprehension, ambiguity, uselessness or conflict. ADELFE also provides tables with fields to fill up concerning the name of the NCS, its generic type, the state in which the agent must be to detect it, the conditions of its detection and what actions the agent must perform to treat it.

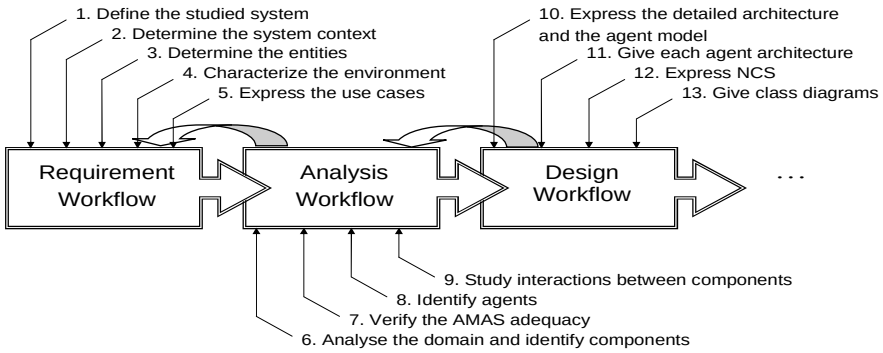

Figure 1. Overview of the three first core workflows of ADELFE.

4. Step-by-Step Details of the Methodology

In this section, we present the different steps of the ADELFE methodology. A case study about timetabling problem[3] illustrates the methodology at each step. The main actors of this case study are teachers and students who need to book several rooms to achieve theirs tasks. This case study has been chosen because it is based on a non-predictable environment and it is an open problem. Actually, some rooms could become unavailable because of some kind of problem or a teacher's availability could change and giving a solution to this problem means adapting in order to react to these dynamic changes of the environment. Furthermore, this problem can be extended to a diary management, for example, in which new people, and so new agents, must be added in the running system; this problem can be considered are being open.

4.1. Requirements Workflow

The aims of this stage are to define the system to be, to transform this view in a use-case model, and to organize and to manage the requirements (functional or not) and their priorities. In fact, at this stage, the designer has to define the function of the studied system and to model its environment.

Definition of the studied system. This definition is the result of the analysis of the requirements set artifact. The output of this step is a keyword set defining the system. From the user requirements, for the timetabling problem, the following keywords and concepts are highlighted: planning, rooms, teachers, students, constraints, organization and constraint managing. As an outlook, the problem can be viewed as the organization of teachers and students in rooms (and all the participants are subject to constraints). This problem belongs to the constraint satisfaction problems class.

Definition and Environment Modeling. A detailed definition of the environment of the system is necessary to develop adaptive systems, which are able to respond to any change. This step firstly focuses on what may be in interaction with the studied system in terms of passive or active entities, or constraints. In our example, teachers, students, the planning manager and the room manager are active entities because they are able to change by themselves their own constraints or they can interact with the system. Rooms are passive because they represent resources and they cannot modify their characteristics by themselves. The PPN (or National Pedagogic Plan) is the database that contains all the information concerning the courses (maximum number of sessions per week, hour quotas for each formation, etc): it is a passive entity.
In a second time, this step must define the context of the system. It requires a characterization of data flows and interactions between passive or active entities and the system. This step produces collaboration diagrams and sequence diagrams (entity/system or entity/entity). In our example, there are two kinds of data flows between the system and

[3] We elaborated this example as a case study to compare and discuss different methodologies and multi-agent platforms for the ASA Group of the Artificial Intelligence French Association.

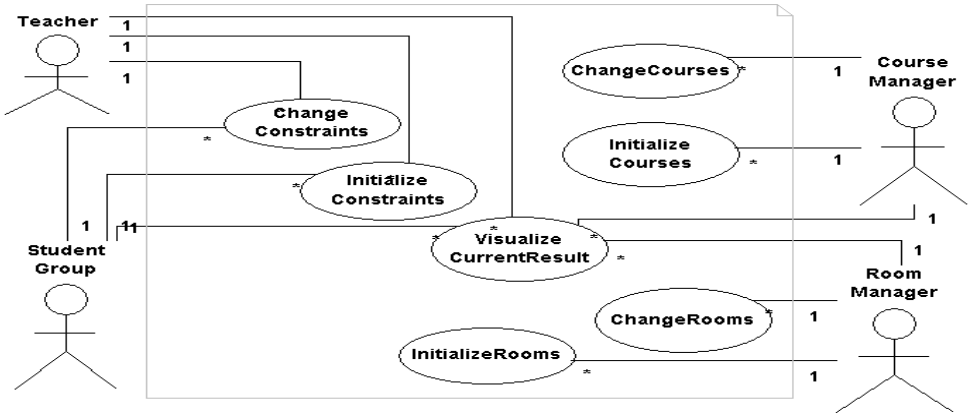

Figure 2. The use cases and their related actors for the timetabling problem.

passive entities: when the system consults the PPN and when the system consults room constraints. When an active entity wants to interact with the system, it may only have to change constraints (owner constraints or room constraints). In the other sense, the system interacts with the active entity by displaying the planning.

The third stress of this step is to characterize the environment according to the classification given by Russell in [14]. This characterization may enable the designer to detect some special use cases that aim to respond to the environment behavior. In the case study, we can characterize the environment of the system as following:

- Dynamic: the evolution of the active entities does not depend on the system, it is unpredictable from the point of view of the system;
- Accessible: the system can obtain information on the state of the environment;
- Non-deterministic: the system is not able to know what could be the effects of its actions on the active entities;
- Continuous: the number of interactions between the system and the entities is infinite.

Determination of the Use Cases. The main objective of this step, which ends the requirements workflow, is to clarify the different functionalities the system has to respond to. Only the active entities are implied in these use cases, which are the results of a functional requirements set. The use cases for the timetabling problem are shown in figure 2.

4.2. Analysis Workflow

From a multi-agent point of view, the identification of the agents must take place in this workflow. The analysis workflow has to develop an understanding of the system, its structure in terms of components and to know if the AMAS theory is required.

Domain Analysis and Architecture Study. 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. The result of this

step is a set of entities in preliminary class diagrams. `Teacher`, `CourseManager`, `StudentGroup`, `Room`, `RoomManager` and `PPN` classes appear naturally as real world entities. In a second time, we tried to determine what entities could be useful for our system. We propose a board to visualize the organization (the `Grid` and `Cell` classes) and the `ConstraintManager` class to control constraints for each entity that owns a `Constraint` class instance. Cells represent intersections of different dimensions (days, rooms, etc).

Adequacy of the AMAS Theory. This step aims to help the designer to decide if the AMAS theory is adequate to solve his problem because, for certain applications, this kind of programming can be useless. A software has been developed with several criteria to study the adequacy at two levels:

- At the global level to answer the question "is a system implementation using AMAS needed?"
- At the local level to try to answer the question "do some components need to be implemented as AMAS?" i.e. is some decomposition or recursion useful during design?

For the case study, the decision tool clearly suggests to use the AMAS to design the global level. Moreover, the tool indicates that some entities could be decomposed as AMAS. So, once the agents are identified, the designer has to reuse the method on them, as developed below.

Agent Identification. In this step, we are only interested in agents that enable a designer to build our sort of AMAS. The designer has to determine which entities fit with this agent type. This identification is done considering the notion of cooperative agent and the agent's characteristics such as autonomy, locality, requirement of interactions, individual goal to achieve, capacity of negotiation. Firstly, we have to know where a lot of evolution or dynamic is required. Then, for each entity identified during the domain analysis, we must examine if it has to face up to unpredictable events and has to treat Non Cooperative Situations. Teachers and students are autonomous, have local views, are plunged in the rooms and have to negotiate to find partners and to resolve resource problems. This kind of situations may create cooperation failures (NCS). At this stage, we identify teachers and students groups as being cooperative agents. All other entities are considered as objects.

Adequacy of the AMAS Theory at the Local Level. If the first step of adequacy to the AMAS theory indicates a possible decomposition, each agent has to be analyzed as a system. The goals of an agent, `Teacher` or `StudentGroup`, are to find different places and partners to follow or to give each course. These goals raise the problem of ubiquity. Agents cannot be at different place at different moments. Therefore, we propose to create one agent per course for each teacher or student group. Two agent levels are distinguished:

- `RepresentativeAgent` (RA): at the highest level, it represents a teacher or a student group within the system;

- `BookingAgent` (BA): at the lowest level, it is responsible for finding partners and booking rooms for a `RepresentativeAgent`. There are as many BAs as the number of courses a teacher has to give or a student group has to follow.

The identified agents have to be added to the preliminary class diagram as shown in fig. 3.

Study of 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 and at each level. As the RUP is use-case guided, for each use case, which has been defined between the system and the environment, a sequence diagram has to be defined to show the internal view and the interactions within the system. The interaction between agents can be written with AUML**.**

4.3. Design Workflow

In this section, we detail the design workflow in four steps. Because the complete design cannot be described in this paper, we only detail the steps which are not existing in other methodologies such as the agent model and the modeling of Non Cooperative Situations.

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. In our case study, four packages appear:

- Agent package, to manage BAs and RAs;
- Grid package, to manage the different dimensions of a grid and its cells;
- Constraint package, which has to be accessible to rooms and agents;
- Interface package, to enable a user to interact with the system.

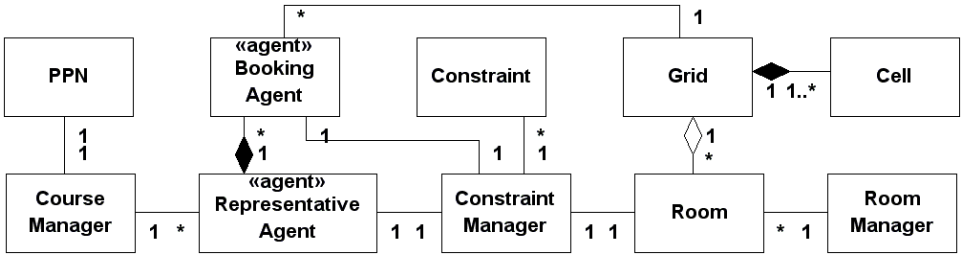

Figure 3. The revised preliminary class diagram for the timetabling problem gives us an idea of the MAS model.

Table 1. `BookingAgent` specification: skills, aptitudes and interaction languages are methods; representations are attributes.

BookingAgent	
Representations	
• constraints • bookState • partnershipState • brotherConstraints	• partners • recentlyMetAgents • RAFather • currentCell
Skills	
• moveInTheGrid • manageConstraints • manageBooking	• managePartnership • manageMessages
Aptitudes	
• bookARoom • cancelBooking • negotiateBooking • establishPartnership	• cancelPartnership • negotiatePartnership • SendMessage • InterpretMessage
Interaction Language	
• messageInteraction	• contactInteraction

Cooperative Agent Architecture. This step helps the designer to fill in a generic architecture given for an agent used in the AMAS theory. This architecture can be considered as a design pattern supplied by ADELFE in which an agent is composed of a skill model, representations models, an interaction language, aptitudes and a social attitude named cooperation. Following an object-oriented methodology, skills, representations or aptitudes possessed by an agent could be represented by methods. However, one of the main characteristics of the agents used in AMAS is the fact that sometimes they must be able to learn new skills, new representations or new aptitudes to adapt themselves to an evolutionary environment. Methods cannot change as the time goes by, so, if these characteristics are dynamic, they must adopt new representations: they must be AMAS themselves. In such a case, ADELFE recommends the designer to apply again the methodology to realize an adaptive multi-agent system to implement the skills, the representations or the aptitudes of an agent. The interaction language can be implemented by a set of classes or by a design pattern, which is closed to specific communication tools between agents like ACL implementation. Finally, the social attitude of an agent is guided by cooperation, it must be able to solve Non Cooperative Situations and a model is purposely defined in the following step. An example for a `BookingAgent` is given in the table 1. A BA is able to move in the grid, to meet other BAs and to negotiate partnerships. It can also send and receive messages and interpret them. To achieve its task, it must know the constraints of all the BAs working for its `RepresentativeAgent`. Constraints can be relaxed during a negotiation.

Table 2. The Booking Conflict NCS table of a `BookingAgent`.

Booking Conflict
State
Any
Description
The BA is in a cell that is interesting to book but this cell is already booked
Conditions
The BA is in a cell **AND** this latter is already booked **AND** yet the cell would be suitable if not booked
Actions
IF the cost of the new booking is less than the older one **THEN** the BA books the cell **ELSE** the BA moves elsewhere

Non Cooperative Situation Model. This step represents the main contribution of ADELFE to this workflow. During it, the designer must fill up a table describing each NCS encountered by each previously identified agent. This table contains the name of the NCS, the state in which the agent is when detecting it, the textual description of the NCS and the conditions and actions linked to it. The conditions describe the different elements that enable the agent to locally detect this NCS. The actions describe what the agent has to do to remove it. For instance, the NCS for a `BookingAgent` are:

- *Partnership incompetence*: the BA meets another BA that may be an uninteresting partner;
- *Booking incompetence*: the BA is in a cell that is uninteresting to book;
- *Message unproductiveness*: the BA receives a message that is not correctly addressed;
- *Partnership conflict*: the BA meets another BA that is interesting, but this other BA has already a partner;
- *Booking conflict*: the BA is in a cell that is interesting to book but this cell is already booked (shown in table 2);
- *Booking uselessness*: the BA meets its partner: they must separate to explore more efficiently the grid.

Even if all the behaviors of the cooperative agents are given (agent model + NCS model), the MAS as a whole can adapt itself because the interactions are not a priori coded. Changing the interactions between agents (self-organization) changes the global function of the system and, then, allows the "strong adaptation" of the MAS.

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 previous class diagrams. ADELFE does not need to provide additional functionalities to build them because the designer follows the same method as the one used in an object-oriented methodology**.** The figure 4 shows the final class diagram for the system.

4.4. Implementation and Tests Workflows

This workflow is similar to what is done in the RUP. However, to facilitate the work of the developer, we plan to provide a rapid prototyping tool that will be integrated in the

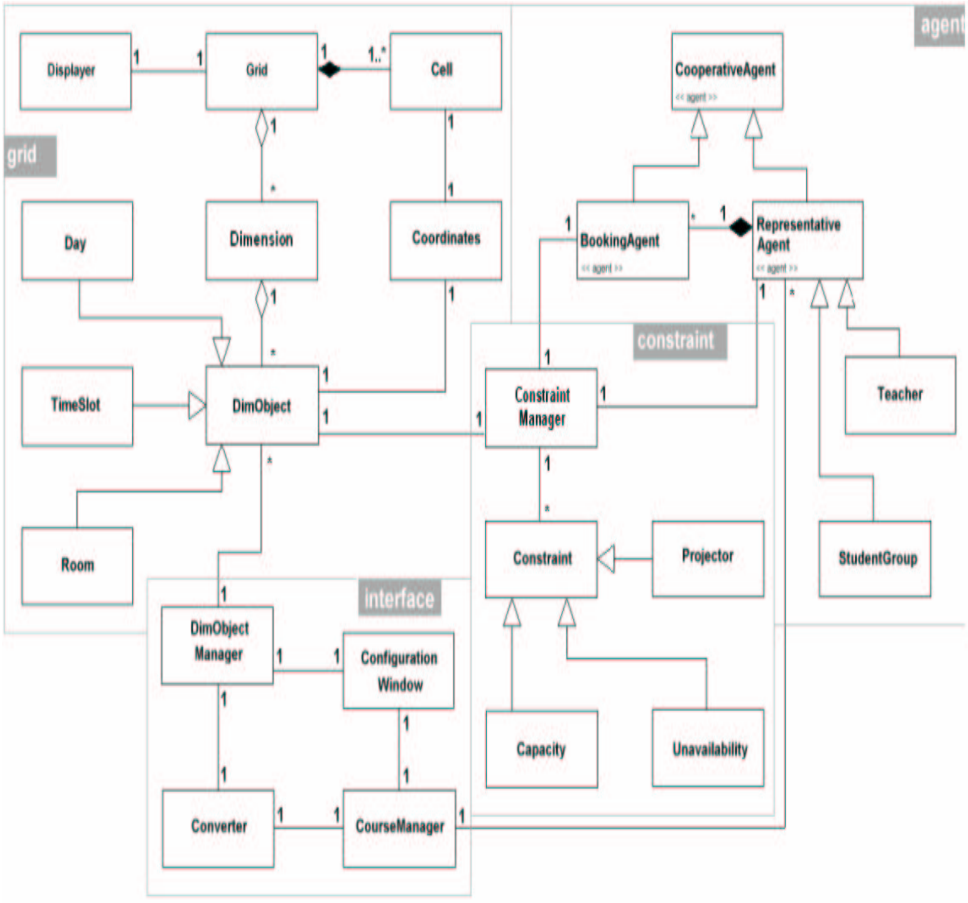

Figure 4. The final class diagram for the timetabling problem and the static inter package links between agents and objects.

OpenTool© software provided by our partner TNI (http://www.tni.fr). OpenTool is a graphical tool like Rational Rose, which supports the UML notation.

Concerning the timetabling problem, we have implemented the proposed architecture. Agents find adequate organizations and provide relevant results. At the moment, we only have tested the system on simple examples that appear in the requirements, but we hope to enlarge its application.

5. Contributions, Strengths and Limitations of ADELFE

TROPOS [6] expresses the dynamic and openness of the application in the requirements phases with the model of the environment and with particular soft goals. However, it does

not give guidelines to design 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 run-time [16]. However some improvements to GAIA are suggested in [18] to support applications in dynamic domains. Many other methodologies, like AAII [11], MaSE [15] or MESSAGE [4], do not focus on the dynamic aspect of the software environment and on the adaptation abilities of the software.

The specificity of ADELFE is to provide a methodology to design adaptive multi-agent systems coupled with a theory on those systems. The provided methodology covers the whole "classical" life cycle of software and reuses UML notations. These notions are already familiar to most of designers making this tool easily usable by developers not specialized in designing AMAS.

In adaptive multi-agent systems, the environment (in which the system is operating) is a key notion; but in a general way, the environment modeling is not a central point in existing methodologies. In DESIRE [2], the environment is taken into account at the agent level in the "world interaction management module": an agent maintains and interacts with its environment in the same way as with other agents. In MaSE, GAIA and AUML there is no particular model of the environment. In TROPOS, the environment model is described in terms of actors, their goals and interdependencies. 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.

For an industrial, it is very important to know very early if the system to develop justifies some investment in a new methodology or technique. Therefore, ADELFE guides the developer to make him decide if and where the adaptive multi-agent system technology is required in the system that he is developing. This explains the importance of the adequacy checking in the analysis stage.

ADELFE helps the designer to find what components of his system demand to be treated like agents belonging to the AMAS theory (cooperative agents). The specific agent architecture recommended by ADELFE can be viewed as one more design-pattern provided to the developer. Furthermore, ADELFE guides him to build agents; it provides a functionality to endow an agent behavior with the NCS model. This is because it is well known that the autonomous behavior of an agent which results from its perceptions, its knowledge and its beliefs is very difficult to define in complex systems as well as in dynamic systems. Actually, it is very difficult to enumerate all possible actions for each state of the environment. ADELFE is also a recursive or iterative methodology, when the developer has to implement an agent he must reuse the entire methodology to develop it.

ADELFE does not allow the design of every agent-based application. This drawback can be taken away by coupling another methodology (such as MESSAGE or PASSI) with ADELFE. If the adequacy phase tells that adaptive systems are not necessary, another methodology could be proposed. There is no automated tool for consistency checking of the workflows results but it is a future improvement.

6. Conclusion and Future

Few of the existing agent-oriented methodologies deal with concepts like dynamism or evolving environment. The aim of this paper was then to present the ADELFE methodology which is a multi-agent-oriented methodology suited to adaptive multi-agent systems based on the AMAS theory. ADELFE provides a new methodology to design a society of agents showing a coherent activity. It allows the society of agents to self-adapt to its environment. Till now, ADELFE has been used in two case studies : an Intranet system design [1] and a timetabling problem. In the former one, the requirements, the analysis and the design have been realized. In the later, the system is now implemented and operational. We plan to experiment more ADELFE in a European project (SYNAMEC) to develop a mechanical self-design system.

ADELFE is aimed to be a development toolkit of software with emerging functionalities and not only a "mere" methodology. Therefore, we have some perspectives for it, ADELFE 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;
- Assist the designer if another methodology is more adequate to the system he wants to build.

References

[1] C. Bernon, M-P. Gleizes, G. Picard & P. Glize – The ADELFE Methodology for an Intranet System – *Agent-Oriented Information Systems (AOIS'02) at CAiSE'02*, Toronto, May 2002.

[2] F. M. T. Brazier, P.A.T. Van Eck & J. Treur - Modelling a society of simple agents: From conceptual specification to experimentation - *Journal of Applied Intelligence*, 14:161–178, 2001.

[3] P. Burrafato & M. Cossentino – Designing a Multi-Agent Solution for a Bookstore with the PASSI Methodology – *AOIS'02 at CAiSE'02*, Toronto, May 2002.

[4] 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*.

[5] V. Camps, M-P. Gleizes, S. Trouilhet - Properties Analysis of a Learning Algorithm for Adaptive Systems – In *International Journal of Computing Anticipatory Systems*, Editions Chaos, Liège, Belgium, 1998. Available at http://www.irit.fr/SMAC.

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

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

[8] 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. Available at http://www.irit.fr/SMAC.

[9] 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.

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

[11] D. Kinny, M. Georgeff, & A. Rao - A Methodology and Modeling Technique for Systems of BDI Agents - In W. Van de Velde and J.W. Perram, editors, *Agents Breaking Away: Proceedings of the 7th European Workshop on Modeling Autonomous Agents in a Multi-Agent World* (LNAI 1038), pp 56-71. Springer Verlag, 1996.

[12] J. Odell, H.V. Parunak, & B. Bauer - Extending UML for Agents - In *Proceedings of the Agent Oriented Information Systems (AOIS) Workshop at the 17th National Conference on Artificial Intelligence (AAAI)*, 2000.

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

[14] S. Russel & P. Norvig - Artificial Intelligence: a Modern Approach - *Prentice-Hall*.

[15] C. H. Sparkman, S. A. Deloach, A. L. Self – Automated Derivation of Complex Agent Architectures from Analysis Specifications – *AOSE-2001*, Montreal, Canada, May 29th 2001.

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

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

[18] F. Zambonelli, N. R. Jennings, and M. Wooldridge - Organisational abstractions for the analysis and design of multi-agent systems - In P. Ciancarini and M. Wooldridge, eds., *AOSE'00*, LNCS, Springer-Verlag, 2000.