Towards an MDA-based multi-agent system

Sana Nouzri and Aziz El Fazziki*

Department of Computer Science,
Faculty of Science Semlalia,
Cadi Ayyad University,
BP 2390, Marrakech, Morocco
Email: sana.nouzri@gmail.com
Email: elfazzìki@ucam.ac.ma
*Corresponding author

Abstract: Most agent-oriented methodologies do not provide a development approach that ensures the transition from the design phase to the implementation phase. To certify the system consistency during the development process, and to avoid developers putting an extra effort to implement the system on a specific platform, we propose a new approach based on agent meta-model, business process (BP) meta-model, MAS-ML meta-model, Model-Driven Architecture (MDA) and Multi-Agent System (MAS). This approach defines different meta-models and models that instantiate the defined meta-models. This allows a mapping from the requirement model to the platform-specific model. To validate our proposal, we propose a case study in visual agent environment.

Keywords: MAS; model-driven architecture; MAS; multi-agent systems; business process; meta-modelling; MAS-ML; visual agent.

Reference to this paper should be made as follows: Nouzri, S. and El Fazziki, A. (XXXX) ‘Towards an MDA-based multi-agent system’, *Int. J. Reasoning-based Intelligent Systems*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Sana Nouzri is a member in the research laboratory (engineering information system) of the Department of Computer Science at the Faculty of Semlalia (Marrakech, Morocco) for the preparation of her PhD; she worked on agent-oriented approach directed by models for the development of information systems. Her current research interests include artificial intelligence, agent-oriented methodologies, the model-driven architecture, transformations and development agent-oriented languages.

Aziz El Fazziki is Professor of Computer Science at Marrakech University, where he has been since 1985. He received an MS from the University of Nancy (France) in 1985. He received his PhD in Computer Science from the University of Marrakech in 2002. His research interests are in software engineering, focusing on information system development. In the MDA arena, he has worked on agent-based systems, service-oriented systems and decisional systems. He is co-author of *Agent Based Image Processing*, and is the author of over 50 papers on software engineering.

This paper is a revised and expanded version of a paper entitled ‘Towards an MDA Based Multi-Agent Systems’ presented at the ‘3rd International Conference on Web and Information Technologies (ICWIT’10)’, Marrakech, Morocco, 16–19 June 2010.

1 Introduction

Generally, the agent-oriented software engineering is greatly concentrated on analysis, design methods and agent programming languages (Weiss, 2002). In particular, developers are required to put in a large effort to support the system development process or switch from one methodology to another to achieve the system development. Each methodology has its own meta-models and requires a specific implementation platform.

The specification of a particular system requires some flexibility where the developer can define semantics which most fits its application domain. On the other hand, to guarantee a development quality, the development environment should allow expression and validation of properties during the development phases.

Effectively, the Multi-Agent Systems (MASs) are complex and require a great number of interconnected components. In such context, it is necessary to be sure that the not functional and functional properties remain are kept intact during the development phases. The application of a modelling approach changes from one project to another and, therefore, from one situation to another; consequently, we prove the need to have adaptive components, which is the case of MASs.

The main objective of this work is to contribute towards improving MAS engineering. With this intention, we propose MAS development phase based on meta-modelling, Model-Driven Architecture (MDA) and MAS. It also proposes meta-modelling Business Processes (BPs) with their enhancement in the agent paradigm to make them more adaptive. That would allow applications to have two fundamental properties: flexibility in the design and execution of their BPs with the ability to federate existing heterogeneous applications. It allows dynamically managing processes, adapting them to the organisational level, accessing to heterogeneous data easily and responding to alerts on the running processes.

To realise this approach, we propose the use of visual agent as a software development environment which offers the necessary assistance from the specification phase until the implementation phase.

In the next section, we go into the related works. In Section 3, we formally define the description of our approach. In Section 4, we describe the BPs modelling, where we offer meta-model synthesising classical representations and the MAS meta-model. In Section 5, we present the transformation rules used for the mapping among meta-models. In Section 6, we try to illustrate our proposal with a case study E-business application. In Section 7, we present the application of transformation rules. Finally, Section 8 concludes this paper and presents our contributions and perspectives of this work.

2 Related work overview

Most of the multi-agent methodologies are not based on MDA, such as GAIA, PASSI, INGENIAS, ADELFE, TROPOS (Selma , 2007) and OADMAP (Jarraya, 2006). Most of these methodologies offer a single meta-model for MAS development. At structuring level, all these approaches are similar and are based on the same concepts: agent, environment, interactions and organisation. Their limitation is driven lack of the system implementation on a given platform.

A few of these methodologies are based on the MDA principles. In this context, we can mention few works:

- Cunha and colleagues (Cuhna et al., 2003) propose that the use of ontology of application and MICs are similar. Our approach is based on a domain meta-modelling and MAS meta-modelling.
- Thiefaine and colleagues (Brandão and Torres da Silva Carlos, 2005) presented an approach for developing MAS based on MDA, but this approach is very specific to the platform DIMA and too dependent on the structure of agent. In contrary, our approach has no restriction in the platform choice.
- Kazakov and colleagues (2002) have developed an MDA-based approach for the MAS designing, particularly for the mobile agents. This approach is dedicated to a very specific application domain. In contrary, our proposal is not specific to a particular application domain.

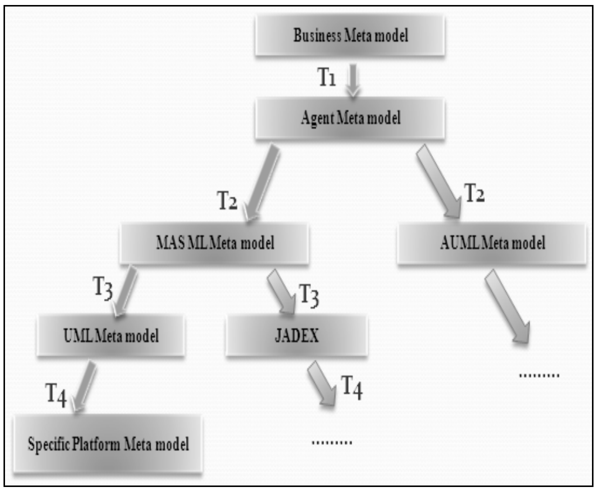
3 The proposed approach

Our approach proposes the adoption of meta-models definition, by using several transformation rules based on correspondence. The models are obtained by meta-models' instantiation. This approach is composed of the four abstraction levels which as defined in the MDA architecture are:

- CIM level (computation-independent model) represents the set of requirements in a particular domain; at this level, we try to define the BP meta-model and the MAS meta-model.
- PIM level (platform-independent model) represents the system requirements at the design phase; at this level, we present the MAS-ML meta-model independently from a specific platform.
- PSM level (platform-specific model) represents the MAS components in terms of platform development; this level is presented by the UML meta-model.
- Framework level infrastructure: represents the business and MAS components at the implementation level and the system deployment.

We subsequently present the different phases of meta-model that describe our approach. For example, we choose to use transformations (T_i) in the left as shown in Figure 1.

Figure 1 The used meta-models



4 The meta-models

4.1 CIMM level

In this level, we realise for the first time a business meta-model and multi-agent meta-model based on the BP meta-model. It is the BP meta-model that we use to analyse a case study. Then we deduce an MAS meta-model that matches our example. The chosen meta-models are just suggestions for our approach. Any user of our approach can define other meta-models suitable to his application domain.

4.1.1 Business process meta-model

The design of dynamic information systems becomes more complex, hence the need for the concept of process. Today, the modelling of BPs has an important role in the domain of information systems (Alter, 1999; Butler et al., 1999). In this paper, we propose a unified framework for modelling BPs. It is a detailed description of activities that actors should do. Figure 2 represents the meta-model built around the concepts of activity, actor and agent.

4.1.2 Multi-agent meta-model

Classes, interfaces, objects, inheritance, etc., are the concepts to express an object-oriented (OO) model. MAS uses the same concepts and considers other ones such as organisation, agent, perception, interaction, goal, task, etc. The meta-model that we use describes the entities that should be part of an MAS and their relationships. These entities meet the needs of our case study. We consider a set of concepts which we find sufficient for the development of our application (E-business). Figure 3 illustrates the MAS meta-model.

Figure 2 Business process meta-model

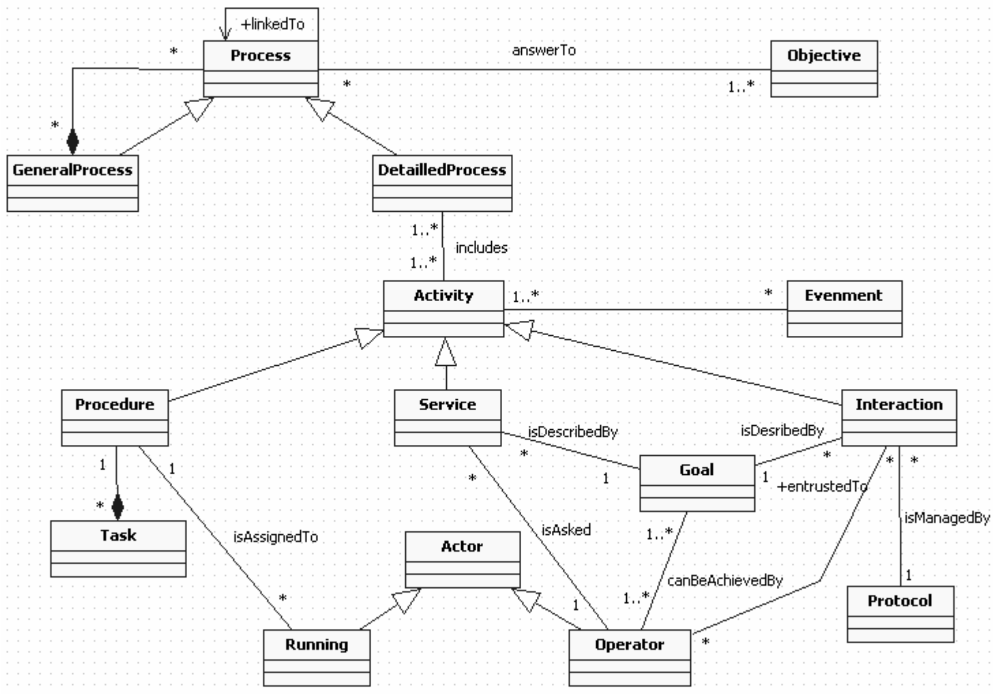
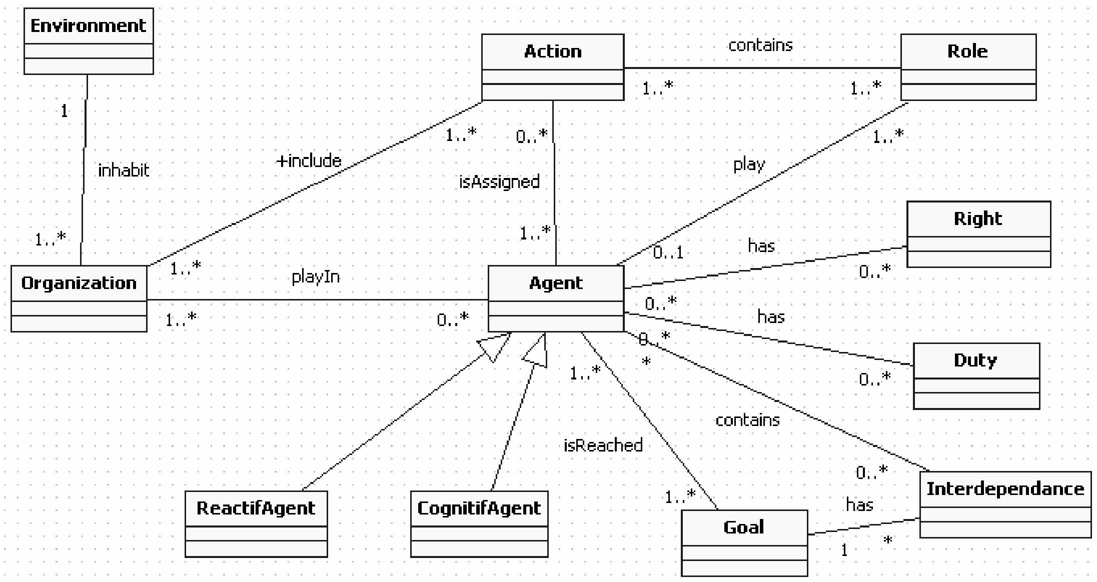


Figure 3 Multi-agent meta-model



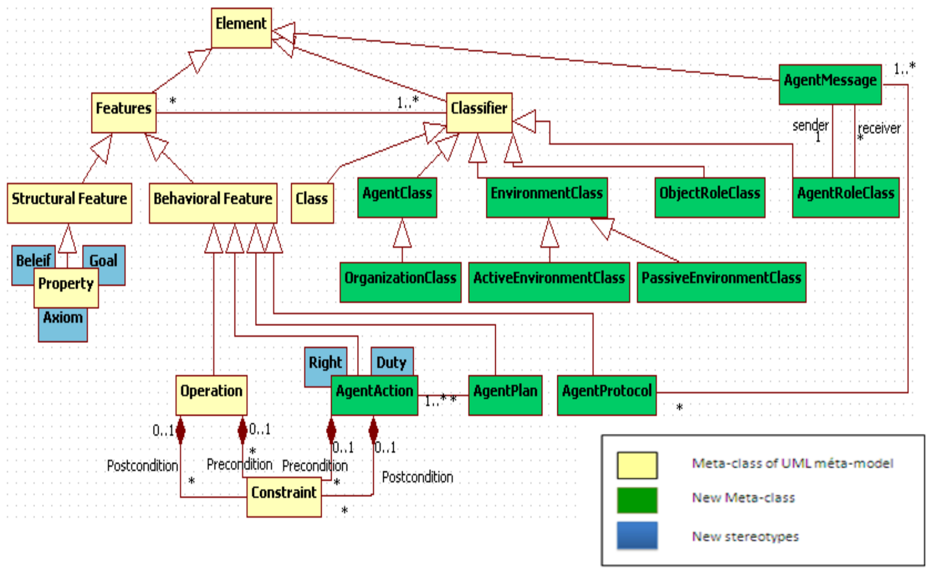
After the definition of these meta-models, which constitute the basis of CIM development cycle, it is time to define the PIM meta-model and the implementation platform. Several phases of transformation rules are necessary in our approach. The first one consists of preparing the correspondence between the concepts of meta-model agent and business.

4.2 PIMM level

MAS-ML was chosen to model MAS PIMM; it is not a platform-dependent modelling language. Although MAS-ML uses agent and OO abstractions, MAS-ML does not restrict the implementation of its models to a specific implementation platform (Alvez de Maria et al., 2005).

MAS-ML is a modelling language that makes additive extensions to UML to provide support for MAS modelling. New modelling capabilities were incorporated into UML, contributing to its evolution. The UML meta-model extensions proposed by the MAS-ML are based on the concepts defined in the TAO (Torres da Silva and Choren Noya, 2003).

Figure 4 MAS-ML meta-model representing the MAS-ML entities and properties (see online version for colours)



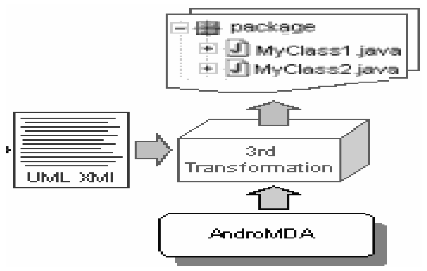
4.3 PSMM level

The ASF framework allows the implementation of MAS by using the OO technology. By using this framework, it is possible to implement the agents, roles, organisations and environments modelled in MAS-ML. All entities defined in the MAS-ML meta-model together with their properties (Figure 4) were mapped into OO classes defined in the ASF frameworks. Moreover, all MAS-ML relationships that link MAS-ML entities were mapped into UML relationships to link the OO classes defined in ASF (Alvez de Maria et al., 2005).

4.4 Implementation platform (AndroMDA)

The development tool that we have chosen offers Java code generation. This generation is based on the UML XMI generated by the tool. This functionality was developed by using the AndroMDA (Figure 5).

Figure 5 Transforming by using AndroMDA



5 Transformation rules

5.1 First transformation (mapping between agent and business process meta-models)

We chose to represent the BP meta-model presented earlier by the agent paradigm. The final MAS meta-model is obtained from the correspondence between the business meta-model and agent meta-model. So, a BP is represented by MAS where the concept of actor is represented by an agent. Activities are modelled to achieve the goal and are subject to a control by an agent. If several agents contribute to achieve a goal, communication protocols are a standard part of this collaboration. Table 1 shows this correspondence.

Table 1 Mapping between agent and business concept meta-models

Business entity	*Agent entity*
Activity (procedure)	Action
Event	Action
Activity (service)	Duty right
Running	Reactive agent, Organisation
Operator	Cognitive agent
Interaction (protocol)	Interdependence
Goal	Goal

At this level, we are yet far from having a PSM. It is now time to model the concepts of final MAS meta-model using the MAS-ML language offered by the visual agent.

5.2 Second transformation (from CIMMs into PIMMs)

To ensure the passage from CIMM to PIMM, we used the static and dynamic aspects of TAO meta-model that provides an ontology-based agent and object software engineering (Silva and Lucena, 2003). These aspects are related to the characteristics of MAS-ML which allows us to model the concepts of final meta-model using the MAS-ML language at PIMM level.

Based on the following TAO static aspects, we do the correspondence between the concepts of final meta-model and MAS-ML concepts. Table 2 shows this correspondence:

- *Object*: an object is a passive element in the domain whose instances have state and behaviour. An object may evolve from state to state. However, it has no control over its behaviour.
- *Agent*: an agent is an autonomous, adaptive and interactive element in the domain whose instances are expressed through mental components such as beliefs, goals, plans and actions.
- *Organisation*: an organisation is an element in the domain whose instances group agents, objects and other organisations (sub-organisations). An organisation has goals, beliefs (as agents) and axioms.
- *Object role*: an object role guides and restricts the behaviour of an object through the description of a set of features that are viewed by other elements.
- *Agent role*: an agent role guides and restricts the behaviour of an agent through the description of a set of goals, beliefs and actions. An agent role defines duties, rights and protocols that restrict an agent instance.
- *Environment*: an environment is an element in the domain whose instances are the habitat for agents, objects and organisations. An environment can be passive as an object or active as an agent.
- *Event*: an event is an element whose instances are generated by other elements in the domain. An event can be generated by objects through the execution of their operations, by agents through the execution of their actions and by the environment when it is an active element (Torres da Silva and Choren Noya, 2003).

The TAO meta-model defines relationships between concepts. These are Inhabit, Ownership, Play, Specialisation, Control, Dependency, Association and Aggregation.

5.3 Third transformation (PIMMs into PSMMs)

The MAS-ML models should be transformed into UML models in two phases:

- The first phase consists of describing MAS-ML models into MAS-ML XMI file created using the MAS-ML DTD.
- The second phase consists of transforming MAS-ML XMI to UML XMI, according to the ASF framework (Alvez de Maria et al., 2005; Table 3).

Table 2 Mapping between the entities of final meta-model and TAO meta-model

MAS entity	*TAO static aspect*
Organisation, Reactive agent	Organisation
Environment	Environment
Process	Plan of organisation
Objective	Goal of organisation
Activity (procedure)	Action of organisation
Activity (service)	Duty, right
Interaction (protocol)	Protocol of agent role
Task	Object
Cognitive agent	Agent
Goal	Goal of agent
Event	Action of agent
Role	Agent role

Table 3 Rules used to map MAS-ML XMI into UML XMI (Alvez de Maria et al., 2005)

Entity	*Transformation rules*
Agent	1. Create a concrete class that will represent the agent and will extend the abstract class *Agent*.
	2. Create concrete classes that extend *Plan* and *Action* classes to implement the behaviour defined by plans and actions.
	3. Implement the constructor method of the concrete class created in 1 to instantiate goals, beliefs, plans and actions according to the agent properties. The creation of beliefs and goals is realised by the instantiations of *Belief* and *Goal* classes, respectively.
Organisation	1. Create a concrete class that will represent the organisation. This class will extend the abstract class *Organisation* or *MainOrganisation*, according to its definition.
	2. Create concrete classes that extend *Plan* and *Action* classes to implement the behaviour defined by plans and actions.
	3. Implement the constructor method of the concrete class created in 1 to instantiate goals, beliefs, axioms, plans and actions according to the agent properties. The creation of beliefs, goals and axioms is realised by the instantiation of *Beliefs*, *Goal* and *Axiom* classes, respectively.
Agent Role	1. Create a concrete class that will represent the agent role. This class will extend the abstract class *AgentRole*.
	2. Create concrete classes to represent protocols and that extend the abstract class *Protocol*.
	3. Implement the constructor method of the concrete class created in 1 to instantiate rights and duties according to the agent role properties. The creation of rights and duties is realised by the instantiation of *Right* and *Duty* classes, respectively.
Object Role	1. Create a concrete class that will represent the object role. This class will extend the abstract class *ObjectRole*.
Environment	1. Create a concrete class that will represent the object role. This class will extend the abstract class *Environment*.

6 Models of a sample application instantiated by meta-models

6.1 Modelling example

An electronic commerce example is an appropriate example of an MAS. The virtual marketplace is composed of a main-market where users are able to negotiate any type of item. In addition, the main-market defines two market types that negotiate items with particular characteristics. The markets of special goods negotiate expensive high-quality items, and the markets of used goods negotiate low-priced low-quality items. Users can buy items in the main-market in markets of special goods or in markets of used goods. The users can also sell their items in the used goods. In the main-market and in markets of special goods, users buy the items available in the market.

In the main-market and in markets of special goods, the user looks for a seller and sends it a description of the desired item. The seller, created by the market to negotiate with the buyer, is responsible for verifying if there is an item with the same characteristics in the environment (the virtual marketplace). The environment stores all the items to be sold in these markets. If the item is found, the seller negotiates with the buyer.

In markets of used goods, the sellers and the buyers are users. The users who want to sell items announce them, and the users who want to buy items look for announcements in the market. If the buyer finds the desired item, it starts a negotiation with the seller (Torres da Silva and Choren Noya, 2003).

6.2 Architecture application

The example shows a main-organisation (general store) and two sub-organisations (market of expensive products and market of used products). Two types of agents are modelled in this system (user agent and store agent). Two roles of agents (buyer and seller) are defined in each organisation, played by user agent and store agent, respectively.

The environment (virtual market) will be modelled as a passive element and is represented as a package that includes all the entities that inhabit it.

Figure 6 shows the overall architecture (MAS1) that is composed of two subsystems (MAS2 and MAS3). Each system MAS represents an organisation and defines a process for its operation, and the architecture defines the relationships between these systems.

6.3 CIM level

After defining the sample application, we define at the first time all the BPs involved in the operation of the system.

The operating system is realised by three BPs, a general process of the general system and two processes inherited from the latter, for specific subsystems defined previously (Figure 7).

Figure 6 System architecture (see online version for colours)

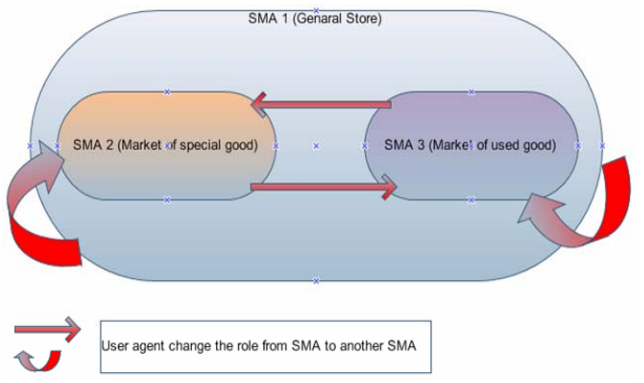
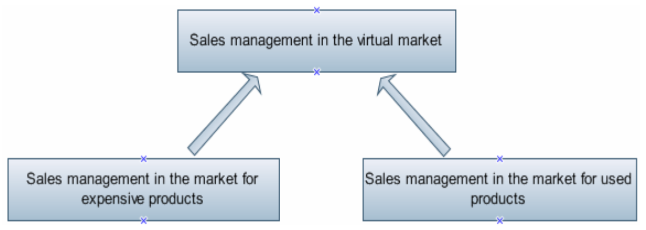

SMA 1 (Genaral Store)
SMA 2 (Market of special good)
SMA 3 (Market of used good)
User agent change the role from SMA to another SMA

Figure 7 Application business process (see online version for colours)


Sales management in the virtual market
Sales management in the market for expensive products
Sales management in the market for used products

In this paper, we focus on modelling the BP 'sales management in the market for expensive products'. We present three types of activities that are part of the BP: procedure, service and interaction. The main objective of the virtual market is the management of sellers and buyers in the virtual marketplace. We detail the types of activities performed by different actors in the following model.

6.4 Business process model

To present the activities of each agent (reactive agent and cognitive agent), we use the activity diagram that allows us to define the activities of the process 'sales management process in the market for expensive products'.

Figure 10 presents the imported bookstore which plays the role of a market of special goods. The goals of the imported bookstore are to manage the inclusion of new buyers in the market and to manage the creation of sellers. To achieve its goals, it defines activities (a) to negotiate the entrance of a new buyer, checking if the buyer has the needed characteristics (according to the protocol that defines the interaction between itself and a buyer), (b) to register the new buyer (according to the protocol that defines the interaction between itself and a buyer of imported books) and (c) to create the seller of imported books (according to a protocol that defines the interaction between itself and a seller of imported books). It also defines an activity to receive sales information (according to the protocols defined between itself and its sellers and according to the axiom defined in the market of special goods). Then, it sends information to the main-organisation by following the duties specified in the role. The beliefs of the imported goods store are its buyers, sellers, sales and the main-organisation.

The buyer defined in the main-organisation defines the ‘simple negotiation’ protocol with the seller and the ‘entering organisation’ protocol with both the market of special goods and market of used goods. The buyer of imported books specifies (a) the ‘registration’ protocol with the markets of special goods, (b) the ‘searching for seller’ protocol with the market of special goods and (c) the ‘simple negotiation’ protocol with the seller of imported books.

The seller of imported books also defines the ‘simple negotiation’ protocol. The seller of imported books defines the ‘simple negotiation’ protocol with the buyer of imported books and the ‘register sale’ protocol with the market of special goods (Figure 8).

Figure 8 Activity diagram of business process activities (see online version for colours)

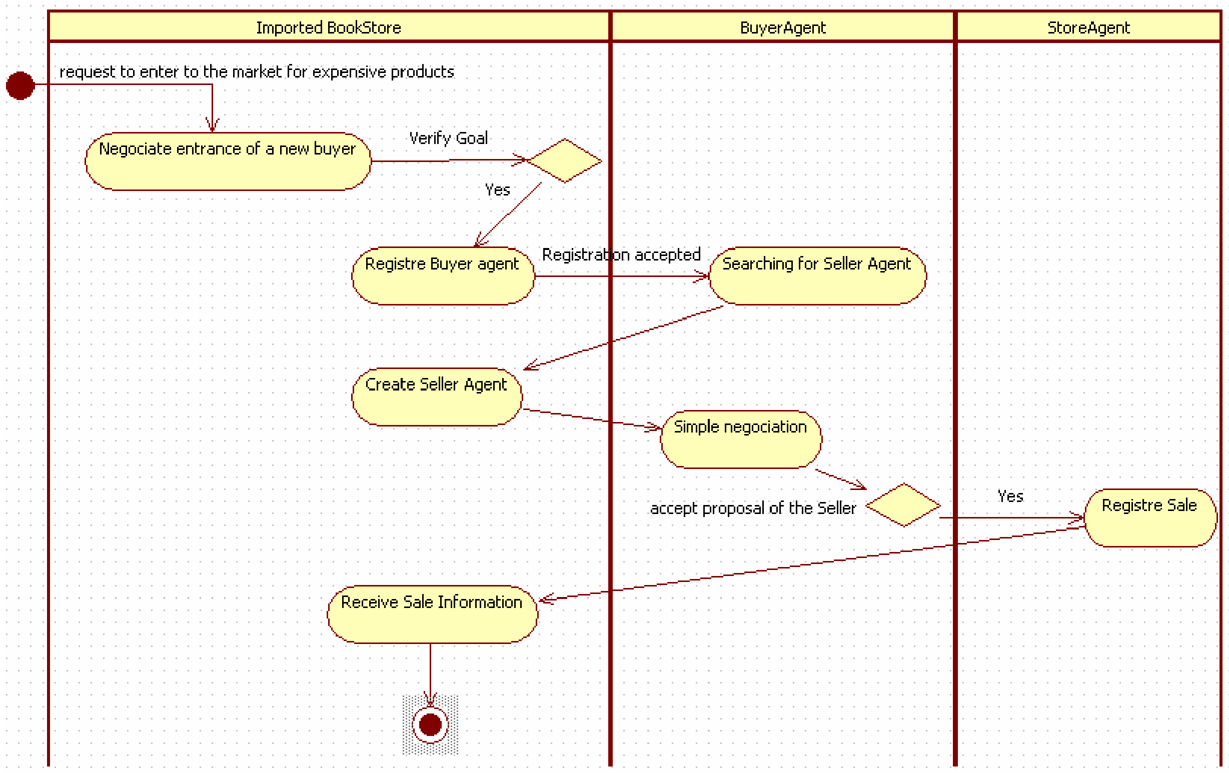
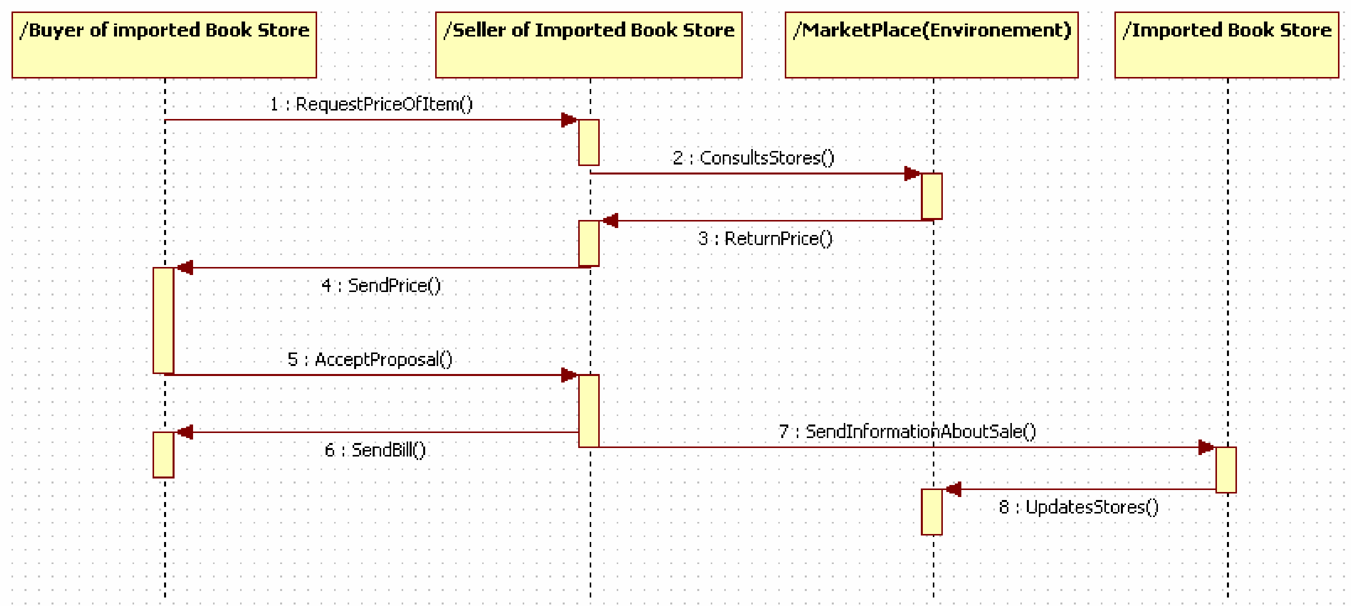


Figure 9 Sequence diagram of protocol ‘simple negotiation’ (see online version for colours)



The protocol 'simple negotiation' defines that a buyer asks a seller the price of an item. After consulting the environment, the seller sends the price to the buyer and the buyer can accept or reject the seller proposal. The choice of accepting or rejecting a given seller proposal is one of the buyer's role *rights*. If the buyer accepts the seller's proposal, the seller sends the bill to the buyer. Then, the seller sends the information about the sale to the main-organisation, as specified in its axioms (Figure 9).

6.5 Agent model

The treatment of the sample application allows us to instantiate the agent meta-model; we present instances of action performed by different agents in the organisation and also instances of roles associated with agents, rights and duties of agent in Table 4.

6.6 PIM level

After describing the elements that must be defined to create each diagram MAS-ML, we start with the static aspects of an application that must be modelled using three static diagrams proposed by MAS-ML (organisation, role and class diagram). An organisation diagram models an environment, an organisation, its sub-organisations, the roles defined in the organisation and the elements that play those roles.

The role diagram complements the organisation diagram by modelling the relationships between object roles and agent roles. A class diagram models the relationships between objects, agents, organisations and environments. The dynamic aspects of an application are modelled by using the dynamic sequence diagram that models the interactions between agents playing roles, organisations playing roles, environments and objects while either playing roles or not (Torres da Silva and Choren Noya, 2003).

Table 4 Instance of entities agent meta-model (see online version for colours)

Cognitif Agent (Organization)	Imported Book Store
Reactif Agent	Buyer Agent
	Seller Agent
Role	Buyer of Imported BookStore
	Seller of Imported BookStore
Duty of Seller Agent	Send bill to Buyer Agent
	Send information about sale
Right of Buyer Agent	accept or reject the seller proposal
interdependance	simple negociation
Goal of Buyer Agent	bought item
Goal of Seller Agent	Sell item
Goal of Organization	Management of seller and buyer
Environment	Virtual Market Place
Interdependance Of Buyer Agent	Searching for Seller agent
	Simple negociation
Action of Buyer Agent	Request to enter the market for
	expensive products
Interdependance of Seller Agent	Simple negociation
Action of Organization	Negociate entrance of a new buyer
	Registre Buyer Agent
	Create Seller Agent
	Receive sale information

We briefly present the organisation diagrams of the systems (MAS1) and (MAS3) (Figures 10 and 11).

Figure 10 MAS1 organisation diagram (MAS-ML model) (Torres da Silva and Choren Noya, 2003) (see online version for colours)

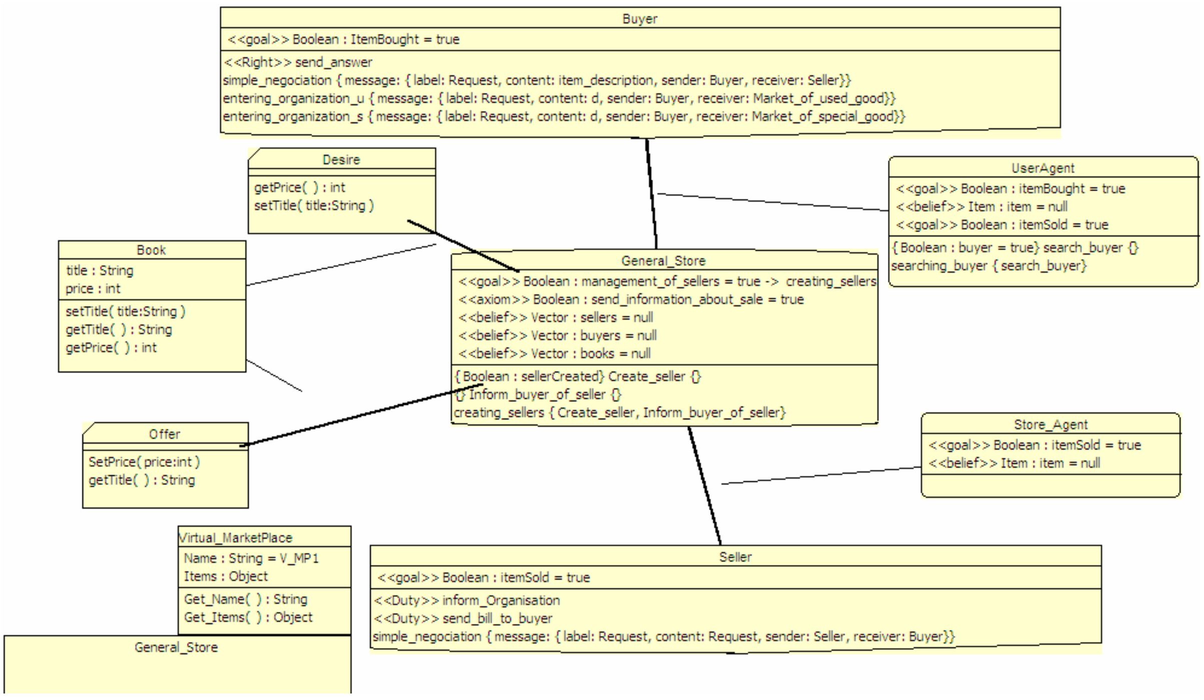
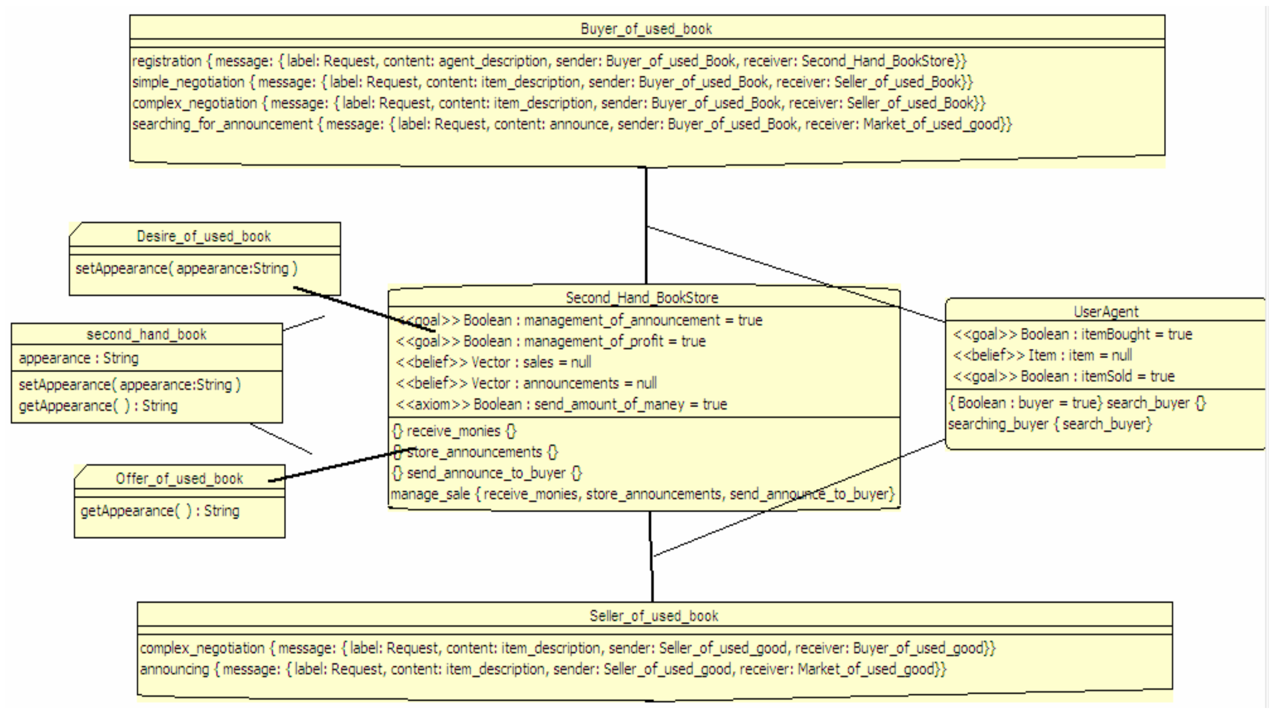

Buyer
<<goal>> Boolean : ItemBought = true
<<Right>> send_answer
simple_negociation {message: {label: Request, content: item_description, sender: Buyer, receiver: Seller}}
entering_organization_u {message: {label: Request, content: d, sender: Buyer, receiver: Market_of_used_good}}
entering_organization_s {message: {label: Request, content: d, sender: Buyer, receiver: Market_of_special_good}}
Desire
getPrice( ) : int
setTitle(title:String)
UserAgent
<<goal>> Boolean : itemBought = true
<<belief>> Item : item = null
<<goal>> Boolean : itemSold = true
{Boolean : buyer = true} search_buyer {}
searching_buyer {search_buyer}
Book
title : String
price : int
setTitle(title:String)
getTitle( ) : String
getPrice( ) : int
General_Store
<<goal>> Boolean : management_of_sellers = true -> creating_sellers
<<axiom>> Boolean : send_information_about_sale = true
<<belief>> Vector : sellers = null
<<belief>> Vector : buyers = null
<<belief>> Vector : books = null
{Boolean : sellerCreated} Create_seller {}
{} Inform_buyer_of_seller {}
creating_sellers {Create_seller, Inform_buyer_of_seller}
Offer
SetPrice(price:int)
getTitle( ) : String
Store_Agent
<<goal>> Boolean : itemSold = true
<<belief>> Item : item = null
Virtual_MarketPlace
Name : String = V_MP1
Items : Object
Get_Name( ) : String
Get_Items( ) : Object
General_Store
Seller
<<goal>> Boolean : itemSold = true
<<Duty>> inform_Organisation
<<Duty>> send_bill_to_buyer
simple_negociation {message: {label: Request, content: Request, sender: Seller, receiver: Buyer}}

Figure 11 MAS3 organisation diagram (MAS-ML model) (Torres da Silva and Choren Noya, 2003) (see online version for colours)


Buyer_of_used_book
registration { message: { label: Request, content: agent_description, sender: Buyer_of_used_Book, receiver: Second_Hand_BookStore}}
simple_negotiation { message: { label: Request, content: item_description, sender: Buyer_of_used_Book, receiver: Seller_of_used_Book}}
complex_negotiation { message: { label: Request, content: item_description, sender: Buyer_of_used_Book, receiver: Seller_of_used_Book}}
searching_for_announcement { message: { label: Request, content: announce, sender: Buyer_of_used_Book, receiver: Market_of_used_good}}
Desire_of_used_book
setAppearance(appearance:String)
second_hand_book
appearance : String
setAppearance(appearance:String)
getAppearance( ) : String
Offer_of_used_book
getAppearance( ) : String
Second_Hand_BookStore
<<goal>> Boolean : management_of_announcement = true
<<goal>> Boolean : management_of_profit = true
<<belief>> Vector : sales = null
<<belief>> Vector : announcements = null
<<axiom>> Boolean : send_amount_of_maney = true
{} receive_monies {}
{} store_announcements {}
{} send_announce_to_buyer {}
manage_sale { receive_monies, store_announcements, send_announce_to_buyer}
UserAgent
<<goal>> Boolean : itemBought = true
<<belief>> Item : item = null
<<goal>> Boolean : itemSold = true
{ Boolean : buyer = true} search_buyer {}
searching_buyer { search_buyer}
Seller_of_used_book
complex_negotiation { message: { label: Request, content: item_description, sender: Seller_of_used_good, receiver: Buyer_of_used_good}}
announcing { message: { label: Request, content: item_description, sender: Seller_of_used_good, receiver: Market_of_used_good}}

Relations defined in this diagram are described as follows:

- A virtual market environment is inhabited by all components (agents, organisation and item).
- A buyer agent belongs to the General Store organisation and plays buyer's role.
- A seller agent belongs to the General Store organisation and plays seller's role.
- An imported bookstore belongs to the General Store organisation and plays market of special good role.
- A hand bookstore belongs to the General Store organisation and plays market of used good role.
- An item belongs to the General Store organisation and plays desire and Offer role.

The entities and interactions defined in this diagram are described as follows:

- A buyer agent belongs to the second-hand bookstore organisation and plays buyer of second hand books role.
- A seller agent belongs to the second-hand bookstore organisation and play seller of second hand books role.
- An item belongs to the second-hand bookstore organisation and plays desire of second hand books and offer of second hand books role.

We present a detailed organisation diagram of MAS2 in Figure 12.

The entities and interactions defined in this diagram are described as follows:

- A buyer agent belongs to the imported bookstore organisation and plays buyer of imported bookstore role.
- A seller agent belongs to the imported bookstore organisation and plays seller of imported bookstore role.
- An item belongs to the imported bookstore organisation and plays desire of imported bookstore and offer of imported bookstore role.

However, the organisation diagram does not describe the relationship between roles. The role diagram completes the organisation diagram by modelling the relations between object roles and agent roles as indicated in Figure 13.

We can model the rest of the entities, whose correspondence we do not find in the MAS-ML language, with the class diagram. To keep the relationship between these objects and those modelled in the organisational diagram, we complete the missing links once having the UML model in phase PSM, the case of the entity task, which will be connected to the entity's action of organisation.

6.7 *Level PSM*

The UML XMI generated in the previous stage represents a PSM of the application. By using graphical tools, such as Rational Rose1 or Poseidon2, which import XMI files, it is possible to generate UML models of the application. The UML model created in this stage is a UML class diagram that contains the ASF framework classes and the classes related to the application that instantiate the framework (Alvez de Maria et al., 2005) (Figure 14). In order to illustrate the mapping between MAS-ML models and UML models, we describe how the organisation *General Store*, shown in Figure 10, was implemented using ASF in Figure 16.

Figure 12 MAS2: Organisation diagram (MAS-ML model) (see online version for colours)

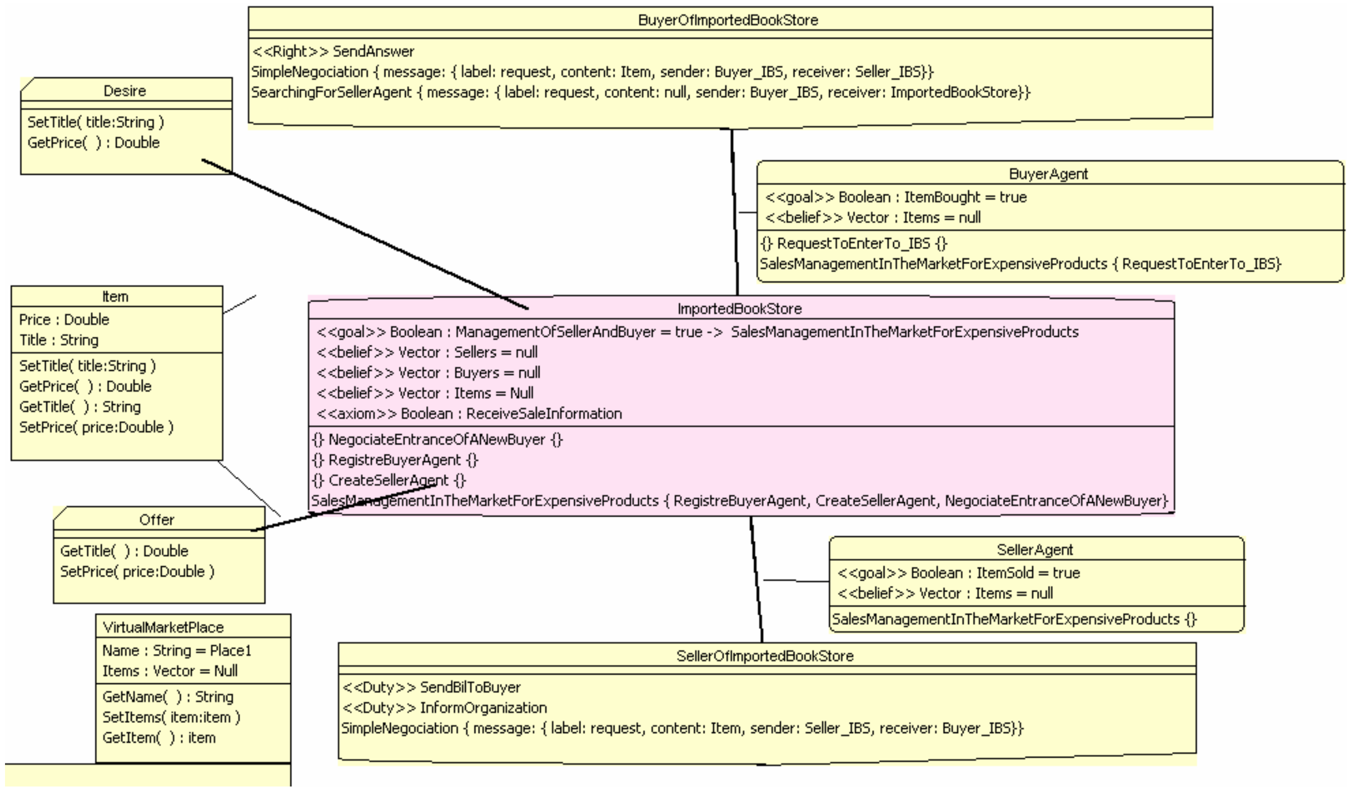

BuyerOfImportedBookStore
<<Right>> SendAnswer
SimpleNegociation { message: { label: request, content: Item, sender: Buyer_IBS, receiver: Seller_IBS}}
SearchingForSellerAgent { message: { label: request, content: null, sender: Buyer_IBS, receiver: ImportedBookStore}}
Desire
SetTitle(title:String)
GetPrice( ) : Double
BuyerAgent
<<goal>> Boolean : ItemBought = true
<<belief>> Vector : Items = null
{} RequestToEnterTo_IBS {}
SalesManagementInTheMarketForExpensiveProducts { RequestToEnterTo_IBS}
Item
Price : Double
Title : String
SetTitle(title:String)
GetPrice( ) : Double
GetTitle( ) : String
SetPrice(price:Double)
ImportedBookStore
<<goal>> Boolean : ManagementOfSellerAndBuyer = true -> SalesManagementInTheMarketForExpensiveProducts
<<belief>> Vector : Sellers = null
<<belief>> Vector : Buyers = null
<<belief>> Vector : Items = Null
<<axiom>> Boolean : ReceiveSaleInformation
{} NegociateEntranceOfANewBuyer {}
{} RegistreBuyerAgent {}
{} CreateSellerAgent {}
SalesManagementInTheMarketForExpensiveProducts { RegistreBuyerAgent, CreateSellerAgent, NegociateEntranceOfANewBuyer}
Offer
GetTitle( ) : Double
SetPrice(price:Double)
SellerAgent
<<goal>> Boolean : ItemSold = true
<<belief>> Vector : Items = null
SalesManagementInTheMarketForExpensiveProducts {}
VirtualMarketPlace
Name : String = Place1
Items : Vector = Null
GetName( ) : String
SetItems(item:item)
GetItem( ) : item
SellerOfImportedBookStore
<<Duty>> SendBilToBuyer
<<Duty>> InformOrganization
SimpleNegociation { message: { label: request, content: Item, sender: Seller_IBS, receiver: Buyer_IBS}}

Figure 13 MAS2 role diagram (see online version for colours)

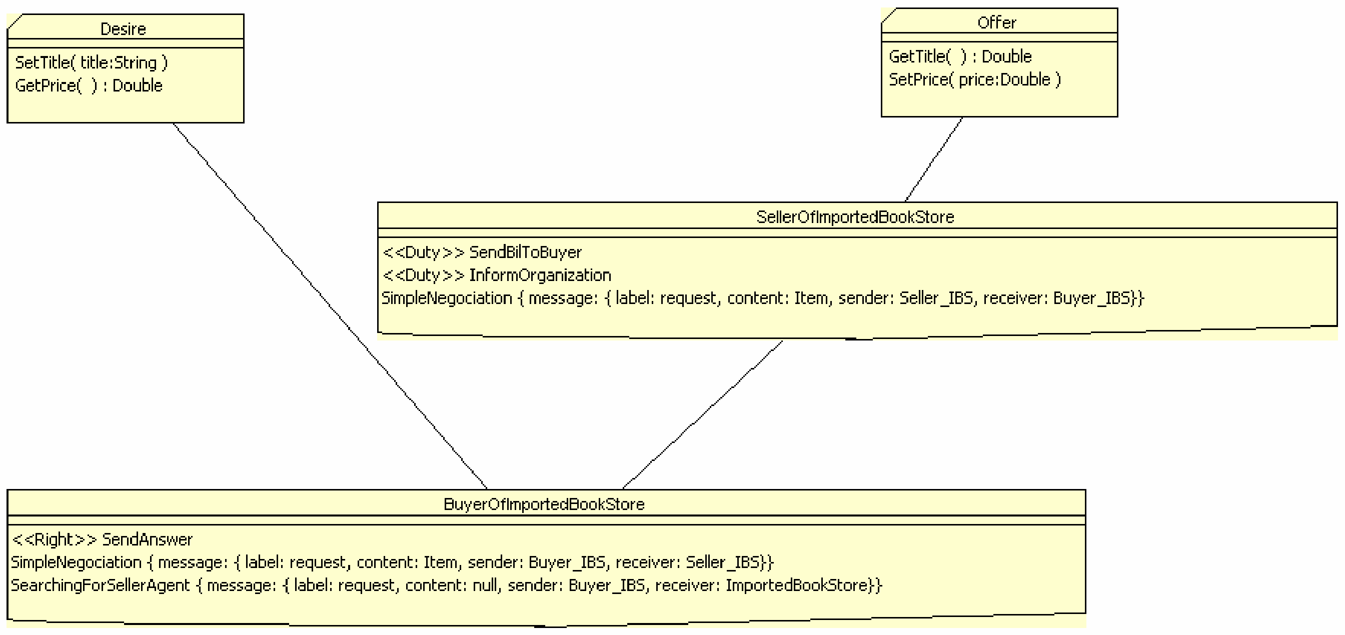

Desire
SetTitle(title:String)
GetPrice( ) : Double
Offer
GetTitle( ) : Double
SetPrice(price:Double)
SellerOfImportedBookStore
<<Duty>> SendBilToBuyer
<<Duty>> InformOrganization
SimpleNegociation { message: { label: request, content: Item, sender: Seller_IBS, receiver: Buyer_IBS}}
BuyerOfImportedBookStore
<<Right>> SendAnswer
SimpleNegociation { message: { label: request, content: Item, sender: Buyer_IBS, receiver: Seller_IBS}}
SearchingForSellerAgent { message: { label: request, content: null, sender: Buyer_IBS, receiver: ImportedBookStore}}

Figure 14 UML model for sample example (MAS1) (see online version for colours)

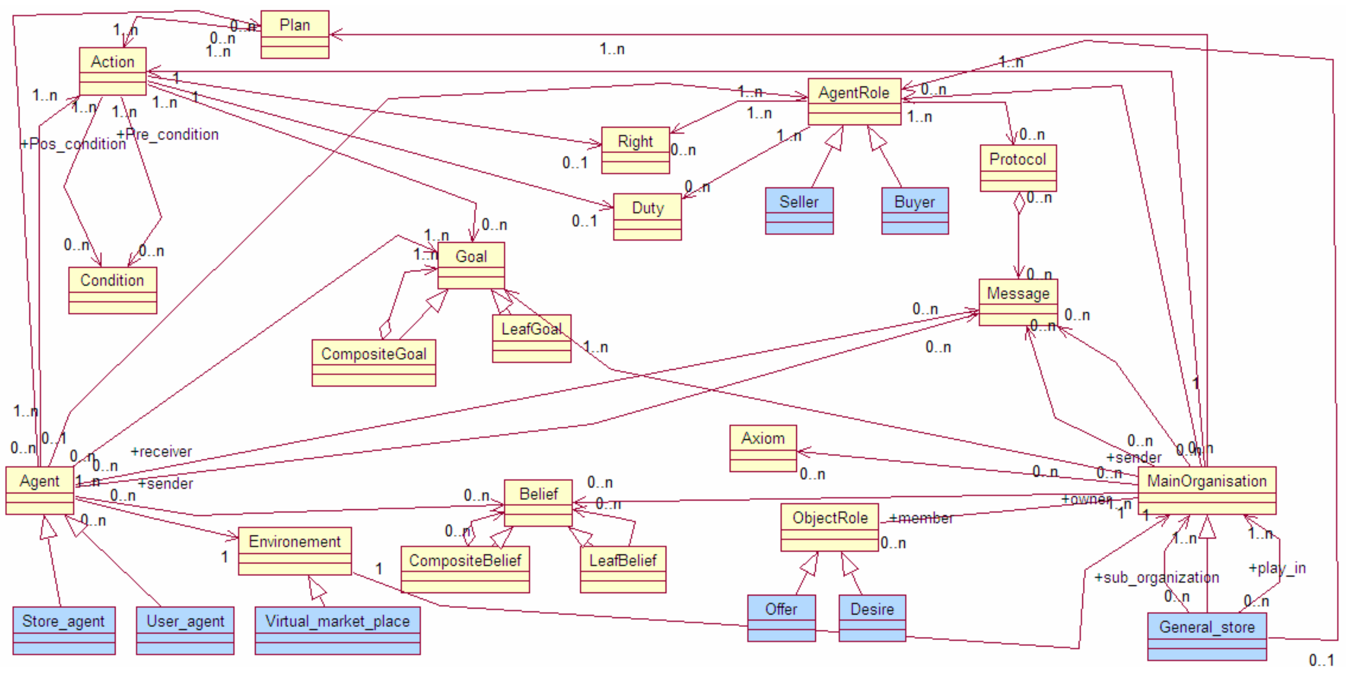
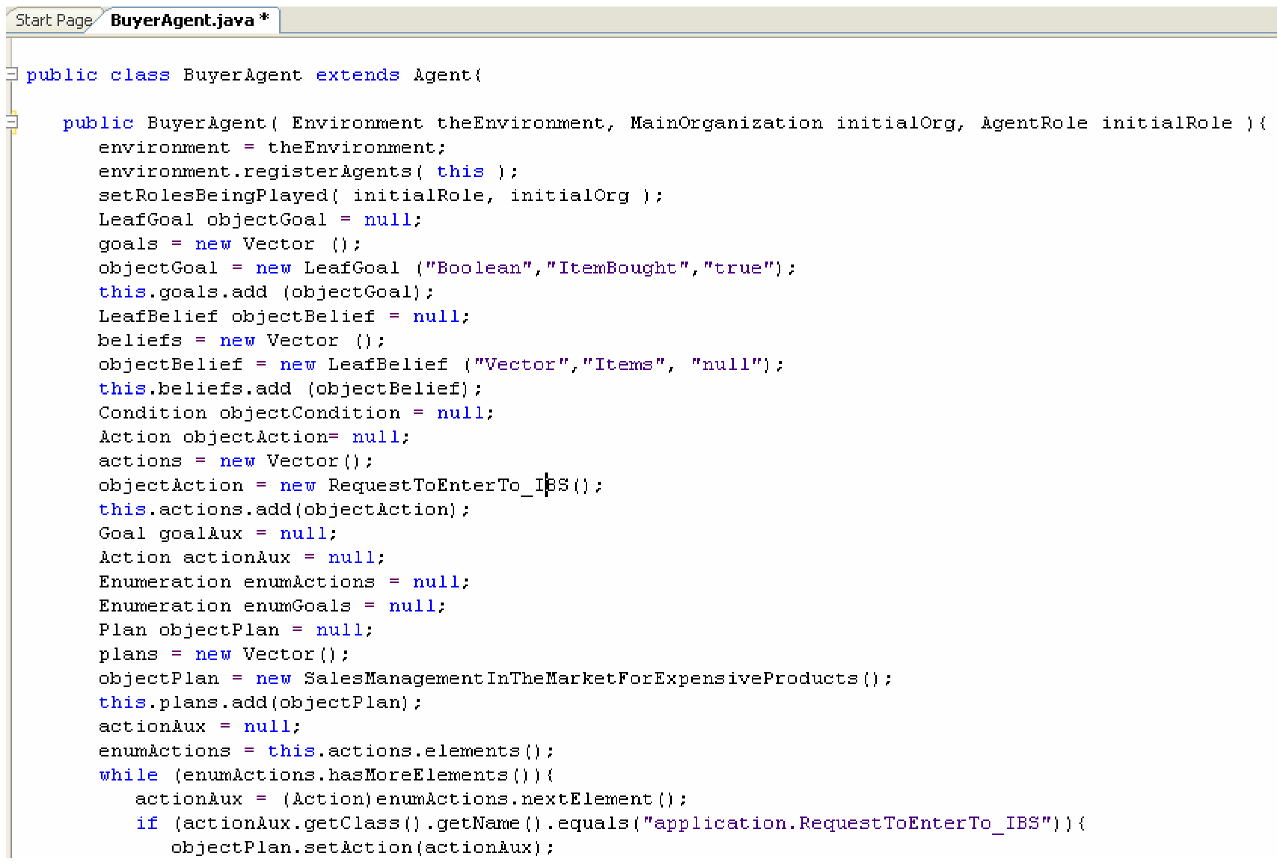


Figure 15 Application code (see online version for colours)



```
public class BuyerAgent extends Agent{

   public BuyerAgent( Environment theEnvironment, MainOrganization initialOrg, AgentRole initialRole ){
      environment = theEnvironment;
      environment.registerAgents( this );
      setRolesBeingPlayed( initialRole, initialOrg );
      LeafGoal objectGoal = null;
      goals = new Vector ();
      objectGoal = new LeafGoal ("Boolean","ItemBought","true");
      this.goals.add (objectGoal);
      LeafBelief objectBelief = null;
      beliefs = new Vector ();
      objectBelief = new LeafBelief ("Vector","Items", "null");
      this.beliefs.add (objectBelief);
      Condition objectCondition = null;
      Action objectAction= null;
      actions = new Vector();
      objectAction = new RequestToEnterTo_IBS();
      this.actions.add(objectAction);
      Goal goalAux = null;
      Action actionAux = null;
      Enumeration enumActions = null;
      Enumeration enumGoals = null;
      Plan objectPlan = null;
      plans = new Vector();
      objectPlan = new SalesManagementInTheMarketForExpensiveProducts();
      this.plans.add(objectPlan);
      actionAux = null;
      enumActions = this.actions.elements();
      while (enumActions.hasMoreElements()){
         actionAux = (Action)enumActions.nextElement();
         if (actionAux.getClass().getName().equals("application.RequestToEnterTo_IBS")){
            objectPlan.setAction(actionAux);
```

6.8 *Transforming PSM into code*

The UML models are transformed into code. This transformation corresponds to the last stage of the MDA process. Graphical tools that import UML XMI are automatically capable of generating code from an UML XMI. We obtained the application code and the framework code as shown in Figure 15.

Figure 15 shows part of the buyer agent class code generation: this class has the membership of the organisation that inhabits the environment, the agent role of this buyer agent, the actions carried by buyer agent, the process and the goal.

7 Application of transformation rules

7.1 Transforming the CIM into PIM

Based on the correspondence between agent and BP meta-models, we define instances components of final MAS depending on TAO meta-model.

Tables 5–7 show the instances of entities MAS1, MAS2, MAS3 in MAS-ML diagram.

Table 5 MAS1 instances depending in TAO concepts (see online version for colours)

Entity MAS ML	Instances	Signification
Main Organization	GeneralStore	General market
Organization	ImportedBookStore	Special good market
Organization	SecondHandBookStore	Used good market
Agent	UserAgent	Buyer agent
Agent	StoreAgent	Seller agent
AgentRole	Seller	Seller
AgentRole	Buyer	Buyer
AgentRole	MarketOfSpecialGood	Market of special good
AgentRole	MarketOfUsedGood	Market of used good
Object	Item	Item
ObjectRole	Desire	Element called
ObjectRole	Offer	Element offered
Environment	VirtualMarketPlace	Virtual market place

7.2 Transforming PIM into PSM

During this transformation, the UML XMI file representing the framework is inherited with the details of the application that are described in the file MAS-ML XMI; UML XMI file represents the application modelled in MAS-ML and implemented by the instantiation of the ASF framework.

This file is automatically generated by visual agent using the ASF framework by the two steps described earlier in Section 5.3.

For example, we show the organisation class of the framework and the imported bookstore class of the application that inherit it (Figures 16 and 17).

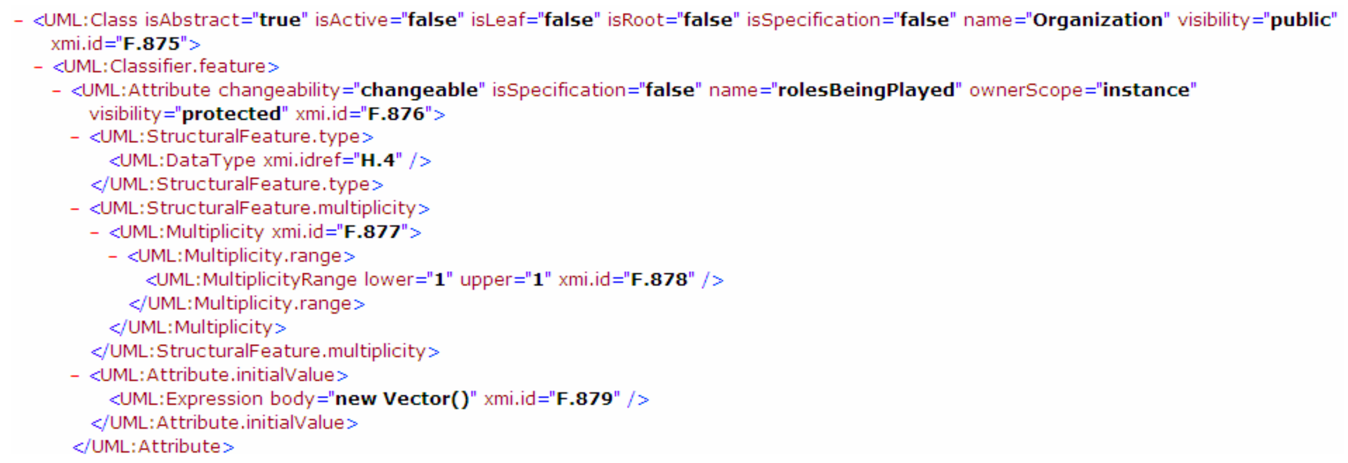
Table 6 MAS2 instances depending on TAO aspects (see online version for colours)

Entity MAS ML	Instances	Signification
Organization	ImportedBookStore	Special good market
Agent	UserAgent	Buyer Agent
Agent	StoreAgent	Seller Agent
AgentRole	Seller	Seller
AgentRole	Buyer	Buyer
Object	Item	Item
ObjectRole	Desire	Element called
ObjectRole	Offer	Element offered
Environment	VirtualMarketPlace	Virtual Market Place

Table 7 MAS3 depending in TAO aspects (see online version for colours)

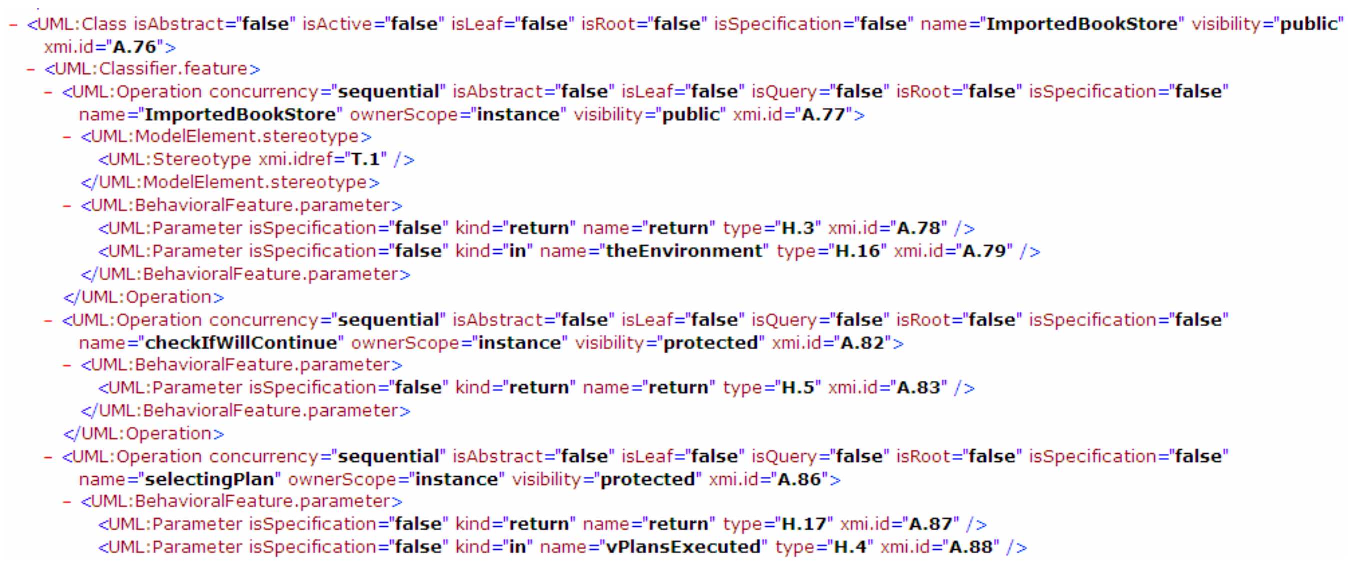
Entity MAS ML	Instances	Signification
Organization	SecondHandBookStore	Used good market
Agent	UserAgent	Buyer Agent
Agent	StoreAgent	Seller Agent
AgentRole	Seller	Seller
AgentRole	Buyer	Buyer
Object	Item	Item
ObjectRole	Desire	Element called
ObjectRole	Offer	Element offered
Environment	VirtualMarketPlace	Virtual market Place

Figure 16 Organisation class (UML XMI) (see online version for colours)



```
- <UML:Class isAbstract="true" isActive="false" isLeaf="false" isRoot="false" isSpecification="false" name="Organization" visibility="public"
   xmi.id="F.875">
  - <UML:Classifier.feature>
    - <UML:Attribute changeability="changeable" isSpecification="false" name="rolesBeingPlayed" ownerScope="instance"
        visibility="protected" xmi.id="F.876">
      - <UML:StructuralFeature.type>
          <UML:DataType xmi.idref="H.4" />
        </UML:StructuralFeature.type>
      - <UML:StructuralFeature.multiplicity>
        - <UML:Multiplicity xmi.id="F.877">
          - <UML:Multiplicity.range>
              <UML:MultiplicityRange lower="1" upper="1" xmi.id="F.878" />
            </UML:Multiplicity.range>
          </UML:Multiplicity>
        </UML:StructuralFeature.multiplicity>
      - <UML:Attribute.initialValue>
          <UML:Expression body="new Vector()" xmi.id="F.879" />
        </UML:Attribute.initialValue>
      </UML:Attribute>
```

Figure 17 Imported bookstore class (UML XMI) (see online version for colours)



```
- <UML:Class isAbstract="false" isActive="false" isLeaf="false" isRoot="false" isSpecification="false" name="ImportedBookStore" visibility="public"
    xmi.id="A.76">
  - <UML:Classifier.feature>
    - <UML:Operation concurrency="sequential" isAbstract="false" isLeaf="false" isQuery="false" isRoot="false" isSpecification="false"
        name="ImportedBookStore" ownerScope="instance" visibility="public" xmi.id="A.77">
      - <UML:ModelElement.stereotype>
          <UML:Stereotype xmi.idref="T.1" />
        </UML:ModelElement.stereotype>
      - <UML:BehavioralFeature.parameter>
          <UML:Parameter isSpecification="false" kind="return" name="return" type="H.3" xmi.id="A.78" />
          <UML:Parameter isSpecification="false" kind="in" name="theEnvironment" type="H.16" xmi.id="A.79" />
        </UML:BehavioralFeature.parameter>
      </UML:Operation>
    - <UML:Operation concurrency="sequential" isAbstract="false" isLeaf="false" isQuery="false" isRoot="false" isSpecification="false"
        name="checkIfWillContinue" ownerScope="instance" visibility="protected" xmi.id="A.82">
      - <UML:BehavioralFeature.parameter>
          <UML:Parameter isSpecification="false" kind="return" name="return" type="H.5" xmi.id="A.83" />
        </UML:BehavioralFeature.parameter>
      </UML:Operation>
    - <UML:Operation concurrency="sequential" isAbstract="false" isLeaf="false" isQuery="false" isRoot="false" isSpecification="false"
        name="selectingPlan" ownerScope="instance" visibility="protected" xmi.id="A.86">
      - <UML:BehavioralFeature.parameter>
          <UML:Parameter isSpecification="false" kind="return" name="return" type="H.17" xmi.id="A.87" />
          <UML:Parameter isSpecification="false" kind="in" name="vPlansExecuted" type="H.4" xmi.id="A.88" />
```

8 Conclusion

The MAS development process presented in this paper aims to provide an approach for modelling and implementing the MAS applications using the MDA approach. The proposed process consists of four stages: the first one is to represent the set of requirements in a particular domain and identify the MAS components. The second stage is to model the MAS by MAS-ML. At the third stage, the MAS-ML models are transformed into UML models. In the fourth stage, the UML models are automatically transformed into code.

For the implementation, we use visual agent that allows representing the graphical models with MAS-ML, to get the MAS-ML XMI file. Using the ASF framework, we generate the UML XMI file which is used to get UML models. And, finally we use AndroMDA for Java code generation.

Our perspectives are the validation of the proposed approach through other case studies using other environments such JADEX and AML.

References

Alter, S. (1999) *Information Systems: A Management Perspective*, 3rd ed., Addison-Wesley, Boston, MA.

Alvez de Maria, B., Torres da Silva, V. and Pereira de Lucena, C. (2005) 'An MDE-based approach for developing multi-agent system', *Electronic Communications of the EASST*, Vol. 3.

Azaiez, S. (2007) *Approche Dirigée par les Modèles pour le Développement de Systèmes Multi-Agents*, PhD Thesis, University of Savoie.

Brandão, A.A.F. and Torres da Silva Carlos, V. (2005) 'A model driven approach to develop multi-agent systems', No. 9/05.

Butler, K.A., Esposito, C. and Hebron, R. (1999) 'Connecting the design of software to the design of work', *Communications of the ACM*, Vol. 42, No. 1, pp.39–46.

Cunha, L.M., Barbosa, S.D.J. and Lucena, C.J.P. (2003) 'Leveraging the construction of semantic web applications using the model driven architecture', in Ashish, N. and Globe, C. (eds): *Semantic Web Technologies for Searching and Retrieving Scientific Data 2003*, Vol. 83. CEUR workshop proceedings of *the 2nd International Semantic Web Conference – ISWC2003*, Sanibel Island, FL. Available online at: www.CEUR-WS.org.

Jarraya, T. (2006) *Réutilisation des Protocoles d'Interaction et Démarche Orientée Modèles Pour le Développement Multi-agents*, PhD Thesis, University of Reims.

Kazakov, M., Abdulrab, H. and Debarbouille, G. (2002) *A Model-Driven Approach for Design of Mobile Agent Systems for Concurrent Engineering: MAD4CE Project*, Rapport Interne no. 01-002, University and INSA de Rouen.

Silva, V. and Lucena, C. (2003) *From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language*, School of Computer Science, University of Waterloo, Waterloo, Ontario.

Torres da Silva, V. and Choren Noya, R. (2003) *Using the MAS-ML to Model a Multi-Agent System*, Computer Science Department, SoC+Agent Group, Rua Marques de São Vicente, Rio de Janeiro.

Weiss, G. (2002) 'Agent orientation in software engineering', *Knowledge Engineering Review*, Vol. 16, No. 4, pp.349–373.