EMFGormas meta-model manual

Emilia Garcia, E. Argente, and Adriana Giret

Departamento de Sistemas Informaticos y Computacion,
Universidad Politecnica de Valencia
{mgarcia,eargente,agiret}@dsic.upv.es

1 EMFGormas metamodel

This section details our approach to model Service-oriented Open MAS using the MDA Eclipse technology. For that reason, we have defined a platform independent unified meta-model that describes the modeling language in a formal way, establishing the primitives and syntactic-semantic properties of organizations and multiagent systems. The unified meta-model that we define is based on GORMAS [6].

GORMAS meta-models represent an interesting starting-point for the specification of a SOMAS model, but they present some drawbacks. Firstly, they imply a rather high modeling cost, as meta-models are composed of different views, so a designer should employ a total of 19 types of diagrams for modeling a system, as well as several instances for each type of diagram. For example, he/she should define an agent model for each identified agent. Thus, the cost of the modeling task may increase exponentially to the number of views. Secondly, GORMAS meta-models have several relationships replicated in different models, in order to make them easier to be understood from a theoretical point-of-view. However, these replicated relationships might introduce several conflicts and inconsistences inside a graphical development tool, so they should be removed. Finally, since GORMAS is mainly an extension of both INGENIAS [15] and ANEMONA [17] meta-models, it is mainly defined from a theoretical point-of-view, but not from a graphical point-of-view, so a final user will find rather difficult to manage all GORMAS diagrams in an easy way. For example, describing an interaction process implies taking into account three different types of diagrams and check that they are consistent with each other. Thus, it is necessary to develop a unified meta-model that clearly takes into account the final usage of diagrams by designers.

In order to resolve these drawbacks, we have defined a unified meta-model for Service-Oriented Open MAS that simplifies GORMAS meta-models, reducing the number of core views and entities and also removing all unnecessary entities and relationships, so then making its graphical usage easier. The proposed unified meta-model is composed of 10 views (Figure 3) that are described below. The

1.1 Model Driven Architecture and Eclipse technology

In the software engineering field, a model-driven software development process should be clearly defined by specifying all the phases of the development lifecy-

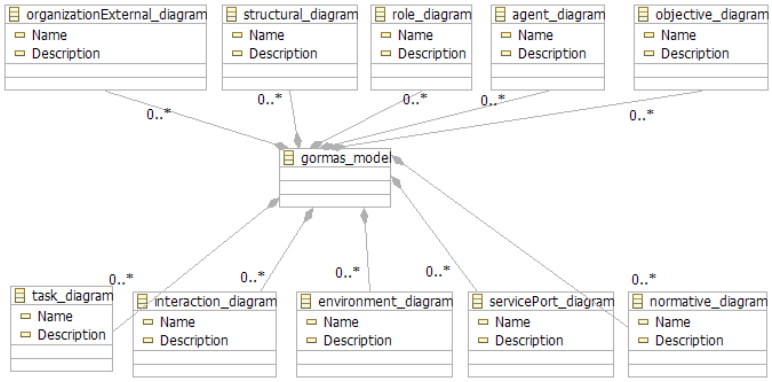

Fig. 1. Unified meta-model for SOMAS

cle. Furthermore, CASE tools should be provided, as a support of the different tasks in the model based design such as analysis and verification of models or automatic transformation from one specification language to another, in a transparent and simple way.

The MDA [7] initiative is proposing a standard to which the meta-models of the specification languages used in the modeling process must be compliant with, that is the Meta Object Facility (MOF), and a set of requirements for the transformation techniques that will be applied when transforming a source model into a target model, referred as the Query/View/Transformation (QVT) approach [20]. Basically, MDA proposes an approach to software development based on modeling and on the automated mapping of source models to target models. The models that represent a system and its environment can be seen as a source model and code can be seen as a target model as well.

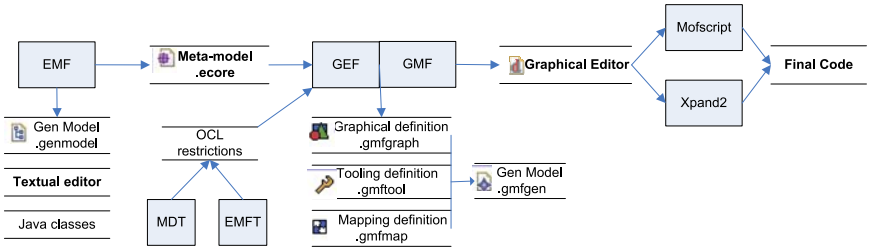

Fig. 2. Eclipse plugins structure

Following these MDA standards, the Eclipse Platform [8] is an open source initiative that offers a reusable and extensible framework for creating IDE oriented tools. The Eclipse Platform itself is organized as a set of subsystems (implemented in one or more plug-ins) built on the top of a small runtime engine.

As shown in Figure 2, some plugins like EMF, GEF, GMF, Mofscript, Xpand2 are offered by the platform in order to allow the creation of IDE tools based on an specific meta-model.

The EMF plug-in offers a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, Rational Rose or the ecore standard, EMF provides tools and runtime support to produce a set of Java classes for the model. Moreover, EMF generates a textual modeler editor from the meta-model, whereas the GEF and GMF plug-in allows developers to create a rich graphical editor from an existing application model.

Recently, model driven approaches have been recognized and become one of the major research topics in agent oriented software engineering community. Some works like [21–23] show how MDA can be effectively applied to agent technologies. Furthermore, they show how the MDA technology can help to reduce the gap between the analysis phase and the final implementation. Some CASE tools to model MAS have been developed using the Eclipse technology [24, 25]. Regarding these studies and tools, it can be said that the current application of the MDA process on MAS development is still in its preliminary phase.

2 Unified meta-model for SOMAS

This section details our approach to model Service-oriented Open MAS using the MDA Eclipse technology. As explained before, this technology requires defining a platform independent unified meta-model that describes the modeling language in a formal way, establishing the primitives and syntactic-semantic properties of organizations and multiagent systems. The unified meta-model that we define, named *EmfGormas*, is based on VOM [26].

The proposed unified meta-model is composed of 10 views (Figure 3) that are described below. The implementation of the meta-model in ecore format can be consulted in *http://www.dsic.upv.es/users/ia/sma/tools/EMFgormas/index.html*.

Firstly, the analysis of the system requirements is carried out, defining the global goals of the organization, the stakeholders and the functionality that the organization provides and requires from these stakeholders. All this is depicted in the **Organization external view** diagram (Figure 5).

Secondly, the analysis of the goals of the organization is carried out in which the global goal of the organization is refined into more functional goals, which represent non-functional requirements that must or should be accomplished by the organizational units of the system. All this is defined in the **Objectives view** diagram (Figure 9).

Next, the components of the organization are defined, i.e. the Organizational Units (OU), which represent groups of members of the organization; the roles defined inside each OU that will be related to the system functionality; their social relationships; the products available by the OUs that can be accessible for their members; and the norms that control the global behavior of the OU

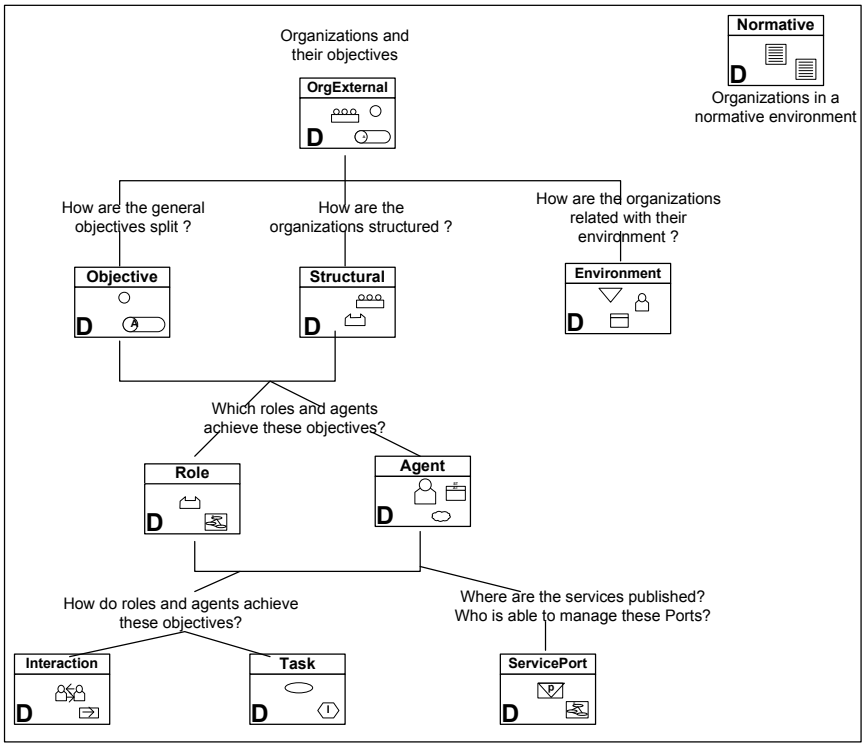

Fig. 3. Unified meta-model for SOMAS

members. These concepts are described in the **Structural view** diagram (Figure 6).

Moreover, the products of the environment are analysed, divided into applications (functional interfaces) or resources (with consumable features). The permissions for accessing these elements of the enviroment are also defined. All them are depicted in the **Environment View** diagram (Figure 13).

Thirdly, the internal functionality of the OUs is defined, by means of the Role and Agent Views. In the **Role View** (Figure 7), the roles are related with their responsibilities (tasks), capabilities (services) and objectives. The **Agent View** (Figure 8) describes the concrete responsibilities of agents (tasks), the roles that they can play inside an OU, the services that offer to other entities, their mental states (believes, events and facts) and the goals that they pursue.

The way in which the roles and agents achieve their goals is defined by means of the Interaction and Task Views. In the **Interaction View** (Figure 11), the participants of the interaction are identifed, as well as the sequence of activities (task and services) and performatives that are employed through the interaction. In the **Task View** (Figure 10), the specific functionality of the services and tasks is detailed, more specifically, the service description (*ServiceProfile*); its specific

implementation, by means of service or task composition; as well as the sequence of tasks that is needed.

Finally, the **Service Port View** (Figure 12) defines the way in which the services must be published so as to be discovered by any agent. Thus, the service publication points (service port) are identified, as well as the entities that control each port and give permissions for registring or accessing services.

Throught the whole process, whenever a restriction on the behavior of the system entities is identified, it should be described in the **Normative View** diagram (Figure 14).

As explained before, this is not a linear process but an iterative one, in which the identification of a new element of the system (such as a new role, a new agent, etc.) implies the integration of this element in all the diagrams in which it is needed. For example, all agents defined in the Agent View diagram must be also integrated into the Structural View diagram. And the services and tasks defined in both Role View and Agent View diagrams must also appear in the Task View diagram. These interdependences between diagrams are all controlled by the *EMFGormas* tool.

3 Detailed description of EMFGormas

3.1 Entities

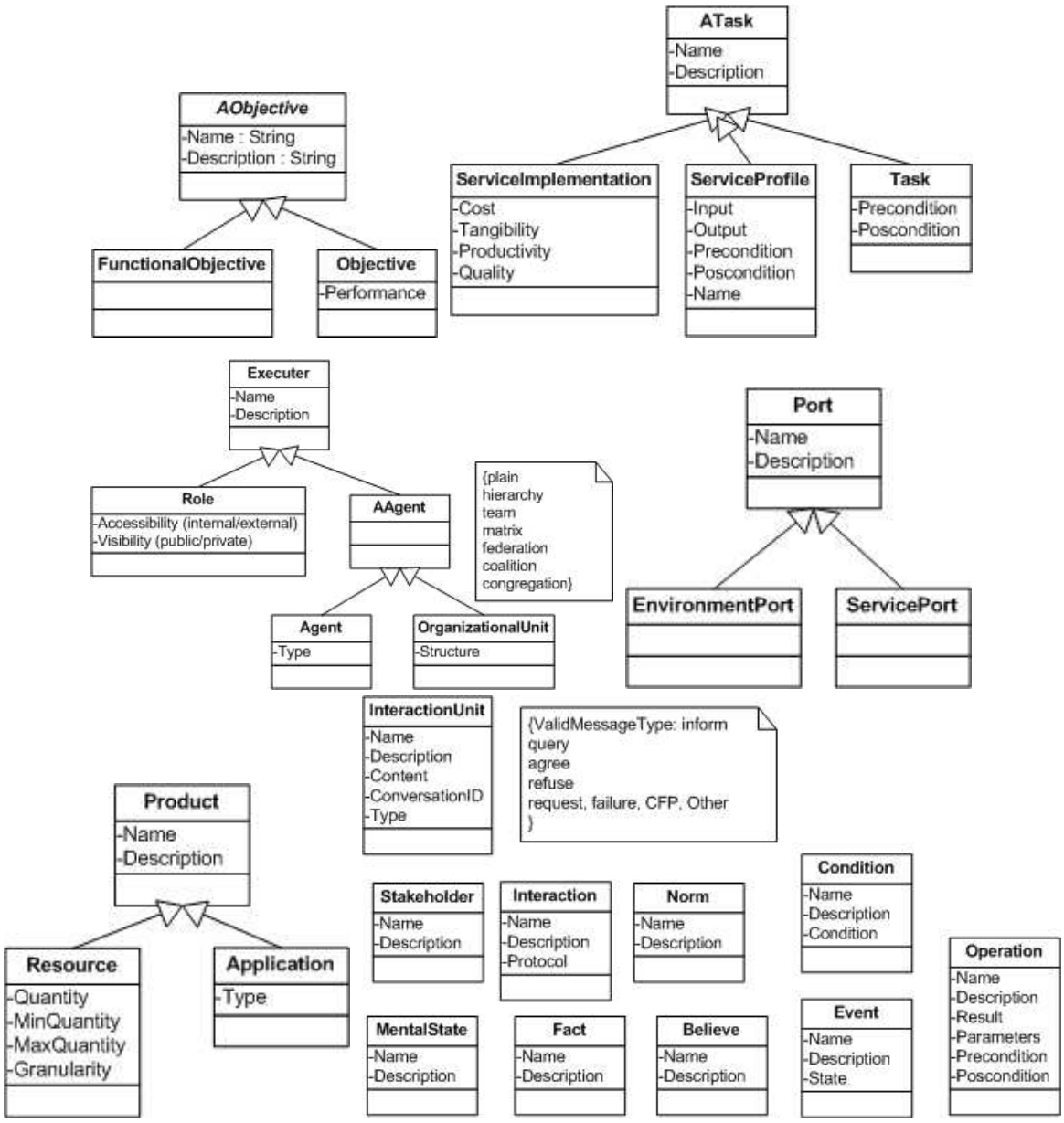

Fig. 4. Entities.

3.2 Organization external view

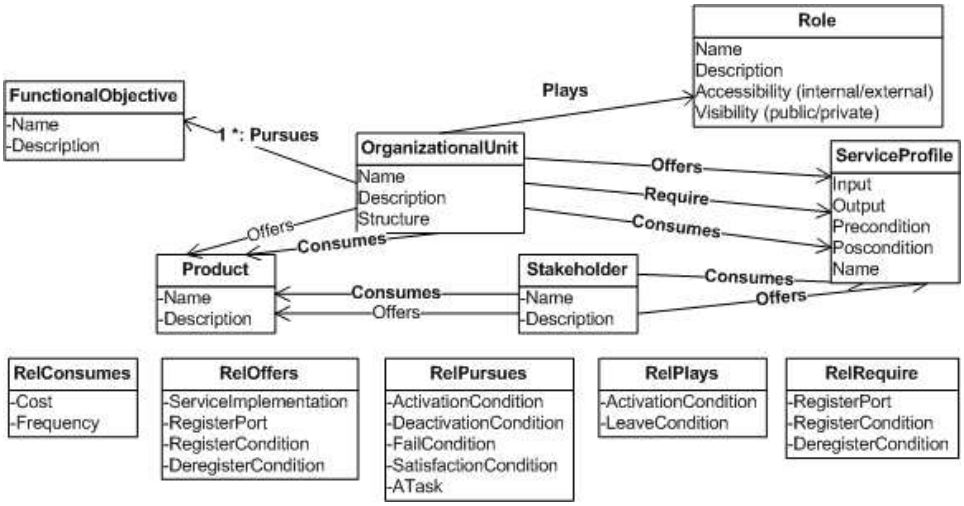

Fig. 5. Emf metamodel Organization External View.

The **Organization external view** (Figure 5) allows defining the global goals of the organizations and the functionality that organizations provide and require from their environment. More specifically, it defines:

- the *functional objectives* that are pursued by the organization, i.e., non-functional requirements (softgoals) that can be defined for describing the global behavior of this organization.
- the *stakeholders* that interact with the organization.
- the results that the organization offers (*products* and *services*, which are described using service profiles). In case of services, a specific implementation of the service can be defined in the *offers* relationship (*ServiceImplementation* attribute), which may be registered in a service directory (*RegisterPort* attribute), so then other entities can find it. Conditions for controlling this registration process can also be specified (*RegisterCondition* and *DeregisterCondition* attributes).
- the services that are *required* by the organization. This "requires" relationship is similar to a job offer advertising of human organizations, in the sense that it represents a necessity of finding agents capable of providing these required services as members of the organization.
- the organization needs from its providers (*Consumes* relationship).
- the *roles* that the organizational unit may play inside other OUs (*Plays* relationship), when considered as a unique entity. *ActivationCondition* and *LeaveCondition* attributes of this relationship indicate in which situation an OU acquires or leaves a role.

Relationship with Gormas metamodel :
It includes Gormas external functionality and mission views.This view allows defining the global goals of the organizations and the functionality that the organizations provide and require from their environment. The entity Mission goal has been removed because it can be modeled as a Functional objective. Moreover, some not necessary relationships has been also removed. For example, the relationship that indicates that a stakeholder interact with an organizational unit can be removed because this information is reflected when a stakeholder consumes or others a service that the organizational unit others or consumes.

3.3 Structural view

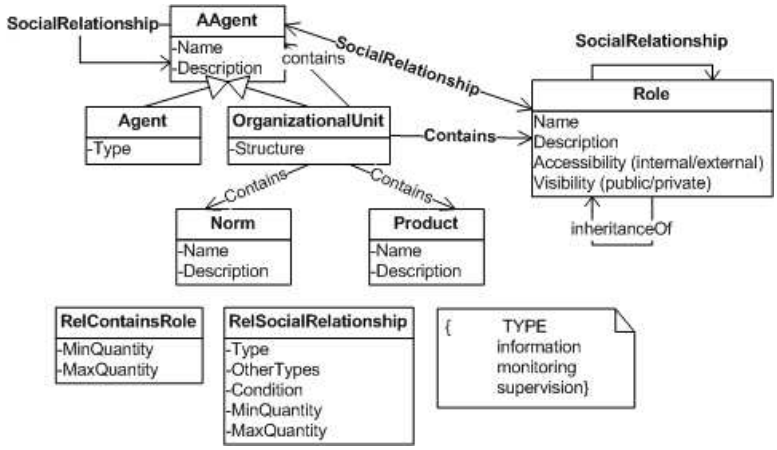

Fig. 6. Emf metamodel Structural View.

The **Structural view** (Figure 6) allows defining the static components of the organization, i.e. all elements that are independent of the final executing entities. More specifically, it defines:

- the entities of the system (*AAgent*), which represent an atomic entity (*Agent*) or a group of members of the organization (*Organizational Unit*), seen as a unique entity from outside.
- the *Organizational Units* (OUs) of the system, that can also include other OUs in a recursive way, as well as single agents.
- the *Roles* defined inside the OUs. In the *contains* relationship, a minimum and maximum quantity of entities that can acquire this role can be specified. For each role, the *Accessibility* attribute indicates whether a role can be adopted by an entity on demand (external) or it is always predefined by design (internal). The *Visibility* attribute indicates whether other entities

can obtain information from this role on demand, from outside the organizational unit (public role) or from inside, once they are already members of this OU (i.e. private role). A hierarchy of roles can also be defined with the *InheritanceOf* relationship.
- the organization *social relationships*. The type of a social relationship between two entities is related with their position in the structure of the organization (i.e. information, monitoring, supervision), but other types are also possible. Moreover, a condition on when this social relationship is active can also be established.
- the *products* available by an OU.
- the *norms* that control the global behavior of the members of the OU.

Relationship with Gormas metamodel :
Structural view includes Gormas structural and social views. This view allows defining the components of the organizations and their social relationships. The reason why these views have been joined, is because most times the type of social relationship between two entities is related with their position in the structure of the organization.

3.4 Role view

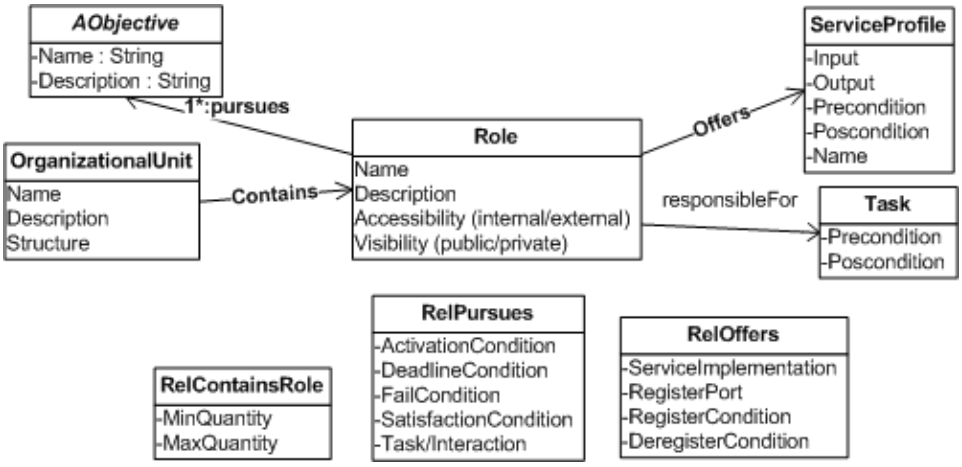

Fig. 7. Emf metamodel Role View

The **Role view** (Figure 7) allows defining the internal functionality of the organizational units and MAS systems, associating the roles with responsibilities, capabilities, objectives and the services that agents playing each role should offer. More specifically, it defines:

– the *goals* (*AObjective*) pursued by a role, which can be *functional objectives* (i.e. softgoalS or non-functional requirements) or operational goals (i.e. hardgoals or *objectives*). In the *pursues* relationship, activation and deadline conditions can be defined to stablish a temporal timeline in which the objective is followed. In case of an operational goal (*objective*), a satisfaction or fail condition can be defined in order to stablish when this objective has been fulfilled.
– the services (*ServiceProfile*) related to the role, i.e., the services that the role is enabled to offer or provide to other entities.
– the *tasks* that the role is responsible for, i.e. the specific functionality that the role is expected to be able to carry out.

Relationship with Gormas metamodel :
Internal functionality view is derived from the Gormas internal functionality view, but the information related to the agent and organizational units tasks and services other have been removed because this information should be expressed in the agent and mission view respectively. This view allows defining the internal functionality of the organizational units and MAS systems associating the roles with responsibilities, capabilities and objectives.

3.5 Agent view

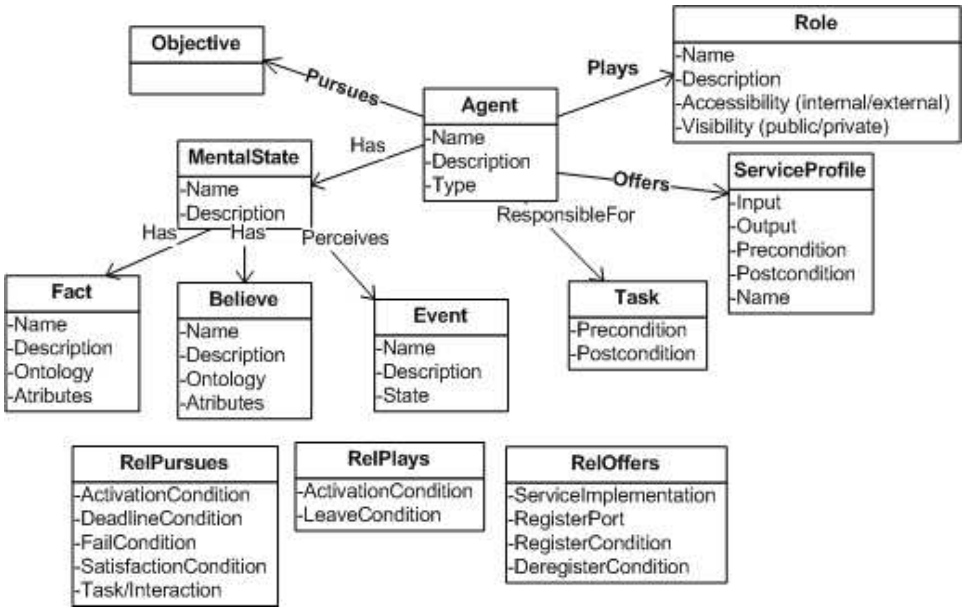

Fig. 8. Emf metamodel Agent View.

The **Agent view** (Figure 8) describes concrete responsibilities of agents and their internal functions, i.e., the agent goals, roles and tasks. The sociability features of agents are represented by means of services, which are implemented through tasks. Thus, an agent is not only capable of executing some tasks, but also of providing a specific functionality (services) to other agents. More specifically, it defines:

- the *objectives* pursued by agents. Activation and deadline conditions can be defined to stablish a temporal timeline in which the objective is followed. Moreover, a satisfaction or fail condition can be defined in order to stablish when this objective has been fulfilled.
- the *roles* played by the agent. Conditions for acquiring and/or leaving the role can be defined.
- the services (*ServiceProfile*) related to the agent, i.e., the services that the agent might offer to other entities. When adopting a role as a member of an organization, the concrete set of services that the agent will be allowed to provide is determined by its own set of offered services and those ones related to the adopted role.
- the *tasks* that the agent is responsible for, i.e. the set of tasks that the agent is capable of carrying out.
- the *Mental States* of the agent, using believes, events and facts.

Relationship with Gormas metamodel :
Agent view is derived from Gormas Agent meta-model. This view describes the agent goals, roles and tasks. Moreover, it can represent the different Mental States of the agent using believes, events, facts and objectives. The difference with Gormas meta-model is that some relationships have been removed due to they have been defined in other views, for example, which goals follow the roles that is defined in the Internal functionality view.

3.6 Objective view

The **Objectives view** (Figure 9) allows defining the objectives decomposition and dependencies. More specifically, it defines:

- The *AObjective* components, which can be *functional objectives* (i.e. softgoal or non-functional requirements) or operational goals (i.e. hardgoals or *objective*).
- The *Functional objectives* represent the expected results of organizational units, which are split into the specific and measurable results that their members are expected to achieve.
- The *Objectives* represent the operational goals, i.e., the specific goals that agents or roles have to fulfill. They can also be refined into more specific objectives. They might be related with a Task or Interaction needed for satisfying this objective.

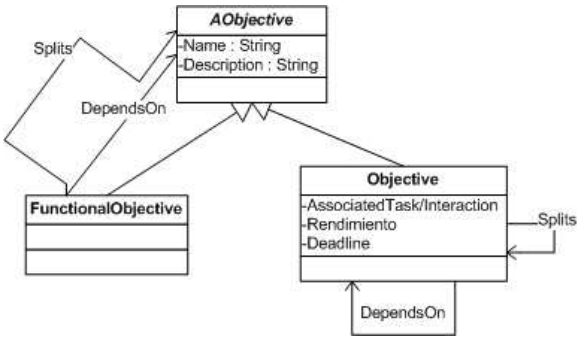

Fig. 9. Emf metamodel Objectives View

Relationship with Gormas metamodel :
Objectives view is derived from the Gormas objectives view and they do not have significant differences. This view allows defining the objectives decomposition and dependencies. Services view is derived from the Gormas Services view and uses some concepts of the Gormas ServicePort view. This view allows defining the service description and shows its composition (defined in the tasks view).

3.7 Task view

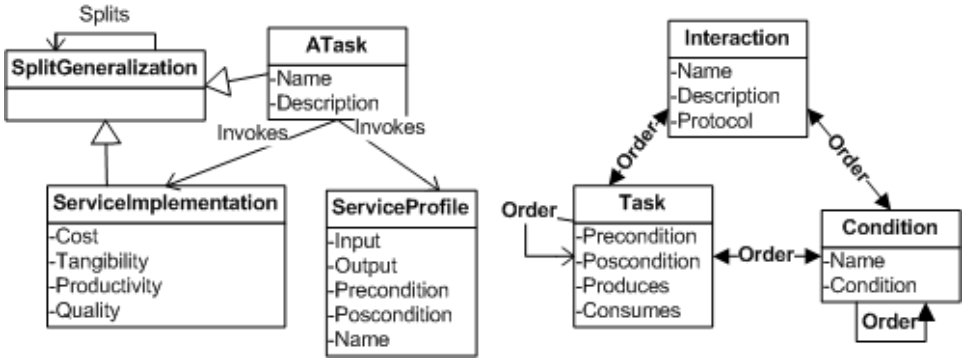

Fig. 10. Emf metamodel Tasks View.

The **Tasks view** (Figure 10) allows defining the functionality of the services and tasks. More specifically, it defines:

– The *ATask* component describes the service functionality and represents both concrete tasks, task-flows or service composition (*Invokes* relationship). This *ATask* component can be split into other ATasks, thus allowing service refinement or task composition.

– A *Task* represents a basic functionality, that consumes and produces changes in the agent's Mental States.
– The *order* relationship between tasks, in which ordering conditions can be defined, as well as interactions. The entity *Condition* allows defining the sequence of tasks depending on a condition.
– The service interface (*ServiceProfile*), which indicates activation conditions of the service (preconditions), its input and output parameters and its effects over the environment (postconditions). It can be lately used in an OWL-S service description.
– The service specific functionality (*ServiceImplementation*), which describes a concrete implementation of a service profile.
– The service composition, by means of *invokations* between services.

Relationship with Gormas metamodel :
Tasks view includes Tasks and Tasks ow views. This view allows defining the functionality of the services and tasks, i.e., which entities they produce and need to be executed. Moreover, it allows defining how services and task can be decomposed into more simple elements and the order of these elements.

3.8 Interaction view

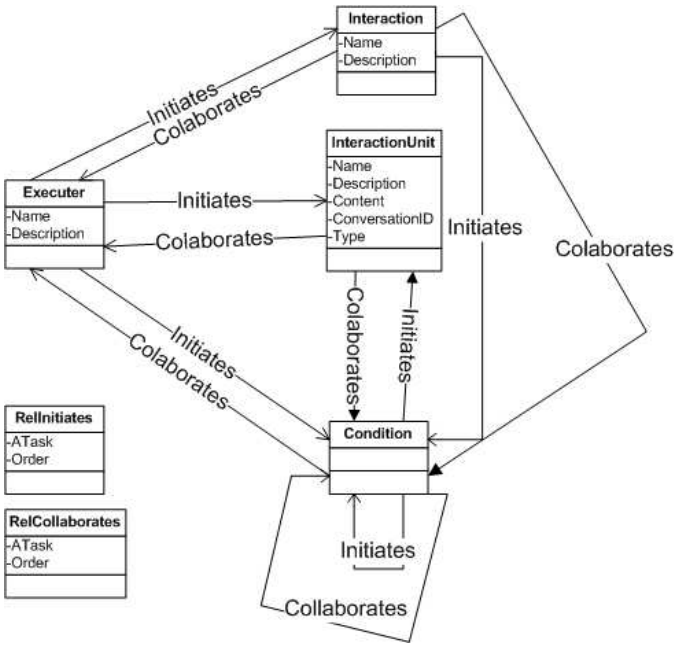

Fig. 11. Emf metamodel Interaction View

The **Interaction View** (Figure 11) allows defining the sequence of interactions. More specifically, it details:

- The participants of the interaction (*Executer*). The *Initiates* and *Collaborates* relationships indicate the sequence of activities (task and services) that have been executed in an interaction. The *Collaborates* relationship represents a response activity.
- The performatives (*InteractionUnit*) employed during the interaction.
- The entity *Condition* allows changing the sequence of interactions depending on a condition.

Relationship with Gormas metamodel :
Interaction view summarizes the concepts of the whole Gormas interaction metamodel without losing expressiveness. This view allows defining the sequence of interactions and which objectives and services are related. Moreover, the entity Condition has been added in order to change the sequence of interactions depending on a condition.

3.9 ServicePort view

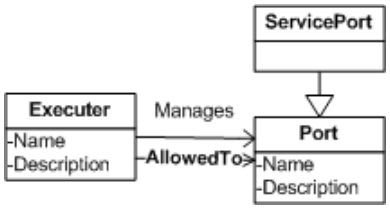

Fig. 12. Emf metamodel ServicePort View.

The **Services Port View** (Figure 12) allows defining the service ports, who is responsible for each port and who can use it. More specifically, it details:

- A *service port*, which is considered as a service publication point that offers the possibility of registering and searching services by their profile.
- The entities (*Executer*) that control the service port (*Manages* relationship), so they can define the usage permissions over this port.
- The entities (*Executer*) that are *allowed to* use the port, for registring new services or accessing existing ones.

Relationship with Gormas metamodel :
Interaction view summarizes the concepts of the whole Gormas interaction metamodel without losing expressiveness. This view allows defining the sequence of interactions and which objectives and services are related. Moreover, the entity Condition has been added in order to change the sequence of interactions depending on a condition.

3.10 Environment view

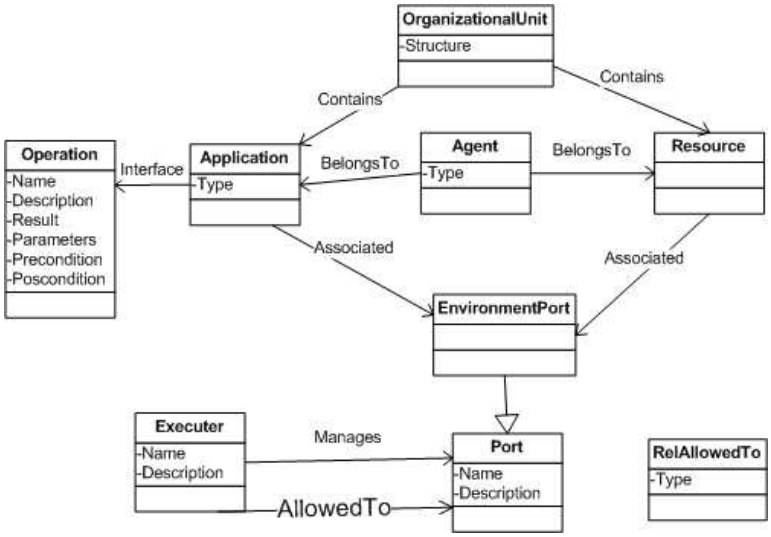

Fig. 13. Emf metamodel diagrams: Environment View

The **Environment View** (Figure 13) describes the environment elements (resources and applications). More specifically, it defines:

- The *products* that are contained in the Organizational Units or that belong to specific agents. They can be applications or resources.
- *Resources* represent environment objects that do not provide a specific functionality, but are indispensable for task execution. They can be consumable or not, have an initial state (*Quantity*), a lower and upper threshold (*MinQuantity*, *MaxQuantity*) and a capacity *granularity*.
- *Applications* represent functional interfaces that are described with a *name*, several *parameters*, *preconditions*, *postconditions* and *results*.
- Products can be associated to *environment ports*, thus describing who can use each resource or application and who is responsible of giving these permissions.
- The *Executer* that controls the service environment (*Manages* relationship) defines the usage permissions over this port.
- The *Executer allowed to* use the port can employ it for perceiving or acting over the associated product.

Relationship with Gormas metamodel :
Environment view is derived from the Gormas Application view and Gormas

Resource view. It includes concepts from the Gormas Port view. This view allows defining the applications. It allows specifying the functionality, who are the owners and how these application can be accessed. Furthermore, this view allows defining resources and specifying the owners and how these services can be accessed.

3.11 Normative view

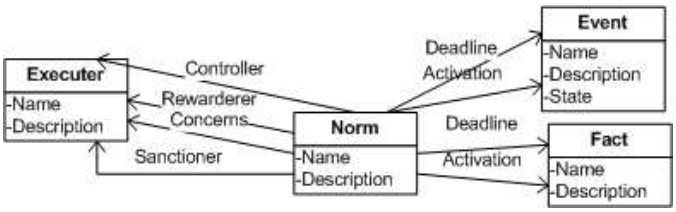

Fig. 14. Emf metamodel diagrams: Normative View

Finally, the **Normative View** (Figure 14) describes normative restrictions over the behavior of the system entities. More specifically, it defines:

- the *Norm* concept, which represents a specific regulation. The properties of the norm detail all facts and events of the environment that provoke the activation or deactivation of the norm.
- The entity (*Executer*) to whom the norm is applied (*Concerns* relationship).
- The *Executer* that is in charge of monitoring the norm satisfaction (*Controller* relationship).
- The *Executer* that is responsible of punishments (*Defender* relationship).
- The *Executer* that is responsible of rewards (*Rewarderer* relationship).
- The *ATask* attribute of all these relationships indicates which task or service will be invoked when monitoring this norm and when punishing or rewarding agents.

Relationship with Gormas metamodel :
Normative view is derived from Gormas Normative meta-model. This view allows defining norms, their goals, their activation and deadline conditions, who control these norms and who are affected by it. As the agent view, the most important difference with the Gormas meta-model is that some relationships have been removed.

References

1. Luck, M., McBurney, P., Gonzalez-Palacios, J.: Agent-based computing and programming of agent systems. In: PROMAS. (2005) 23–37

2. Huhns, M., Singh, M., Burstein, M., Decker, K., Durfee, E., Finin, T., Gasser, L., Goradia, H., Jennings, N.R., Lakartaju, K., Nakashima, H., Parunak, V., Rosenschein, J., Ruvinsky, A., Sukthankar, G., Swarup, S., Sycara, K., Tambe, M., Wagner, T., Zavala, L.: Research directions for service-oriented multiagent systems. IEEE Internet Computing **9**(6) (2005) 52–58
3. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. In: International J. Supercomputer Applications. (2001) 23–37
4. Boella, G., Hulstijn, J., van der Torre, L.W.N.: Virtual organizations as normative multiagent systems. In: HICSS, IEEE Computer Society (2005)
5. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: an Organizational View of Multi-Agent Systems. In Giorgini, P., Muller, J., Odell, J., eds.: Agent-Oriented Software Engineering VI. Volume LNCS 2935 of Lecture Notes in Computer Science., Springer-Verlag (2004) 214–230
6. Argente, E., Julian, V., Botti, V.: Mas modelling based on organizations. In: 9th Int. Workshop on Agent Oriented Software Engineering (AOSE08). (2008) 1–12
7. Soley, R., the OMG Staff Strategy Group: Model driven architecture. Object Management Group White Paper Draft 3.2 (2005)
8. : Eclipse - an open development platform (2008)
9. Argente, E., Julian, V., Botti, V.: Multi-agent system development based on organizations. Electronic Notes in Theoretical Computer Science **150** (2006) 55–71
10. Dignum, V., Dignum, F.: A landscape of agent systems for the real world. Technical report 44-cs-2006-061, Institute of Information and Computing Sciences, Utrecht University (2006)
11. Huhns, M., Singh, M.: Reseach directions for service-oriented multiagent systems. IEEE Internet Computing **Service-Oriented Computing Track. 9**(1) (2005)
12. Trencansky, I., Cervenka, R.: Agent modelling language (AML): A comprehensive approach to modelling mas. In: Informatica. Volume 29(4). (2005) 391–400
13. Ferber, J., Michel, F., Bez-Barranco, J.: Agre : Integrating environments with organizations. Environments for Multi-agent Systems. **3374** (2005) 48–56
14. Gateau, B., Boissier, O., Khadraoui, D., Dubois, E.: Moiseinst: An organizational model for specifying rights and duties of autonomous agents. Environments for Multi-Agent Systems III **4389** (2007) 41–50
15. Pavon, J., Gomez-Sanz, J., Fuentes, R. In: The INGENIAS Methodology and Tools. Volume chapter IX. Henderson-Sellers (2005) 236–276
16. Dignum, V., Vazquez-Salceda, J., Dignum, F.: Omni: Introducing social structure, norms and ontologies into agent organizations. LNAI 3346 (2005)
17. Botti, V., Giret, A.: Anemona. a multi-agent methodology for holonic manufacturing systems. Springer Series in Advanced Manufacturing **XVI** (2008) 214
18. Esteva, M., Rodriguez-Aguilar, J., Sierra, C., Arcos, J., Garcia, P. Lecture Notes in Artificial Intelligence 1991. In: On the Formal Specification of Electronic Institutions. Springer-Verlag (2001) 126–147
19. Argente, E., Julian, V., Botti, V.: Mas modelling based on organizations. In: Proc. AOSE08. (2008) 1–12
20. : Meta object facility (mof) 2.0 query/view/transformation specification. Object Management Group (Document ad 07-07-07)
21. Rougemaille, S., Migeon, F., Maurel, C., Gleizes, M.P.: Model driven engineering for designing adaptive multi-agents systems. (2008) 318–332
22. Perini, A., Susi, A.: Automating model transformations in agent-oriented modelling. In Springer, ed.: In Agent-Oriented Software Engineering VI: AOSE 2005. LNCS (2005)

23. Amor, M., Fuentes, L., Vallecillo, A.: Bridging the gap between agent-oriented design and implementation using mda. In: In Proceedings of the Fifth International Workshop on Agent-Oriented Software Engineering (AOSE 2004). LNCS (2004) 93–108
24. Hachicha, H., Loukil, A., Ghedira, K.: Mamt: an environment for modeling and implementing mobile agents. In: Sixth International Workshop From Agent Theory to Agent Implementation (AT2AI-6). (2008)
25. Morandini, M., Nguyen, C.D., A. Perini, A.S., Susi, A.: Tool-supported development with tropos: the conference management system case study. In: Agent Oriented Software Engineering (AOSE), at AAMAS. (2007)
26. N. Criado and E. Argente and V. Julian and V. Botti: Designing Virtual Organizations, Proc. PAAMS'09, 2009, "Advances in Soft Computing",55,440-449