International Journal of Computer Applications (0975 – 888)
Volume 47– No.4, June 2012

An Agent-Oriented Information System: A Model Driven Approach

Abdelaziz EL Fazziki
Computer System engineering laboratory
Cadi Ayyad University, Morroco

Sana Nouzri
Computer System engineering laboratory
Cadi Ayyad University, Morroco

Mohamed Sadgal
Computer System engineering laboratory
Cadi Ayyad University, Morroco

ABSTRACT

To face the challenges of rapid enterprises environment change, enterprises need agile information systems that are flexible, reactive and adaptive. The process of mapping business requirements to the system functionalities involves several constraints that make the whole development process again very susceptible to errors. In this paper, we present a model-driven approach combined with software agent to develop an agile information system. In this work, we focus on the development of multi-agent systems (MAS) and a set of model transformation rules. Different ways of transforming a model into another exist. The choice of a target model differs according to quality criteria and is determined on the basis of specific requirements. The development process proposed is based separate aspects of systems, a BPMN meta-model, AML agent meta-model and a JADEX meta-model and the automated transformation rules with ATL language. The approach leading elements are: the meta-modeling, and mapping rules. Finally, we illustrate our proposals with a case study.

General Terms

Information system development, Model Driven Architecture, multiagent systems, business process modeling

Keywords

Multi-agent systems, business process, model driven architecture, BPMN language, transformation rules, AML language, JADEX platform.

1. INTRODUCTION

Improving of information systems engineering needs to integrate the organizational constraints and the constraints of flexibility and adaptability of systems.

The concept of business process has a major role in controlling the evolution of information systems and associated computer systems. Indeed, due to the diversity and evolution of information technology, the agility of the systems depends on the coherence between enterprise strategy, business process management and information systems [1] [2]. Therefore information system modelling requires an agile methodology for the enterprise domains structuring into a set of business processes. Generally a business process is a procedure which has an important role for the economic added value of an enterprise. More precisely a business process is a set of activities that are specific for the enterprise. The activities are target oriented and have a logical and temporal context. A better understanding of processes can be accomplished if they are represented by appropriate models. A model also contains further processes characteristics, involved actors, events, and documentations.

As almost organizations are reinventing themselves to meet the challenges of global competition and e-business, there is growing demands to develop and deploy new technologies that are adaptive, robust and reactive to rapid and unexpected change.

Agent Oriented methodologies [3] are emerging as a powerful new approach in software engineering. Concepts and techniques from the agent paradigm could well be the basis for the next generation of typical information systems. Agent concepts hold a great promise for responding to the new realities of new information systems generation. They offer higher abstractions level and mechanisms that address issues such as knowledge representation and reasoning, communication, coordination, collaboration among heterogeneous and autonomous parties, perception, commitments, goals, beliefs, intentions etc.

In model-driven architecture (MDA) [4] [5], IS developers must face the challenge of building IS solutions that are aligned with business processes in order to satisfy business requirements. To this endeavor, they must understand those business processes, which require extensive collaboration with domain experts and analysts. Because business processes are typically expressed using dedicated modeling notation language (BPMN) [6]. Thus, IS developers are required to master these particularities. In order to support these requirements in performing the business information system alignment, we define a mapping from domain model to BPMN models and from BPMN models to agent models and from agent models to a specific platform model. This approach allows the re-expressing business processes in a language that is closer to the IS developers.

The objective of this chapter is to describe an agent oriented Model-Driven approach. This approach is therefore put forward and based on a set of business models under continuous maintenance of business actors to reveal the current business needs, models being associated with adaptive agents that interpret the captured requirements to behave dynamically, always fulfilling current requirements. Consequently, the maintenance of the models is the maintenance of the actual software system. This provides a means of model-based adaptation rather than code-based adaptation.

In this paper, we define a mapping from domain model to BPMN model, from BPMN model to Agent model (AML) and from agent model to a specific platform model (JADEX).

The reminder of this paper is organized as follows: Section 2 presents the MDA concepts; section 3 describes an overview of the BPMN language. The subject of the section 4 is the development environment description. Section5 presents the description of the proposed approach. Section presents the proposed approach. Sections 6 and 7 define the main mapping rules. Section 8 describes the implementation of the transformation rules. Section 9 illustrates the proposed approach with a case study. Section10 presents an argumentation of the proposed approach. Finally, the section 8 concludes and presents the perspectives of this paper.

2. THE MODEL-DRIVEN ARCHITECTURE

Model Driven Architecture (MDA) [4], is software development approach, proposed and supported by the OMG. This is a particular variant of the model driven engineering (MDE). Its principle is to describe firstly the functionality of a system in a set of platform independent models (PIM: Platform Independent Models). Secondly to take into account implementation details of these features in a specific platform (PSM: platform specific Model).

According to the MDA approach, the process of software development is driven by hierarchical models in four levels according to a set of meta-models. The key feature of the MDA is the concept of model mapping [4]. A model transformation is a set of transformation rules and techniques of exploitation on a source model to attain a target model. In general, the transformation rules concern the mapping from a source model to a target model.

OMG defines three kinds of models that are at different abstraction levels. According to OMG, there are three types of model: the CIM (Computational Independent Model), the PIM (Platform Independent Model) and PSM (Platform Specific Model).

3. BUSINESS PROCESS MODELLING AND NOTATION

3.1. Description

The BPMN (Business Process Modelling Notation) [6] is a graphical notation for modeling business processes. Developed by the OMG (Object Management Group) its main purpose is to provide a unique notation and understandable by all stakeholders of the organization, and facilitates interactions between analysts, designers, and business developers to the technical managers who will implement and automate these processes.

The BPMN can be used to model the whole process or part of the process. Processes can be modeled at different fidelity levels. It is also suitable for internal business processes and for the public ones (in collaboration) [6]. Internal business processes are focused on one enterprise, and define the activities of its own activities and can define the interactions with external partners. Collaborative Public processes show the interactions between all organizations involved. These process models must be described from a general point of view, and should show the interactions between the various participants.

3.2. BPMN Meta-model

BPMN is a standard for business processes modeling; it provides a graphical notation for specifying processes in a business process diagram (BPD) [6] based on a flowcharting technique very similar to the UML activity diagram (UML) [6].

A process is divided into one or more (pools) corresponding to the participants. A pool can be divided into several lanes, representing the involved actors and organizational roles. Each lane contains a part of the process in the form of atomic or composite activities associated with actor in the same control domain and are connected by Sequence Flows and Message Flows. Sequence Flows describes the sequence in which activities must be completed while Message Flows describes the message exchange between pools. . The figure 1 presents the BPMN meta-model.

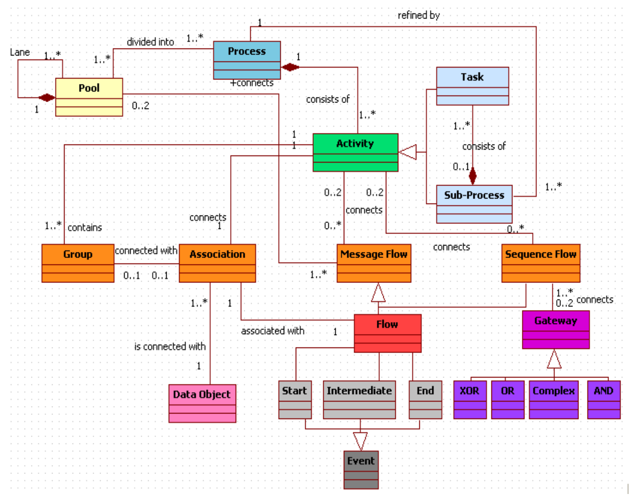


Figure 1. AML definition levels

4. THE DEDEVELOPMENT ENVIRONMENT

The environment development is composed with AML language modeling and Jadex platform.

4.1 AML language

The Agent Modeling Language (AML) [7] is a semi-formal language for modeling. It is an agent-specific extension to the generally used UML 2.0. It is designed to support business modeling, requirements specification analysis, and design of software systems based on software agent concepts and principles. The primary application context of AML is in systems explicitly designed to use agent concepts. However, it could also be applied to other domains such as business systems, social systems, robotics etc. In this section we will give an overview of AML packages and its elements as presented in figures below.

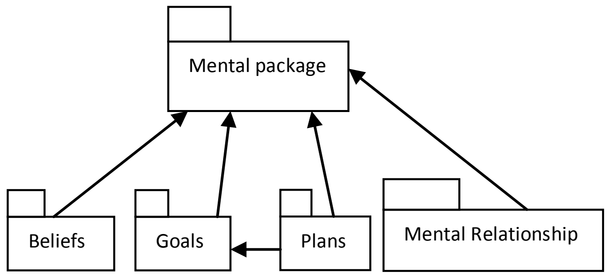


Figure 2. The structure of mental package

Mental States package defines fundamental meta-classes, that are used to specify meta-classes in other sub-packages. Beliefs, Goals, Plans sub-packages as their name denotes, define elements for capturing corresponding terms, as it can be seen in the agent structure. The Mental Relationship sub-package defines relations between mental elements to support reasoning processes.

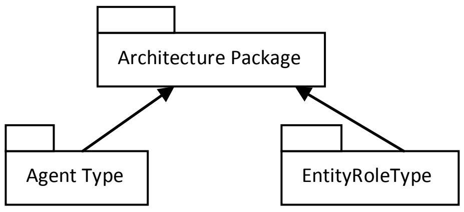


Figure 3. The structure of architecture package

The Architecture package defines the meta-classes used to model architectural aspects of multi-agent systems. These aspects are captured in more sub-packages like Agents, resources, environments, etc.

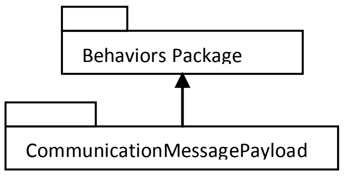


Figure 4. The structure of behaviours package

The Behaviors package defines the AML meta-classes used to model behavioral aspects, as behavior decomposition, mobility, and communicative interactions.

4.2 JADEX platform

Jadex is an Agent oriented reasoning engine based on BDI architecture. It can be used with different kinds of agent middleware that provides basic agent services, such as communication infrastructure or management facilities. Rational agents in Jadex have an explicit representation of their environment and objectives that they are trying to achieve. In this case rationality means that agent always performs the most promising step to achieve its objectives. In Jadex belief, goals and plans are first class objects that can be accessed inside an Agent [8]. The Development of agents consists of creating two types of files: plan implementations in Java programming language, and an XML file called the Agent Definition File (ADF) [8].

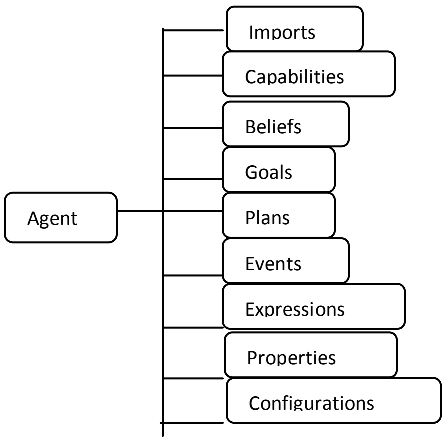


Figure 5. The structure of Agent Definition Files

5. MODELLING APPROACH

The main objective of this work is to propose a hybrid approach for information system development. This approach must take into account the agent technology concepts, the meta-modeling concepts, the principle of model driven architectures (MDA: Model Driven Architecture) and the dynamics of business processes. The aim of relying on such architecture is to separate the different system views and have a consistency with the proposed multi-faceted view. It is specifically based on the principle of the MDA to express how to transform, or match the concepts of the various models in order to have a global representation of the system. The MDA paradigm also covers the different phases of the development cycle by using transformation rules between the different modeling levels. The modeling approach proposed is based on the following steps:

- Definition of the modeling framework formed by the meta-models and transformation rules;
- Using the modeling framework by creating models based on meta-models defined and applying transformation rules between these models;

This approach allows the integration of various aspects of a multi-agent system and facilitates the modeling work through its process. It recommends following the order of meta-models using all their hierarchy starting with the domain modeling until the specific platform modeling.

5.1. An overview of the approach

The main objective of this approach is to allow the developer to create a framework that is specific to its application. This framework will contain:

- Domain modeling technique;
- BPMN modeling technique;
- Multi-agent classification technique;
- Multi-agent structuring technique;
- Transformation rules from the CIM models until a specific platform model;

5.2. The development process

The phases of the modeling approach to define business processes oriented agent are presented in the figure below. There are five phases, the first three define the different models of the system (phase modeling), while the last two (verification and optimization) to verify and optimize these models. The development process is an iterative process allowing incremental development and provides the rollback possibility to a previous phase due to the use of the MDA. This development framework is based on the following criteria:

- Concepts and specific properties of the domain,
- Concepts and specific properties of MAS;
- The transformation rules between domain concepts and agent concepts;
- Concepts and properties which are based on the specific platform;
- The transformation rules between the domain concepts, the business process concepts, and agent concepts to the target platform concepts. Figure 4 shows the different phases of the process.

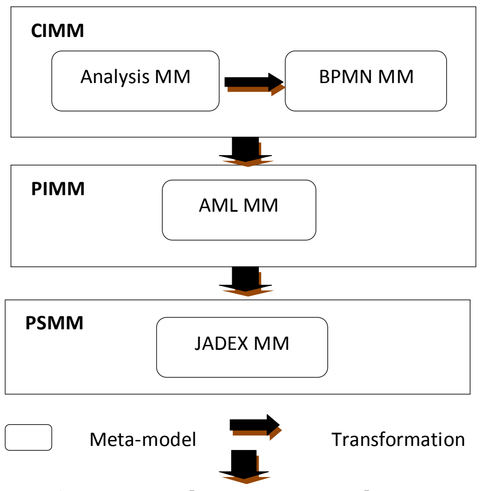


Figure 6. Development approach

In our approach, we propose several meta-models. We start by the analysis meta-model described by use cases and the BEC class diagram [7], the business meta-model based on the BPMN concepts to graphically represent business processes, the agent meta- model based on the AML [6] concepts and finally the JADEX meta-model JADEX [9] concepts.

5.3. The domain analysis.

Before starting the BPMN modeling, firstly, it is necessary to define the requirements for the review of each domain. To do this, we must:

- Identify enterprise domains,
- Identify the actors of each domain,
- Identify the functions associated with each actor,
- Define the corresponding use cases of each domain,
- Structure the domain use cases to define the corresponding business process.
- Associate to each process a multi-agent system.

Figure 7 defines the interpretation of the use cases by a classification of classes as follows:

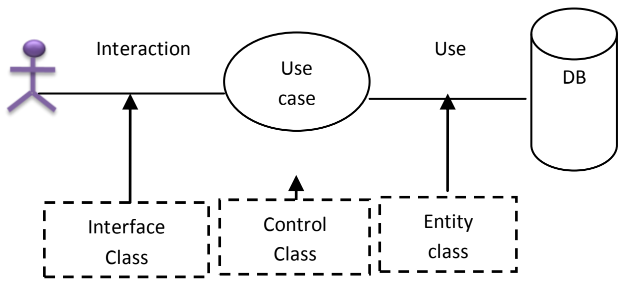


Figure 7. Relationship between use cases and analysis classes

5.4. Business Process Agent Modeling

We chose to represent a BP by a MAS, where the concept of actor is represented by an agent (on the basis of autonomy), the activities are modeled only in terms of goals and plans and are under the responsibility of an agent whose internal behavior is transparent. If multiple agents are in charge of an activity, protocols like social convention represent a standard framework of their collaboration. The BP agent modeling offers an agent classification into two categories: Intelligent Agents and reactive gents. The proposed approach is performed in two steps:

The first one, concerns the collect of requirements and domain analysis performed using UML2 which consists of defining the functional requirements, nonfunctional requirements and identifying the main artifacts that will constitute the solution. On these artifacts, it is a first estimate of the classes that realize the system. They are divided into three categories: class interface (or Boundary), the Controller classes and Entity classes (BEC Class Diagram). Every application must be made by the collaboration of these three kinds of classes.

- Interface classes: they represent the interface between an actor and the invoked use cases and the system.
- The control classes: They represent the process activities, that is means the management of interactions identified during the formulation of the requirements collect and analysis. These classes perform activities between interface classes and business classes.
- The business (entity) classes: These are the classes described in use cases that represent the data manipulated in the process. Often these classes correspond to persistent information: they are stored on permanent media such as databases.

After the formalization of requirements, all use cases, their relationships, involved actors, scenarios of each use case, interaction diagrams and the class diagram BEC (Boundary, Entity, Control) are developed.

The second step is the analysis stage, it consists of defining the activities, sequences, control flow and complete the class diagram BEC. This will be done using the modeling language BPMN.

For the multi-agent model development, it will be done in two stages. The first one by using a mapping of class diagram BEC and interaction diagram to the agent concepts. The second one will be done by using a second mapping from BPMN description of the concepts of agent's behaviors. The figure below illustrates this approach.

6. MAPPING BPMN TO AML

Generally the business process modeling defines a system global view. It can be used as a basis for the development of agents' behavior. This, by identifying for each agent: its plans, goals, interactions, statements and resources.

Normally, the derivation of the agents' behavior from a BPN diagram is highly dependent on the type of agents and how their behaviors are implemented in a specific platform. This derivation is essentially deduced from a set of rules based on the concepts of the agent meta- model and BPMN meta-model. It can be performed automatically or manually. Generally there are two types of pools: simple and structured (composed with lanes). Each agent behavior is derived from a simple pool or from a lane of a structured pool. These behaviors cover all the agent activities. In addition, use cases, interaction diagrams and BEC diagrams are used to identify agents with their types and their interactions. The mapping that we adopt is specified by a set of transformation rules, ensuring the gap from BPMN concepts to agent concepts. These rules are not exhaustive and are not detailed. This mapping is realized in two phases:

- The first use a mapping from the BEC class diagram and interactions diagram to the agent concepts. This mapping allows defining the system different agents. Figure 8 illustrates the different mapping levels.

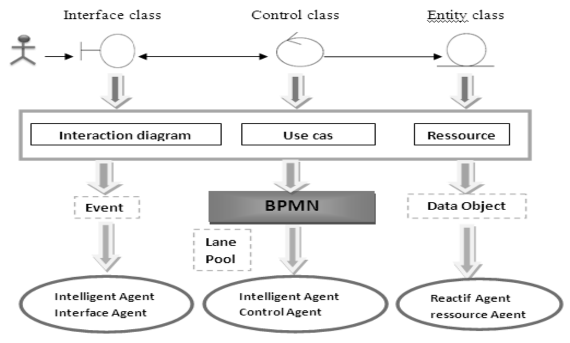


Figure 8. Different level of mapping

- The second use a second mapping from the BPMN description to the agent behaviors. The main transformation rules are presented in the following table:

Table 1.Mapping from BPMN concepts to AML concepts

BPMN concepts	AML concepts
Process	SMA
Pool	Group of agentType
Lane	AgentType
Sub process	Plan
Task	Action (Extention)
Intermediate event	CommunicationMessagePayload
End event	DecidableGoal
Data object	belief

- The third step, define the interactions among the different agents. These interactions are deduced from the sequence diagram, collaborative diagram and inter-pool communications.

7. MAPPING FROM AML TO JADEX

In this section we define the transformation rules between the AML packages elements and JADEX elements defined in the ADF file.

Table 2: Mapping from BPMN to AML

AML Concepts	JADEX Concepts
AgentType	Name of the ADF file
Belief of AgentType	Belief
DecidableGoal of AgentTyp	Achievegoal
Plan of AgentType	Plan
CommunicationMessagePay	Action (stereotype) of AgentType
StartEventMessage	MessageEvent
First plan of AgentType	InitialPlan sub-element of the configuration
Action of AgentType	Function implemented in the plan Java class

8. IMPLEMENTING TRANSFORMATION RULES

The specifications of transformation rules differ from one approach to another. Three approaches exist:

Approach by Programming: Consists in using object-oriented programming languages.

Approach by template: Consists in defining template of the desired target models.

Approach by Modeling: Consists in applying the MDA principles.

In this work we adopt the approach by modeling. The transformation rules are processed by the use of the Atlas Transformation Language (ATL) [10].

Before proceeding to the implementation of transformation rules, we must first, get an exploitable file of the BPMN model. To do this, we use the Intalio Designer [11] which allows generating an XML definition from the BPMN diagram.

To transform a BPMN diagram into an AML model, and an AML model into JADEX model, we have to insert this last one into the BPMN Meta-model, to allow it to become an instance of the MOF or ECORE based BPMN Meta-model [20]. This mapping includes several phases as follows:

8.1 BPMN injector

The BPMN injector is used to insert a BPMN notation into BPMN Meta-model, which produces an XMI or ECORE representation conforming to the BPMN Meta-model. This phase includes three steps: XML generator step, XML injector step and ATL transformation XML2BPMN step. Figure 9 describes this mapping.

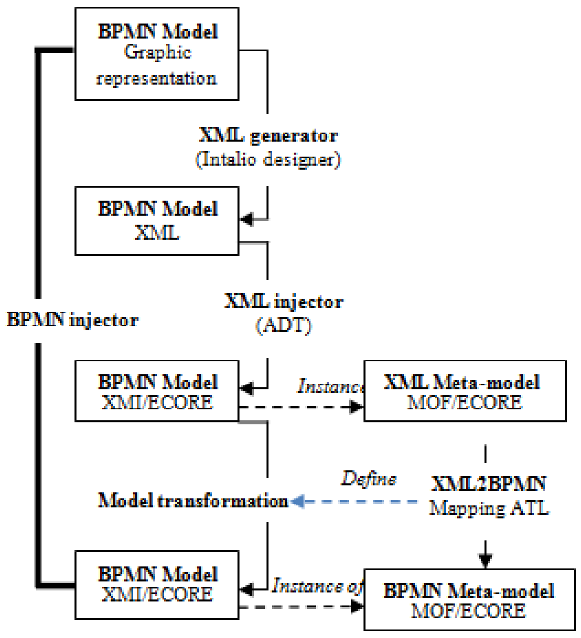


Figure 9. Mapping between BPMN notation model to BPMN model based Meta-model

8.2. ATL transformation BPMN2AML

The figure 10 illustrates the different steps of the mapping:

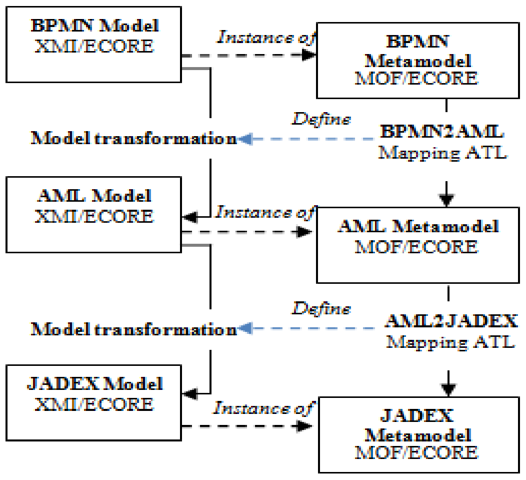


Figure 10. BPMN model to JADEX model

9. A CASE STUDY

In order to validate our work, we propose the modeling of the following case study: management of credit request, this example describes collaboration between a commercial, a financial analyst and a service contract to accept or reject the client request.

9.1. The use case diagram

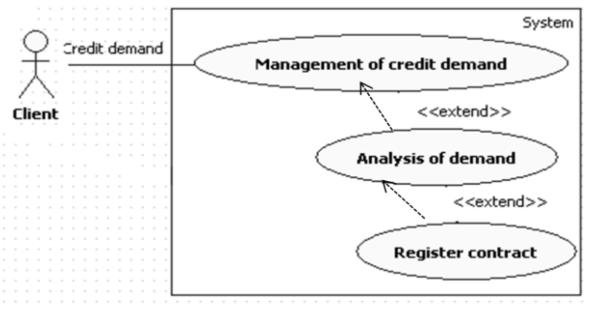


Figure 11. Use case diagram of credit demand

9.2. The BEC class diagram

From the BEC diagram, we deduce three kinds of classes (interface class, control class and entity class) that will allow us eventually to identify the three types of agent (interface agent, control agent and entity agent). Figure 12 presents the BEC diagram and the interaction between its different classes.

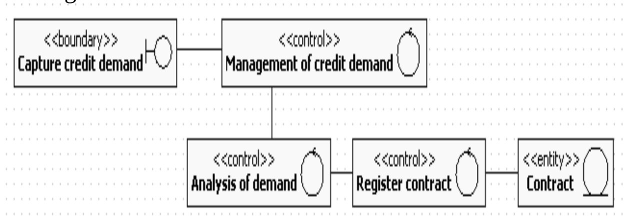

<<boundary>> Capture credit demand
<<control>> Management of credit demand
<<control>> Analysis of demand
<<control>> Register contract
<<entity>> Contract

Figure 12. The BEC class diagram of credit demand

9.3. The BPMN model

By using the BPMN language, we will describe the activities, events, actors, and messages communicated between actors for the credit request management process. Figure 13 illustrates this.

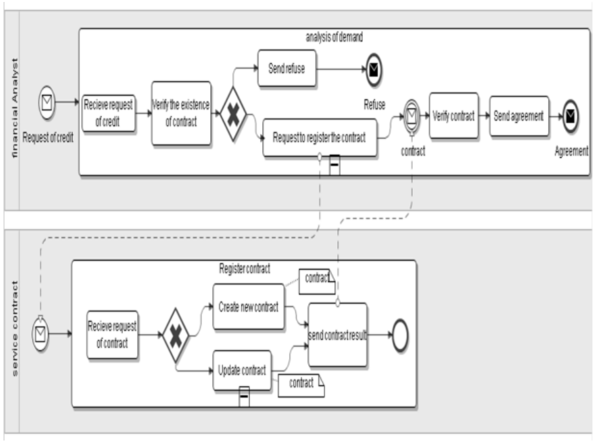


Figure 13. The BPMN model of credit request

9.4. The AML model

According to the BEC diagram, we can extract the different agents that would compose the system and their interactions.

9.4.1. The system agentification

According to the transformation rules:

Each boundary class becomes an interface agent;
Each control class becomes a controller agent;
Each entity class becomes a business agent.
From The BEC class diagram (figure 14), we obtain the system agents.

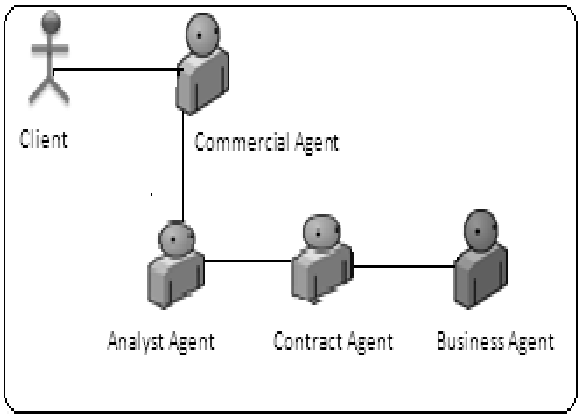


Figure 14. Agent in interaction

9.4.2. The agent structuring

In this section we just describe the Analyst agent structuring. Figure 15 illustrates this.

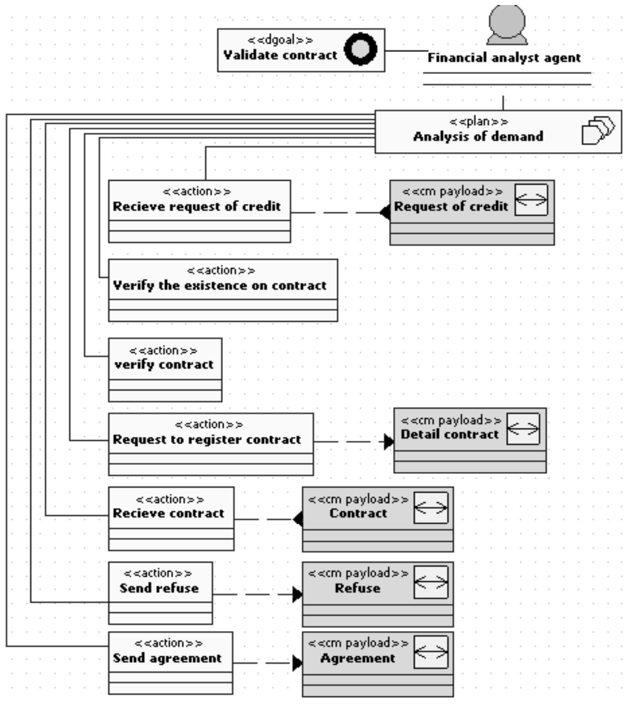

<<dgoal>> Validate contract
Financial analyst agent
<<plan>> Analysis of demand
<<action>> Recieve request of credit
<<cm payload>> Request of credit
<<action>> Verify the existence on contract
<<action>> verify contract
<<action>> Request to register contract
<<cm payload>> Detail contract
<<action>> Recieve contract
<<cm payload>> Contract
<<action>> Send refuse
<<cm payload>> Refuse
<<action>> Send agreement
<<cm payload>> Agreement

Figure 15. AML diagram of Analyst agent

9.4.3. The agent collaborative diagram

We complete the AML diagram by the collaboration diagram to describe the messages communicated between agents as shown in figure 16.

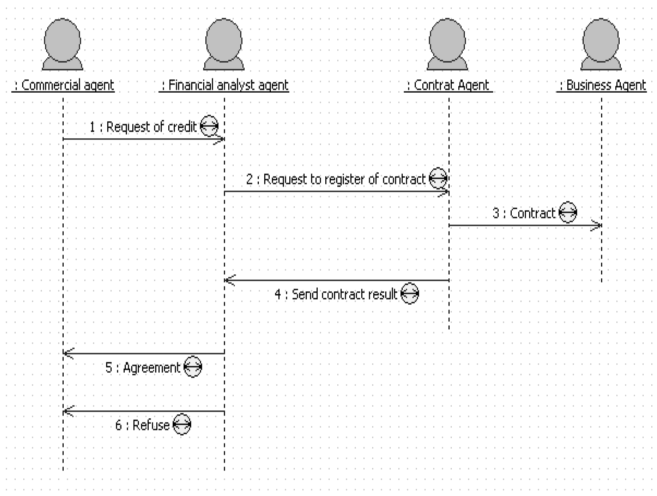
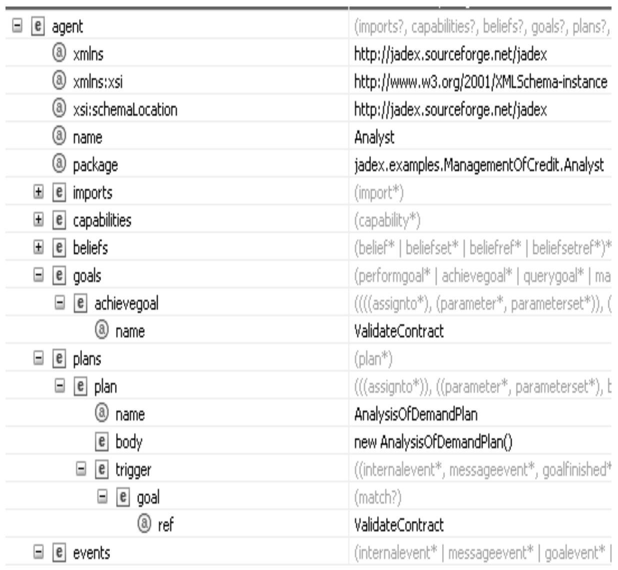


Figure 16. Figure Communicative sequence diagram

9.4.4. The specific platform (JADEX) model

The figure17 shows an overview of the results generated after applying the ATL transformation of the file AML2JADEX.atl, generating an xml file witch describes the concepts of the financial analyst agent according to the JADEX logic. This file is opened in the UML editor of Eclipse [8].



⊟ e agent	(imports?, capabilities?, beliefs?, goals?, plans?,
ⓐ xmlns	http://jadex.sourceforge.net/jadex
ⓐ xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
ⓐ xsi:schemaLocation	http://jadex.sourceforge.net/jadex
ⓐ name	Analyst
ⓐ package	jadex.examples.ManagementOfCredit.Analyst
⊞ e imports	(import*)
⊞ e capabilities	(capability*)
⊞ e beliefs	(belief* | beliefset* | beliefref* | beliefsetref*)*
⊟ e goals	(performgoal* | achievegoal* | querygoal* | ma
⊟ e achievegoal	((((assignto*), (parameter*, parameterset*)), (
ⓐ name	ValidateContract
⊟ e plans	(plan*)
⊟ e plan	(((assignto*)), ((parameter*, parameterset*), t
ⓐ name	AnalysisOfDemandPlan
e body	new AnalysisOfDemandPlan()
⊟ e trigger	((internalevent*, messageevent*, goalfinished*
⊟ e goal	(match?)
ⓐ ref	ValidateContract
⊟ e events	(internalevent* | messageevent* | goalevent* |

Figure 17. ADF file of financial analyst agent

10. THE APPROACH ARGUMENTATION

Why the approach is based on use cases, BEC diagrams, BPMN language, MDA and agents technology?

While use cases are considered more appropriate for modeling capabilities of information systems that will be used to control human activity and process modeling. The principle of communication and tasks within a process can be developed through a process modeling structure that does not take into account these aspects of communication and collaboration between stakeholders. This would then leave open the possibility of using a process-driven UML use cases to develop software support that will enable business process management and improvements, but this would require some relationship between process modeling and identification of needs in defining the functionality of a system. The approach taken here is to group the use cases corresponding to each actor to allow a relationship between the structures of a BPMN diagram and use cases diagram.

There is almost unanimous that Agent technology facilitates the development of complex, distributed and adaptive systems; however a number of major problems remain. One problem is that the developer has only a partial view of the system and the organization and not the whole context.

Currently, the Business Process Modeling Notation (BPMN) is considered as a response to this situation, and then it is suitable for modeling some aspects of multiagent systems. It is well with the modeling of agents' behavior and interactions especially at a high level of abstraction; the BPMN is still too rigid to model every detail of a process and for modeling the dynamic behavior. Therefore, we consider that the BPMN must be integrated into an agent-oriented development approach in order to exploit its strengths and overcome weaknesses. In this work, we have considered only the positive aspects of the BPMN, in particular the modeling of agents' behavior and their interactions [12].

The model driven architectures are based on model transformations to define reusable specific platform models. However, business needs change and cannot be fully and explicitly represented in such models for direct transformation into models of implementation. Thus, the use of agent-oriented MDA approach that uses a set of business process models that are constantly evolving, reflecting the current needs of the enterprise and are associated with adaptive agents that interpret the knowledge captured dynamically [13]. The main contributions of this approach may be:

- At the development level of agent-oriented information systems: a methodology for agile information systems development;
- At level of model driven architectures: a tool for modeling business processes of a high level of abstraction in order to align the systems implemented with their dynamic requirements;
- At level of distributed systems, interoperability of disparate components and services [13].

This approach respects the key property to achieving the agility of a system that has always been the clear separation of the different aspects of a system: The interfaces allow systems to communicate with users and with other systems, business rules supporting business processes and data management.

11. CONCLUSION

This paper provides the first steps towards a mapping from BPMN models to AML diagram. It presents an original attempt to associate the agent oriented paradigm, MDA, BPMN language and a part of UML language. The proposed methodology can make the development of complex systems better aligned with changing business needs easier and less costly. In this paper, we presented a definition of agent oriented approach which can be considered as a set of components which product a respond to development challenges that must solve the requirements of information systems development

Our main objective is the code generation from a JADEX platform model [9]. To this end, it seems to be necessary after defining a set of the principal transformation rules for the mapping between BPMN, AML language and JAdex platform, to solve enough problems, to get an adequate result. First of all, it is necessary to explore for the workflow and data object. Therefore we will also identify any further requirements that are needed on the agent technology in order to capture the desired functionality and behaviors.

12. ACKNOWLEDGMENTS

The authors wish to acknowledge the contributions of other members of the engineering laboratory of computer systems for their helpful discussions and the availability of all resources that have helped make this work in the best conditions. They would also like to thank the anonymous reviewers for their remarks and suggestions.

13. REFERENCES

[1] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Troops: An agent-oriented software development methodology. Technical Report DIT-02-0015, University of Trento, Department of Information and Communication Technology, 2002.

[2] Miller, J. & Mukerji, J. (2003). MDA Guide Version 1.0.1, OMG, Retrieved from <http://www.omg.org/cgi-bin/doc?omg/03-06-01>

[3] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Troops: An agent-oriented software development methodology. Technical Report DIT-02-0015, University of Trento, Department of Information and Communication Technology, 2002.

[4] Miller, J. & Mukerji, J. (2003). MDA Guide Version 1.0.1, OMG, Retrieved from <http://www.omg.org/cgi-bin/doc?omg/03-06-01>

[5] Selma AZAIEZ thèse d'université de Savoie, Approche dirigée par les modèles pour le développement de systèmes multi-agents, 2008 ;

[6] Muehlen, M., Indulska, M. (2010), "Modeling languages processes and business rules: A representational analysis", Journal of Information Systems, 35, 379–390.

[7] Paper by Whitestein Technologies. Agent Modeling Language, language Specification Version 0.9(2004-12-20).

[8] Eclipse Foundation, Eclipse - an open development platform, www.eclipse.org.

[9] Michele Piunti, Alma Mater Studiorum,2008, "Programming BDI agents in Jadex", Università di Bologna – DEIS

[10] K, Ravi and Sapkota, Brahmananda, "WSCDL to WSBPEL: A Case Study of ATL-based Transformation". In: 3rd International Workshop on Model Transformation with ATL, MtATL-2011, 1 July 2011, Zürich, Switzerland.

[11] Intalio designer www.intalio.com

[12] A. BRANDO, V. SILVA, and C. LUCENA. A model driven approach to develop multi-agent systems. Technical report, Departmento de Informtica - Pontifcia Universidade Catlica do Rio de Janeiro - PUC-Rio, 2005.

[13] A. El Fazziki, S. Nouzri, M Najib and M. Sadgal, "Une approche agents pour la modélisation des Processus Métiers", in 6ème Conférence francophone sur les architectures logicielles, May 30-31 2012, Montpelier France.