Developing Virtual Organizations Using MDD

Jorge Agüero, Miguel Rebollo, Carlos Carrascosa, Vicente Julián

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera S/N 46022 Valencia (Spain)
{jaguero, mrebollo, carrasco, vinglada}@dsic.upv.es

Abstract. Virtual Organizations are novel mechanisms where agents can demonstrate their social skills, due to the fact that they can work in a cooperative and collaborative way. Furthermore, organizations are frameworks where agents can achieve different types of agreements. But the development of organizations in MAS (Multi-Agent Systems) requires extensive experience in different methodologies and platforms. MDD (Model Driven Development) is a technique for generating application code developed from basic models and meta-models using a variety of automatic transformations. This paper presents a meta-model of *Virtual Organization* (of agents) using concepts and components at a suitable level of abstraction so that it can be implemented on different systems following a MDD approach. Based on this idea, a service-oriented organizations meta-model that is platform independent is presented. As an example, two model transformations that allow the unified model of the virtual organization to be translated into two different platforms are shown, facilitating the development process of agent-based software from the point of view of the user.

1 Introduction

Advances in new technologies based mainly on the Internet and the Web, such as electronic commerce, mobile/ubiquitous computing, social networks, etc., demonstrate the need to develop distributed applications with some intelligent capabilities. Multi-Agent Systems (MAS) are a powerful technology with very significant applications in distributed systems and artificial intelligence[18]. Supporting all of these developments requires the creation of platforms of highly heterogeneous agents, where agents work together through different interactions to support complex tasks, in a collaborative and dynamic way. Bearing this in mind, it is suitable for agents to display characteristics such as sociability, autonomy, self-organization, etc. Therefore, it is necessary to create open systems composed of a group of cooperative and heterogeneous agents, which work with local or individual goals and that must fulfill global goals. That is, a set of agents can reach agreements to achieve the group goals. The concepts of *organization* allow individual and group entities to be modelled in a very abstract way[3]. The

organization describes the main aspects of a society based on different viewpoints, such as: structure, functionality, norms, interactions, and environment. This type of organization is called *Virtual Organization*[13].

However, to implement an *Agents Virtual Organization* requires vast experience in one or more design platforms. A major challenge when designing MAS is to provide efficient tools that can be used by any user (non-expert users). The MDD approach can facilitate and simplify the design process and the quality of agent-based software[19], since it allows the reuse of software and transformation between models. MDD basically proposes the automatic generation of code from the models using the transformations. In other words, using models that have components that are platform independent, and by means of the transformations, those models are translated into components (or code) that depend on the execution platform, which integrates specific details about the system. Recently some proposals to implement these ideas (MDD techniques) have been proposed in MAS[16, 14, 17, 2], but none of these proposals focus on organization design.

Our purpose is to use the MDD approach for the design of organizations, taking the core and fundamental concepts, to specify an organization meta-model that is platform independent, and then use the translation mechanism to convert the unified meta-model into platform specific models for the execution of the agents. This paper presents a *Virtual Organization* meta-model, focused mainly on the integration of service-oriented technology and MAS, which allows support (from a very high level of abstraction) to be given to dynamic and open agent societies. Two transformation models for moving the unified model of the Virtual Organization to two different platforms are also proposed, allowing the feasibility of the proposal to be verified. These target platforms are: THOMAS[1] [7] and E-Institutions[2][12].

This paper is structured as follows. A brief summary of relevant works and their problems are discussed in Section 2. Section 3 presents MDD concepts and how to apply these concepts to the MAS paradigm. Section 4 explains how to design a virtual organization from the MDD viewpoint. Finally, the conclusions of this work are presented in Section 5.

2 Related Work

This section presents some related contributions with respect to organization modeling in agent-based systems, and discusses some problems. Furthermore, this section explains how, using the MDD approach, these problems can be addressed.

MAS development needs methodologies that allow the design of agent-based software to be optimized. The first methodologies that emerged can be classified as *agent-oriented*, without describing the *organization* explicitly. These systems are generally closed and external agents are prohibited. But in recent times, new *organizational-oriented* methodologies have emerged, which allow (partially) the

[1] http://users.dsic.upv.es/grupos/ia/sma/tools/Thomas
[2] http://e-institutions.iiia.csic.es

design of open MAS, leading to the development *heterogeneous* systems. These organizational-oriented methodologies allow external agents to access the system functionality, but the agents are obliged to adhere to the *social norms* of the system. Among the most important methodologies which allow the design of virtual organizations are: PASSI[9], MOISE[15], OperA[11] and GORMAS[4].

Now, with respect to the application of MDD for MAS, all of the above commented methodologies allow (to a major or minor extent) the design of agent organizations using their own set of specifications. But, in general, these proposals only use specific aspects of the platforms, which are in the majority of cases very different. This situation can create an added complexity for developers which try to employ these approaches. Moreover, some of them do not have an implementation phase, only defining high-level models, and difficulting enormously the developer work when tries to obtain executable code. But with the appearance of the MDD, it is possible to take advantage the flexibility provided by this approach to solve "some" of the problems found in the design of organizations in MAS.

The purpose of MDD is to create models legible by computers that can be understood by automatic tools to generate templates, proof models and even code, integrating them in multiple platforms[19]. From the viewpoint of the development of agent oriented systems, application development consists of obtaining the agent code that can be executed in different platforms. That is, to concentrate on the development of the application from a unified agent model and apply different transformations to get implementations for different platforms. Currently, the most common methodologies for MAS have an identified set of models that specify their characteristics. These models can be adjusted as MDD models that specify the concepts of the MAS as roles, behaviors, tasks, interactions or protocols. The models can be used to model a MAS without focusing on platform-specific details and requirements. After this, it is possible to transform any agent model into agent implementations for different platforms.

Only a few agent development methodologies have integrated the MDD techniques in the MAS design. The most relevant are MetaDIMA[16], TROPOS[6], PIM4AGENT[17], INGENIAS[14], AML [8] and AUML [5]. All of this work is for the application of MDD to the agent modeling process, but the biggest problem found, is that they do not consider the development of virtual organizations.

Therefore, our purpose is to apply the MDD approach to *organizational-oriented* methodologies. In other words, the contribution of this paper is to take as a starting point previous works of organization design in MAS, to develop a *Unified Model of Virtual Organization* which allows flexible implementation (including deployment) on different agent platforms with support for organizations.

3 Modeling Virtual Organizations with MDD

This section presents how to use the MDD approach to model organizations. In order to do this, the core concepts of MDD are presented first. Then, the meta-

models for the organization and the process of creating Virtual Organization using the MDD approach is explained.

3.1 MDD: Core Concepts

The MDD approach uses and creates different models at different abstraction levels, combining them when the application has to be implemented [19]. At high abstraction levels, the models are known as meta-models and they define the structure, semantics and constraints for a family of models (they are the model of a model). Models can be classified into three groups depending on their abstraction level: Computation Independent Model (CIM), which details the general concepts independently whether they are going to be implemented by a machine or not; Platform Independent Model (PIM), which represents the system functionalities without considering the final implementation platform; and the Platform Specific Model (PSM), obtained by combining the PIM with specific details about the selected platform.

A fundamental aspect of the MDD is the definition of sets of transformation rules between models, which allows the models to be automatically converted. Transformations are relational entities that describe how to map the rules concerning how the concepts of one model are transformed into the concepts of another model. These transformations can be applied at different abstraction levels. Horizontal transformations are applied over models that belong to the same level: PIM-to-PIM or PSM-to-PSM. Vertical transformations turn a general model into a more specific one (PIM-to-PSM) [19]. In general, all transformations are known as model-to-model transformations. Additionally, executable code can be automatically generated from a PSM. These transformations are known as model-to-code or model-to-text.

3.2 Using MDD to define an Agent Virtual Organization

One fundamental challenge when defining a platform independent meta-model in an organization is selecting which concepts or components will be included in order to model the organization. It is almost too obvious to mention that this is not a trivial task, since it must define the minimum components necessary for the organization. To achieve this objective, some of the most well-known approaches in the area of MAS organizations were studied (mentioned in Section 2). The purpose of this analysis is to extract the common features from the methodologies studied and adapt them to the current proposal, specifying a platform independent meta-model of the organization. Therefore, the transformation mechanism turns the platform independent model (PIM) into platform specific models (PSM's). Figure 1 shows diagram that illustrates the relationship among the meta-models.

In these methodologies, an agent organization is considered a social entity consisting of a specific number of members which carry out different tasks or functions, and that are structured according to communication patterns and

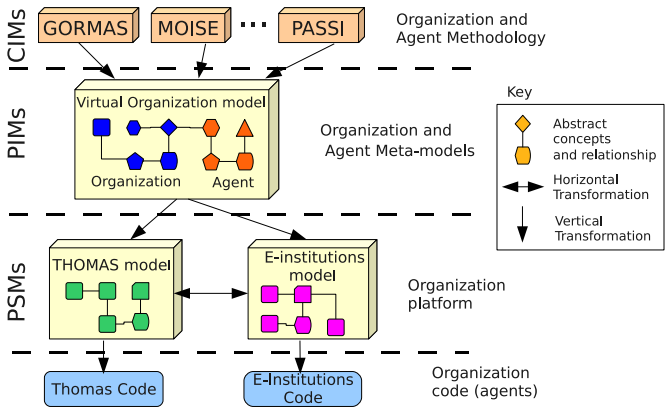

Fig. 1. Using MDD approach in Virtual Organizations

topology specific interactions, to achieve the global objective of the organization, based on behavior rules. The main aspects or factors of an organizations are the structure, functionality, dynamic, normative and its environment. So, to model the characteristics of these components five key concepts are used: *Organizational Unit*, *Service*, *Environment*, *Norm* and *Agent*[3]. These concepts make it possible to represent: (i) how the entities are grouped with each other, defining the relationship between the elements and their environment; (ii) what functionality they offer, including services for the dynamic entry and exit of agents in the organization, and (iii) what restrictions exist regarding the behaviors of system entities[10].

3.3 Organization Meta-model

This section presents our proposal for modeling agent organizations, based on five key concepts(previously mentioned): *Organizational Unit*, *Service*, *Environment*, *Norm* and *Agent*. The proposal is presented defining a set of meta-models, which provide the necessary concepts and primitives to describe structure, functionality, dynamism, environment and normative behaviors of the organization. This set of meta-models will be called π**VOM** (*Platform Independent Virtual Organization Model*) and is structured in different views or perspectives. The motivation for this is to support the evolution of the meta-model and to allow its possible growth in the future. The different views give complementary approaches which, when superposing themselves, generate the complete view of the system. Below, the different views proposed are briefly detailed due to the space limitations of the paper.

Structural view: This view includes all those elements that persist in the organization (see Figure 2(a)). The main concept employed in this view is the Organizational unit (OU). An OU is a group of agents who carry out specific

and differentiated tasks and follow a certain predefined communication and cooperation pattern[4]. This clustering can be seen externally as a single entity that pursues certain objectives, offers and/or requires certain services, and even plays a specific role in order to interact with other entities. Therefore, the organizational unit has a recursive nature and will not only contain agents but also other organizational units acting as atomic entities.

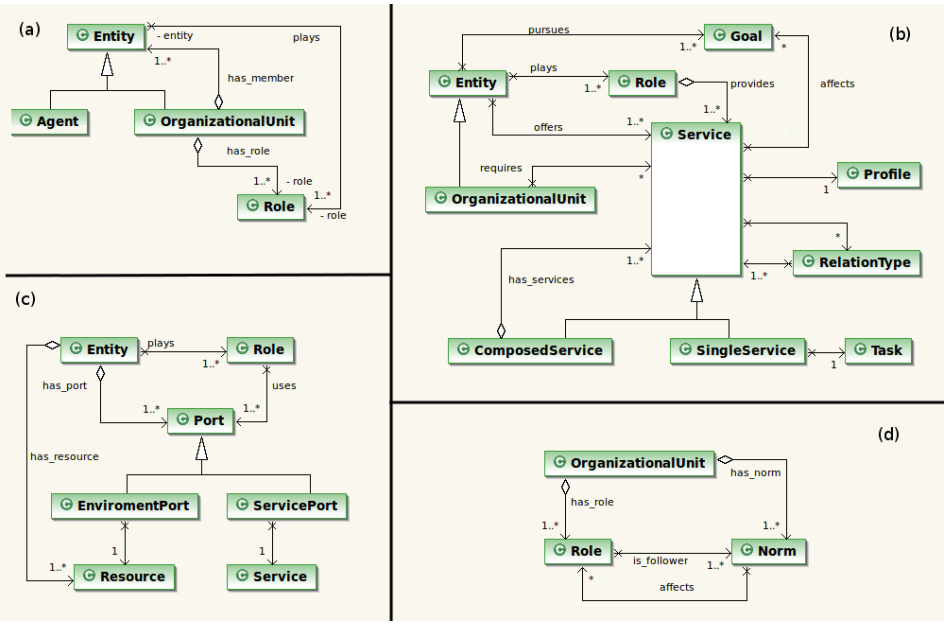

Fig. 2. Concepts used in Strutural(a), Functional(b), Environment(c) and Normative(d) viewpoints of πVOM

Functional view: This view details system functionality, based on services, tasks and objectives. This view shows the general behavior desired by the system. Thus, *Services* represent a certain functionality that an *entity* (Agent or OU) provides other entities, see the meta-model in Figure 2(b). The provider of a *Service* is always associated with a specific *Role*. Service functionality is carried out through the execution of certain *Tasks*, typically executed by the entity providing the service or delegated to other entities. The services can also be composed of several sub-services, and it is possible to define a "workflow" between them using (*RelationType*).

Environment view: This perspective details the elements (*Resources*) of the system. The *resources* are accessed and perceived through an *Environment-Port* (see the meta-model in Figure 2(c)). A *Resource* is an object in the environment that will be consumed by its members. A *Port* represents a point of

interaction between the entity and other elements of the model and serves as an interface to the real world.

Normative view: This model assumes that the organization is managed according to norms. *Norms* are used as mechanisms to limit the autonomy of agents in large systems and solve complex problems of coordination. This view specifies the set of rules and actions defined to control the behavior of members of the organization, specifically the roles of the organization (see Figure 2(d)). These rules correspond to obligations, permissions and prohibitions; also as sanctions and rewards to carry out on its members.

Agent view: A *Agent* is the basic entity of MAS which is within the organization and uses a series of interrelated components, shown in Figure 3. Our *Agent* has a set of basic components: *Capabilities* represent the *know-how* of the *Agent* and follow a pattern of *event-condition-action*. The *Behaviours* implement the roles that the agent can play. The *Task* is the component where the code base of the agent actions is written. For a more detailed explanation of the components of the agent meta-model, please refer to Aguero at el[1].

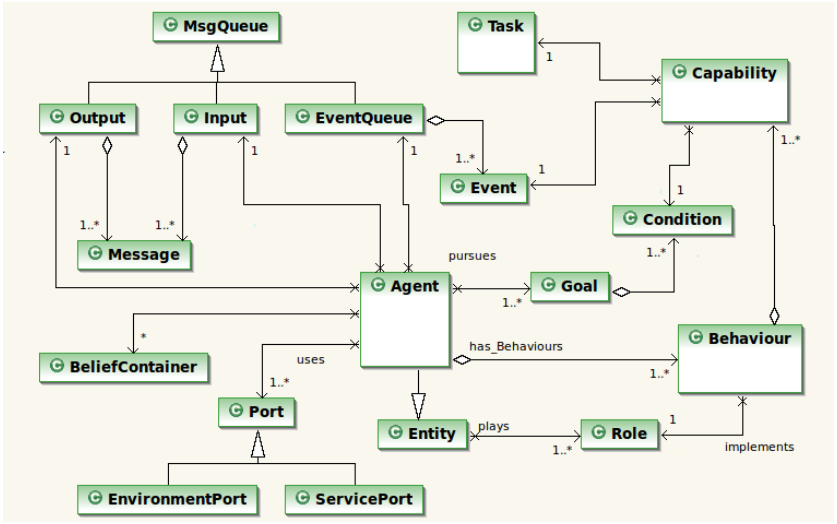

Fig. 3. Concepts used in Agent viewpoint of πVOM

4 Developing an organization with a MDD approach

Once the set of models that characterize our proposed model of *Virtual Organization platform-independent* has been presented, the process for transforming the Virtual Organization into different platforms must be defined. The design process begins by selecting how abstract concepts (which are part of the unified

organization model) are mapped to the target platforms. For this paper, we focus on the study of transformations on two platforms that support agents organizations: THOMAS[7] and E-Institutions[12]. The transformation defines a set of mapping rules. The first mapping rules define which concepts of the source meta-model (πVOM) are transformed to which concepts of the target meta-model, a model-to-model transformation (PIM-to-PSM). This is illustrated by dotted lines in Figure 4. The second transformation translate the models into the code templates of the organization, which can be optionally combined with code written manually by the user. This is a model-to-text transformation (PSM-to-code).

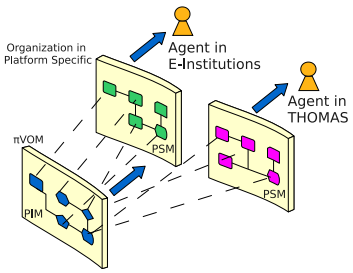

Fig. 4. Tranformation from πVOM to differents platforms

4.1 Development Process Using the THOMAS Platform

The πVOM meta-model is very similar to the model of the organization programmed in THOMAS, since both works are based (partially) on the methodology and artifacts proposed by GORMAS. For this reason, the automatic transformations are relatively easy to describe. Almost all of the abstract concepts of πVOM are represented in THOMAS, so the model-to-model transformation rules are expressed almost as a one-to-one relationship. It is convenient to notice that some concepts in THOMAS have a more detailed feature than πVOM, because THOMAS is a platform-specific model. The transformation rules that must perform the translation between different models are shown in Table 1 (from **rule 1** to **rule 8**).

4.2 Development Process Using the E-Institutions Platform

E-institutions provide a set of tools that are widely used in the area of agents to model organizations, which are defined as a Multi-Agent Systems and can be effectively designed and implemented as *electronic institutions* composed of a vast amount of heterogeneous (human and software) agents playing different roles and interacting by means of speech acts[12]. They take inspiration from traditional human institutions, and offer a general agent-mediated computational model

Rule	Concept	Transformation
	THOMAS	
1	Organizational Unit	πVOM.OU $\Rightarrow$ THOMAS.OU
2	Agent	πVOM.Agent $\Rightarrow$ THOMAS.Agent
3	Role	πVOM.Role $\Rightarrow$ THOMAS.Role
4	Service	πVOM.Service $\Rightarrow$ THOMAS.Service
5	Norm	πVOM.Norm $\Rightarrow$ THOMAS.Norm
6	RelationType	πVOM.RelationType $\Rightarrow$ THOMAS.Process
7	Resource	πVOM.Resource $\Rightarrow$ THOMAS.Resource
8	Goal	πVOM.Goal $\Rightarrow$ THOMAS.Goal
	E-INSTITUTIONS	
9	Organizational Unit	$\bigcup \pi$VOM.OU $\Rightarrow \bigcup$ EI.Scene
10	Agent	πVOM.Agent $\Rightarrow$ EI.Agent
11	Role	πVOM.Role $\Rightarrow$ EI.Role
12	Service	$\forall \pi$VOM.Service $\in$ OU $\Rightarrow \bigcup$ EI.State $\in$ Scene
13	RelationType	πVOM.RelationType $\Rightarrow$ EI.Transition OR EI.Illocutions
14	Norm	πVOM.Norm $\Rightarrow$ EI.Norm
15	Goal	πVOM.Goal $\Rightarrow$ EI.Norm
16	Resource	πVOM.Resource $\Rightarrow$ EI.World

Table 1. Rules from π *VOM* to THOMAS and E-Institutions

that serves to realize an *agent-mediated electronic institutions*. In Figure 5, the main meta-model components of E-Institutions are presented. Transformation rules for E-Institutions are shown in Table 1 (from **rule 9** to **rule 16**). Due to the space limitations of this paper, each transformation rule cannot be detailed. Instead, in order to illustrate its use to the reader, an application scenario of core rules is described in the next section.

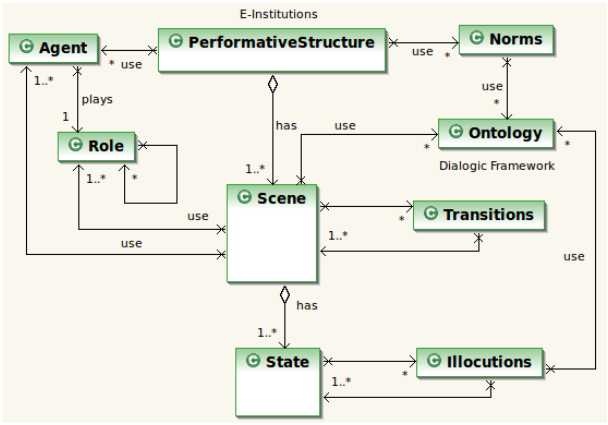

Fig. 5. Core concepts used in E-Institutions

4.3 Usage Scenario

To illustrate the use of the rules, a case scenario for making flight and hotel arrangements is utilized. This is a well known example that has been modeled

by means of electronic institutions in previous works (Dignum [11]; Argente et al [3]). The *Travel Agency* example is an application that facilitates the interconnection between clients (individuals, companies, travel agencies) and providers (hotel chains, airlines); delimiting services that each one can request or offer. The system controls which services must be provided by each entity. Provider entities are responsible for the internal functionality of these services. However, the system imposes some restrictions on service profiles, service requesting orders and service results. In this system, agents can search for and make hotel and flight reservations, and pay in advance for bookings. This case study is modeled as an organization (*TravelAgency*) inside which there are two organizational units (HotelUnit and FlightUnit) which represent a group of agents. Each unit is dedicated to hotels or flights, respectively. Three kind of roles can interact in the Travel Agency example: customer, provider and payee roles. The Customer role requests system services. More specifically, it can request hotel or flight search services, booking services for hotel rooms or flight seats, and payment services. The Provider role is in charge of performing the service. Finally, the Payee role gives the advanced payment service. Figure 6 shows the *TravelAgency* structure, with its units, roles and relationships between them.

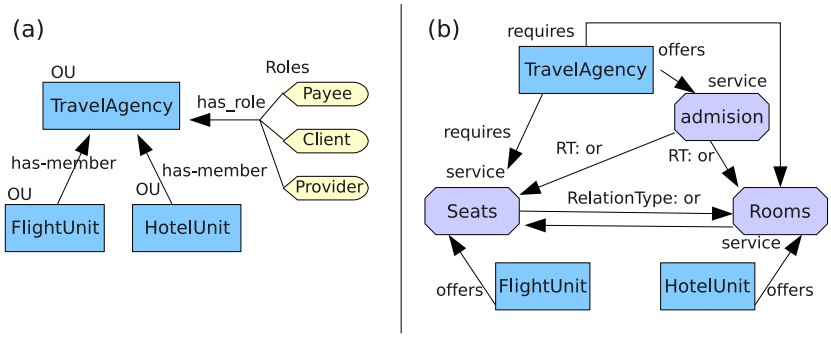

Fig. 6. Travel Agency in πVOM

The process begins by modeling the Travel Agency (structural and functional models, see Figure 6(a)(b)), and applying the rules mentioned previously, in order to obtain the organizations on THOMAS and E-Institutions. In the case of THOMAS the models are very similar and their transformation in almost direct (see in Figure 7(a)). But, getting the organization in E-Institutions and creating the core templates of *PerformativeStructure* seen in figure 7(b), requires the application of the **rule 9** and **rule 13**.

Rule 9 allows all of the *Scenes* in the E-Institutions that correspond to the *OU* of πVOM (3 in total) to be obtained. Additionally the *Scenes* for entrance and exit are needed (root and output). After the application of **Rule 13**, as the *Services* are composed in the order given by the *RelationType*, we can specify the type of *Transitions* between the different *Scenes* in the E-Institutions, which in

this case correspond to *Transitions* of type OR. As previously mentioned, this mapping generates the basic template of the PerformativeStructure of the E-Institutions (Figure 7(b)). With the application of the remaining rules, a more detailed description of the PerformativeStructure is obtained, but due to the limitations of space in this paper, they are not explained.

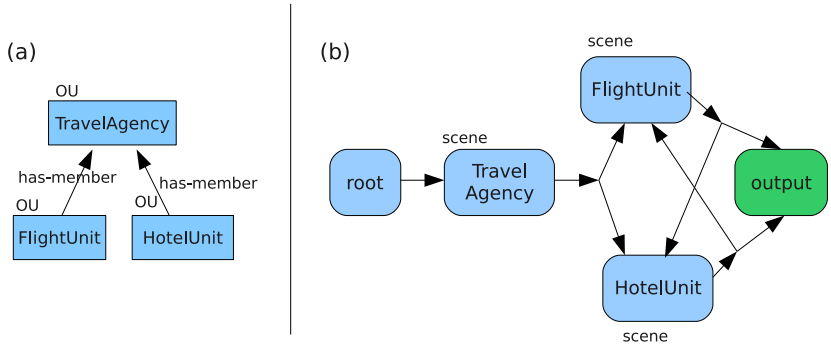

Fig. 7. Travel Agency in THOMAS and E-Institutions

5 Conclusions

This work presents a meta-model of Virtual Organization (called πVOM) which allows organizations in MAS to be modeled using abstract components which are independent of the implementation platform following a MDD approach. This meta-model is divided into five views that focus on the most important aspects of the Virtual Organizations and these views can easily be extended if required to a specific domain.

From unified meta-model, creating code templates for specific platforms of organizations is possible. This was discussed and exemplified using transformations on the THOMAS and E-Institutions platforms. These transformations show that the meta-model can be considered platform independent. As future work, we will propose additional transformations in order to obtain the agent instances (to generate the agent code) in the proposal and other frameworks. We will propose the introduction of specific components/views (additional) to reach agreements. We propose that the Virtual Organization provides the framework and components required to model agreements, or in other words, use the MDD approach to drive design-oriented agreements.

Acknowledgment: This work was partially supported by TIN2006-14630-C03-01 and PROMETEO/2008/051 projects of the Spanish government and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián. Does android dream with intelligent agents? In *Int. Symposium on Distributed Computing and Artificial Intelligence 2008(DCAI2008)*, ISBN: 978-3-540-85862-1, pages 194–204, 2008.
2. J. Agüero, M. Rebollo, C. Carrascosa, and V. Julián. Agent design using Model Driven Development In *7th Int. Conference on Practical Applications of Agents and Multi-Agent Systems(PAAMS2009)*, ISBN 978-3-642-00486-5, pages 60–69, 2009.
3. E. Argente, V. Julian, and V. Botti. Mas modelling based on organizations. In *9th Int. Workshop on Agent Oriented Software Engineering*, pages 1–12, 2008.
4. E. Argente Villaplana. *Gormas: Guías para el desarrollo de sistemas multiagente abiertos basados en organizaciones*. PhD thesis, UPV, Spain, 2008.
5. B. Bauer. Uml class diagrams revisited in the context of agent-based systems. *Proceedings Agent-Oriented Software Engineering*, pages 101–118, 2002.
6. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
7. C. Carrascosa, A. Giret, V. Julian, M. Rebollo, E. Argente, and V. Botti. Service oriented multi-agent systems: An open architecture. In *Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1–2, 2009.
8. R. Cervenka and I. Trencansky. *The Agent Modeling Language – AML*. Whitestein Series in Software Agent Technologies and Autonomic Computing, 2007.
9. M. Cossentino and C. Potts. Passi: A process for specifying and implementing multi-agent systems using uml. Technical report, University of Palermo, 2001.
10. N. Craido, E. Argente, V. Julián, and V. Botti. Designing virtual organizations. In *7th Int. Conference on Practical Applications of Agents and Multi-Agent Systems(PAAMS2009)*, ISBN 978-3-642-00486-5, pages 440–449, 2009.
11. V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. Phd dissertation, Utrecht University, 2003.
12. M. Esteva, J. A. Rodríguez-Aguilar, C. Sierra, and J. L. Arcos. On the formal specifications of electronic institutions. *Lecture Notes in Computer Science*, pages 126–147, 2001.
13. J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: an organizational view of multi-agent systems. *Lecture Notes in Computer Science*, pages 214–230, 2004.
14. I. Garca-Magario, J. Gómez-Sanz, and R. Fuentes. Ingenias development assisted with model transformation by-example: A practical case. In *7th Int. Conf. on Practical Applications of Agents and MultiAgent Systems*, pages 40–49, 2009.
15. B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. In *Third European Workshop on Multi-Agent Systems (EUMAS2005)*, pages 484–485, 2005.
16. Z. Guessoum and T. Jarraya. Meta-models & model-driven architectures. In *Contribution to the AOSE TFG AgentLink3 meeting*, 2005.
17. C. Hahn, C. Madrigal-Mora, and K. Fischer. A platform-independent meta-model for multiagent systems. In *Autonomous Agents and Multi-Agent Systems*, 18(3):239–266, 2009.
18. N. Jennings and M. Wooldridge. Applications of intelligent agents. In *Agent Technology: Foundations, Applications, and Markets*, pages 3–28, 1998.
19. A. Kleppe, J. B. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Professional, 2003.