MAS Modeling Based on Organizations

Estefanía Argente, Vicente Julian, and Vicent Botti[⋆]

Departamento de Sistemas Informaticos y Computacion,
Universidad Politecnica de Valencia,
C/Camino de Vera s/n, 46022, Valencia, Spain
{eargente,vinglada,vbotti}@dsic.upv.es

Abstract. An agent organization model is proposed based on four main concepts: organizational unit, service, environment and norm. These concepts are integrated in ANEMONA meta-models, which are extended in order to include all entities needed for describing the structure, functionality, dynamic, normalization and environment of an organization.

Keywords: Meta-model, Multi-agente systems, Organization.

1 Introduction

Organizational models have been recently used in agent theory to model coordination in open systems and to ensure social order in MAS [1,2]. Agent Organizations rely on the notion of openness and heterogeneity and include the integration of organizational and individual perspectives and the dynamic adaptation of models to organizational and environmental changes [3].

Meta-modeling is a mechanism that allows defining modeling languages in a formal way, establishing the primitives and syntactic-semantic properties of a model [4]. For example, INGENIAS [5] and ANEMONA[6] methods offer several meta-models for MAS analysis and design, by means of their component description (organizations, agents, roles), functionality (goals and tasks), environment (resources and applications), interactions and agent internal features, such as autonomy and mental state processing. INGENIAS follows an iterative development process based on *Rational Unified Process* (RUP). It is supported by powerful tools for modeling, design and code generation. ANEMONA, based on INGENIAS, is a MAS methodology for developing Holonic Manufacturing Systems. They both employ UML notation language for meta-model descriptions. However, they lack of a specific normative description, a deeper analysis of the system dynamics and an open system perspective.

Other MAS frameworks, such as MOISE[7] or E-Institutions[1], do specially focus on the normative specification of the system, but do not take into account the environment description or a more detailed analysis of the organization structure and functionality.

[⋆] This work has been partially funded by TIN2005-03395 and TIN2006-14630-C03-01 projects of the Spanish goverment, FEDER funds and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

In this work, an integration of several methods and modeling languages (such as ANEMONA, AML[8], MOISE, E-Institutions) is proposed for describing the main features of an organization: its structure, functionality, dynamics, environment and norms. In this way, four main concepts are employed: organizational unit, service, norm and environment. These concepts have been extracted from human organizational approaches [9,10,11] and also from multiagent system works [12,8] and service oriented architectures[1]. They are used to represent: (i) how entities are grouped and connected between them and their environment; (ii) which functionality they offer, and which services are employed to manage dynamic entry/exit of agents in the organization; and (iii) which restrictions are needed for controlling entity behavior inside the system.

The proposed MAS modeling employs six different meta-models, which extend the ANEMONA ones: the *organization meta-model*, that describes system entities (agents, organizational units, roles, norms, resources, applications) and how they are related to each other (i.e. social relationships, functionality needed or offered); the *activity meta-model*, that details the specific functionality of the system (services, tasks and goals); the *interaction meta-model*, that defines system interactions, activated by means of goals or service usage; the *environment meta-model*, that describes system applications and resources, agent perceptions and effects and also service invocation through its ports; the *agent meta-model*, that describes concrete agents; and finally the *normative meta-model*, that details organizational norms that agents must follow.

A case-study example based on the travel domain is used to provide a better comprehension of the meta-models. Hotel chains and flight companies offer information about their products (hotels, flights), booking facilities and advance payment. Their functionality is defined using services and it is controlled with norms that describe, for example, which are the minimum services that providers must register in the system in order to participate inside; how services are described (service profiles and processes); or in which order services must be served.

In this paper, the main extensions to ANEMONA meta-models are related, using UML notation language, following GOPPR [13] restrictions. All relationships have a specific prefix that indicates: **O** for organization; **GT** for goals and tasks; **WF** for work flow; **AGO** for social relations; **E** for environment; **N** for norms and **I** for interactions. A *Role* primitive is employed to establish the direction of the relationship, having as suffix an *O* for origin and *D* for destiny. All meta-model extensions are graphically emphasized in dark color. Due to lack of space, only those meta-models with more extensions are explained. More specifically, a description of the *organization meta-model* is detailed in section 2; how services are described using the *activity meta-model* is explained in section 3; extensions to the *environment meta-model* are shown in section 4; whereas section 5 describes how rules are modeled using the *normative meta-model*. Finally, conclusions and discussion are detailed in section 6.

[1] http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf

2 Modeling MAS Organizations

An agent organization is defined as a social entity composed of a specific number of members that accomplish several distinct tasks or functions and are structured following some specific topology and communication interrelationship in order to achieve the main aim of the organization [14]. Agent organizations assume the existence of global goals, outside the objectives of any individual agent, and they exist independently of agents [3].

ANEMONA meta-models offer the *Abstract Agent* (A-Agent) notion [6], which allows defining agent collections as unique entities of a high-level description, modeled as virtual single agents. But A-Agents can be later refined and specified internally, defining all their components (simple agents or groups of agents). Thus, an A-Agent is defined in a recursive way, being an atomic entity or a multi-agent system (with unique entity) composed of A-Agents not necessarily equal.

In this paper, this A-Agent entity has been extended with the **Organizational Unit** (OU) concept, that describes the existing groups of members of the organization. These units have a specific internal structure. They also define several roles or positions that describe a set of functionalities (services offered and required) and goals that represent the organizational expectation for each position. OUs also include resources and applications, that can be accessed by specific members of the organization. And finally, they include all norms that control their members' behavior.

The proposed *organization meta-model* integrates this *Organizational Unit* concept and contains four views: structural, functional, social and dynamic. The first three ones are extensions of those ones employed in ANEMONA, whereas the new dynamic view is used to specify which are the services that an OU must offer to control and manage entry and exit of its entities.

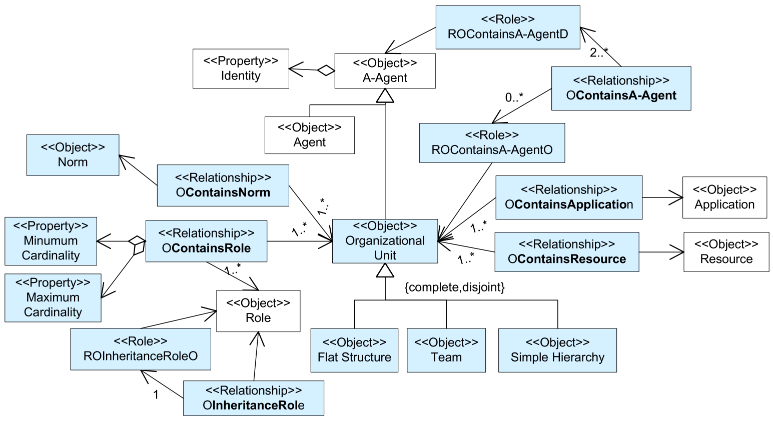

<<Role>> ROContainsA-AgentD
<<Property>> Identity
<<Object>> A-Agent
2..*
<<Relationship>> OContainsA-Agent
0..*
<<Object>> Agent
<<Role>> ROContainsA-AgentO
<<Object>> Norm
<<Relationship>> OContainsNorm
1..*
<<Relationship>> OContainsApplication
<<Object>> Application
<<Property>> Minumum Cardinality
<<Relationship>> OContainsRole
1..*
<<Object>> Organizational Unit
1..*
<<Relationship>> OContainsResource
<<Object>> Resource
1..*
1..*
<<Property>> Maximum Cardinality
{complete,disjoint}
<<Object>> Role
<<Role>> ROInheritanceRoleO
<<Object>> Flat Structure
<<Object>> Team
<<Object>> Simple Hierarchy
1
<<Relationship>> OInheritanceRole

Fig. 1. Organization Meta-model. Structural view.

The **structural view** defines which are the "static" components of the organization, i.e. all elements that are independent of the final executing entities (figure 1). Thus, the system is composed of *Organizational Units* (OU), that can also include other units in a recursive way. Internally, their members are arranged in a hierarchy, team or plain structure. The composition of these units facilitates designing more complex and elaborated structures, such as matrix, federation, coalitions or congregations [14]. The OU acts as a group of agents (*OContainsA-Agent*), but also as their environment. Hence, it contains both resources (*OContainsResource*) and applications (*OContainsApplication*) that can be used by the OUs entities. It also defines the allowed roles inside the unit (*OContainsRole*) and all norms that control their behavior (*OContainsNorm*).

In the travel case study (figure 3.A), the *TravelAgency* organizational unit represents the whole travel system. The *Client* role represents the final user that asks for information on hotels or flights, orders booking rooms or flight seats and even might pay in advance. The *Provider* role offers searching and booking service functionality. Finally, the *Payee* role is responsible for controlling the advance payment. As descriptions and functionalities for travel search and booking services might be rather different for hotels and flights, two organizational units (*FlightUnit* and *HotelUnit*) have been defined, focused on their specific products. In these units, both client and provider roles are specialized into more specific roles (ex. *FlightClient* and *FlightProvider*).

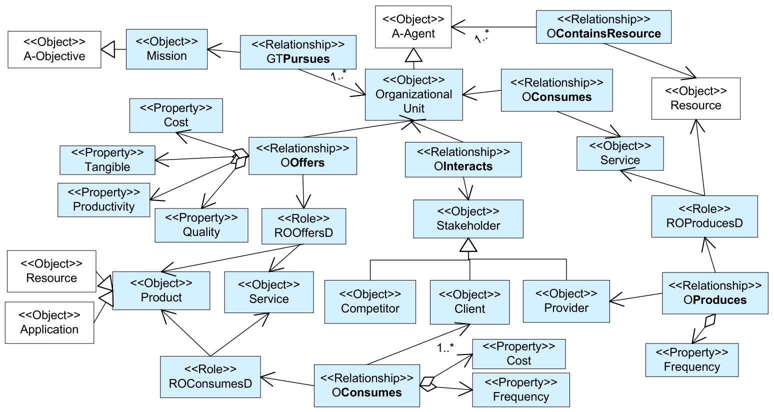


Fig. 2. Organization Meta-model. Functional view. Mission.

The **functional view** describes the organizational mission and how each organizational unit behaves, both externally and internally. It contains three subviews: mission, external functionality and internal functionality.

The *mission* (figure 2) defines organizational global goals (*GTPursues*), who are the stakeholders that interact with the organization (*OInteracts*), which are the results of the organization (*OOffers* products or services), how these results are consumed by its clients (*OConsumes*) and what the organization needs from

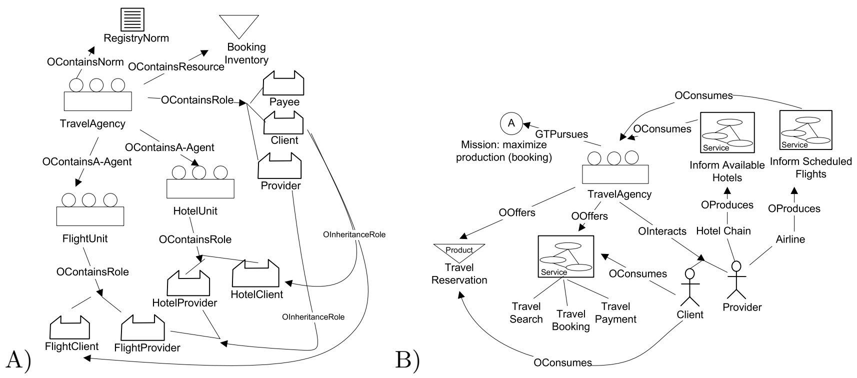


Fig. 3. Example of the organizational model diagram for the travel agency case-study: A) structural view; B) functional view (mission)

its providers (*OProduces* services or resources, *OConsumes* services, *OContainsResource*). In the case-study example (figure 3.B) the system (*TravelAgency* unit) offers the travel reservation product, consumed by its clients (tourists or businessmen). It also offers several services for travel searching, booking and payment. Moreover, this system requires that some providers (hotel chains and airlines) supply all needed information about hotels and flights.

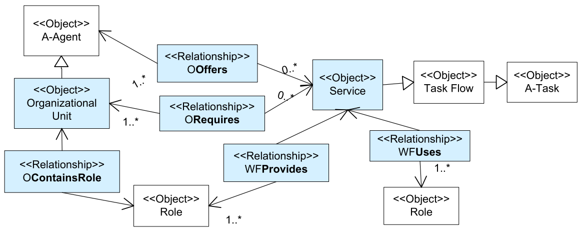


Fig. 4. Organization Meta-model. Functional view. External functionality.

The *external functionality* of an A-Agent (figure 4) represents the set of services that this entity offers to other A-Agents (*OOffers* relationship), independently of the final agent that makes use of them. Moreover, a set of services required by OUs can also be defined. These services represent all functionality that needs to be "hired" to other A-Agents. The *ORequires* relationship is similar to "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 unit. All features, abilities and permissions of providers and clients of these services are modeled by means of roles, using *WFProvides* and *WFUses* relationships. The *OOffers* relationship of this subview is the same one of the *mission* subview, but offered services are more specified. The *ORequires* relation is also related with the *OConsumes* relation of the *mission* subview. In this case, *ORequires* is connected to services that must be provided inside the OU, whereas the *OConsumes* relationship is related to services that are needed by the OU but it is not yet defined whether they are executed inside or outside the organization (i.e. invoke to external entities).

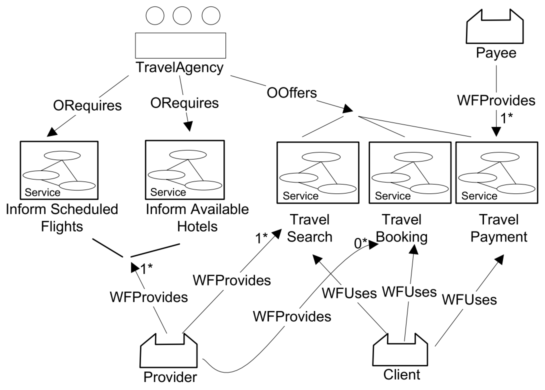


Fig. 5. Example of the Organizational model diagram (external functionality) for the travel case-study

In the travel case-study example (figure 5), the *TravelAgency* unit offers *TravelSearch*, *TravelBooking* and *TravelPayment* services to agents playing the *client* role. Moreover, *provider* agents must supply at least an information service, invoked in the *TravelSearch*. Thus, any agent willing to play a provider role has to be capable of providing a service of this kind. However, the *TravelBooking* service is not compulsory, so providers can freely decide whether to offer it or not. The *TravelPayment* service is assigned to the *Payee* role.

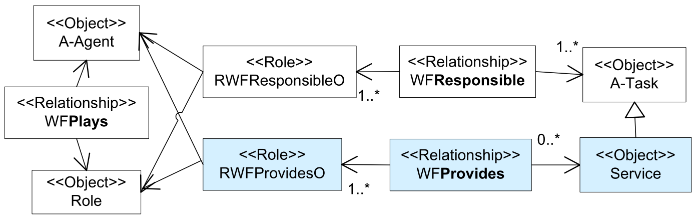


Fig. 6. Organization Meta-model. Functional view. Internal functionality.

Finally, the *internal functionality* of an A-Agent (figure 6) is defined by its tasks (*WFResponsible*), which are delimited by the roles that the entity plays (*WFPlays*) and the services provided by these roles (*WFProvides*). For example, the *Bank* agent (figure 8.A) plays the *Payee* role in the travel case-study, implementing the *TravelPayment* service functionality.

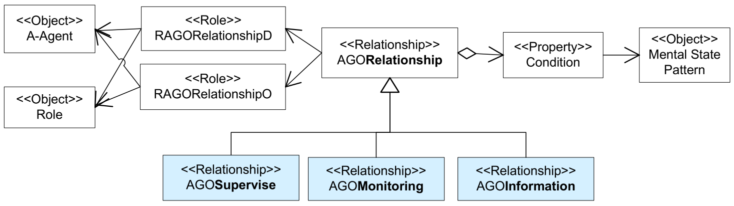

<<Object>> A-Agent
<<Role>> RAGORelationshipD
<<Relationship>> AGORelationship
<<Property>> Condition
<<Object>> Mental State Pattern
<<Object>> Role
<<Role>> RAGORelationshipO
<<Relationship>> AGOSupervise
<<Relationship>> AGOMonitoring
<<Relationship>> AGOInformation

Fig. 7. Organization Meta-model. Social view.

The **social view** (figure 7) describes roles and A-Agent social relationships, divided into three types: supervision, monitoring and information. This social view integrates [15] and [7] works in this meta-model approach.

The *AGOInformation* relationship describes how information or knowledge links are established inside the organization. If two A-Agents are connected by this type of link, then they are entitled to know each other and communicate relevant information. The *AGOMonitoring* relationship implies a monitoring process of agent activity, so the monitor agent is responsible for controlling tasks of its monitored agents. Finally, the *AGOSupervise* relationship implies that a (supervisor) agent transfers or delegates one or more goals to its subordinate agent, which is obliged to include these objectives as its own and pursue them.

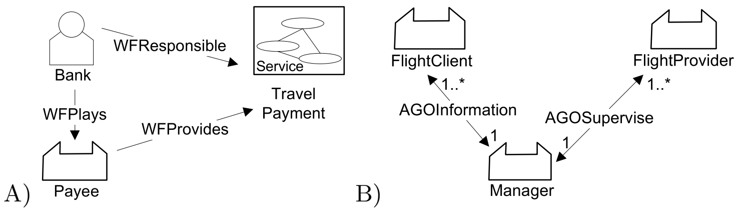


Fig. 8. Example of the Organizational Model diagram for the travel agency case-study: A) internal (functional) view. A *Bank* agents plays the *Payee* role; B) social view.

In the travel case-study example (figure 8.B), the *FlightUnit* has been modeled using a hierarchical structure in which there is a supervisor (*manager* role) that receives all flight requests from clients and invokes *FlightProvider* services, also controlling their behavior.

The **dynamic view** (figure 9) defines the pattern designs for organizational unit services, that enable managing all its structural and dynamic components. These services are divided into structural, informative and dynamic services. The *structural services* are focused on adding or deleting norms, roles or organizational units. The *informative services* provide information about the structure of the organization. And the *dynamic services* manage the inclusion and exit of agents into the unit and the role adoption. These last services need to be published in an open system for allowing external agents to participate inside.

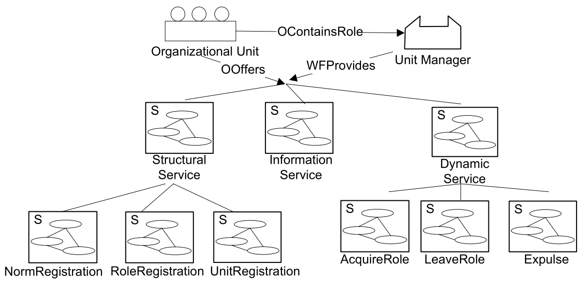


Fig. 9. Pattern Design for the dynamic view (Organization Model)

3 Modeling MAS Services

Services represent a functionality that agents offer to other entities, independently of the concrete agent that makes use of it. Its main features are: (i) synchronization, that implies interaction between entities that offer the service and those ones that require and use it; (ii) publishing, so the service is registered in a service directory and other entities can find it; (iii) participation, i.e. entities that consume the service can differ through time; (iv) entity standardization, as service consumers and providers are related to specific roles, for which restrictions are defined through norms, resource access permissions, etc.; (v) functionality standardization, as services are described in terms of inputs, outputs, preconditions and postconditions, making easier the description its functionality; (vi) tangibility, as services usually produce tangible products which can be employed for evaluating both quality, service efficiency and client satisfaction; and (vii) cost, i.e. service production and consumption imply some costs and/or benefits.

In the **activity meta-model** (figure 10), service system functionality is described by means of its profile and A-Tasks in which a service is split (*WFSplits* relationship). The *ServiceProfile* concept describes activation conditions of the service (preconditions), its input, output parameters and its effects over the environment (postconditions). It can be lately used in an OWL-S service description. The *A-Task* concept (figure 11.A) describes the service functionality. It represents both concrete tasks, task-flows or service composition (*WFInvokes* relationship). A task-flow description (figure 11.B) relates tasks with their environment: usage of resources and mental entities (*WFConsumes*, *WFProduces*), usage of applications (*WFEmploys*), task sequence order (*WFConnects*, *WFInvokes*), task composition (*WFSplits*) and task assignment to agents (*WFResponsible*) and its execution (*WFExecutes*).

The activity model diagram for *TravelSearch* service of the case-study example is shown in figure 12. This service is described using the "Travel Searching" profile and contains four tasks: *CheckPlace*, that checks inputs (country and city); *FlightSearch* and *HotelSearch* (concurrent tasks that invoke *InformScheduledFlights* and *InformAvailableHotels* services, respectively); and *TravelFilter*, that selects the best hotels and flights. The task flow description for the *TravelSearch* service is shown in figure 12.B.

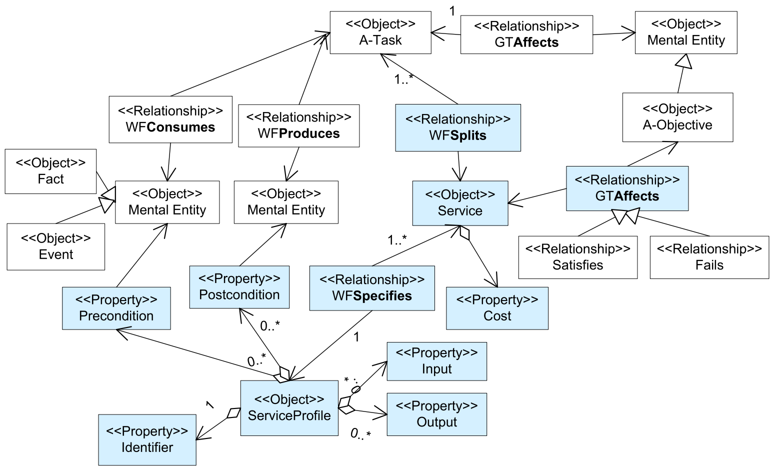

<<Object>> A-Task
<<Relationship>> GTAffects
<<Object>> Mental Entity
<<Object>> A-Objective
<<Relationship>> WFConsumes
<<Relationship>> WFProduces
<<Relationship>> WFSplits
<<Object>> Fact
<<Object>> Event
<<Object>> Mental Entity
<<Object>> Service
<<Relationship>> GTAffects
<<Relationship>> Satisfies
<<Relationship>> Fails
<<Property>> Precondition
<<Property>> Postcondition
<<Relationship>> WFSpecifies
<<Property>> Cost
<<Property>> Input
<<Property>> Output
<<Object>> ServiceProfile
<<Property>> Identifier
1
1..*
0..*
0..*
1
1..*
*
0..*

Fig. 10. Activity Meta-model. Service description.

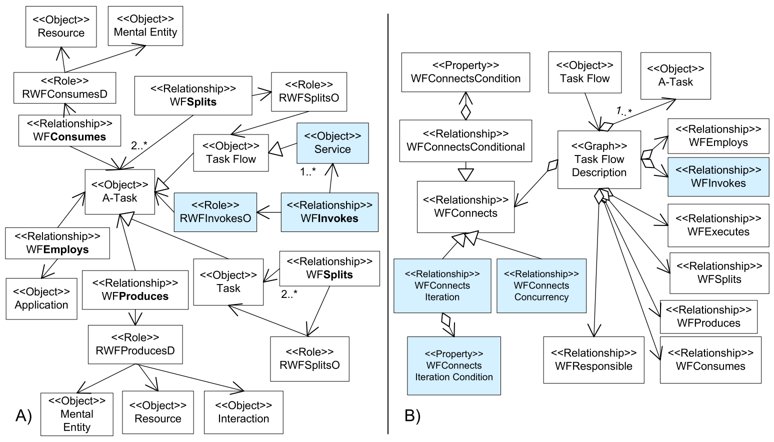

<<Object>> Resource
<<Object>> Mental Entity
<<Role>> RWFConsumesD
<<Relationship>> WFSplits
<<Role>> RWFSplitsO
<<Relationship>> WFConsumes
<<Object>> Task Flow
<<Object>> Service
<<Object>> A-Task
<<Role>> RWFInvokesO
<<Relationship>> WFInvokes
<<Relationship>> WFEmploys
<<Object>> Application
<<Relationship>> WFProduces
<<Object>> Task
<<Relationship>> WFSplits
<<Role>> RWFProducesD
<<Role>> RWFSplitsO
<<Object>> Mental Entity
<<Object>> Resource
<<Object>> Interaction
A)
<<Property>> WFConnectsCondition
<<Object>> Task Flow
<<Object>> A-Task
<<Relationship>> WFConnectsConditional
<<Graph>> Task Flow Description
<<Relationship>> WFEmploys
<<Relationship>> WFInvokes
<<Relationship>> WFConnects
<<Relationship>> WFExecutes
<<Relationship>> WFSplits
<<Relationship>> WFConnects Iteration
<<Relationship>> WFConnects Concurrency
<<Relationship>> WFProduces
<<Property>> WFConnects Iteration Condition
<<Relationship>> WFResponsible
<<Relationship>> WFConsumes
B)
2..*
1..*
2..*
1..*

Fig. 11. Activity Meta-model. A) Task description; B) Task Flow Description.

4 Modeling MAS Environment

Based on human organizations [16,11], environment should be modeled with two different perspectives: structural and functional. The structural perspective describes which are the components of the system (agents, objects, resources), how they are related (i.e. agent groups, behavioral norms, resource access) and how these elements are conceptually represented, by means of an ontology. The functional perspective determines which are the activities related with the environment, i.e., how agent communication is produced (direct or indirect messages, using specific environment elements, etc.), how agents can perceive and act over

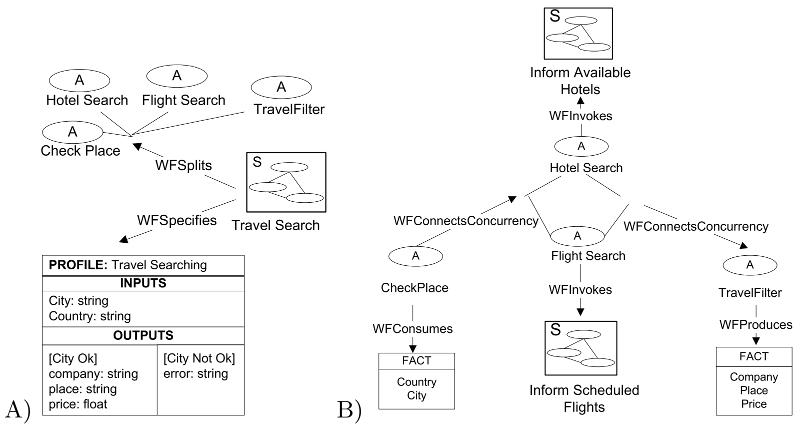


Fig. 12. Activity model diagram for the travel case-study example: A) *TravelSearch* service description; B) *TravelSearch* service task-flow

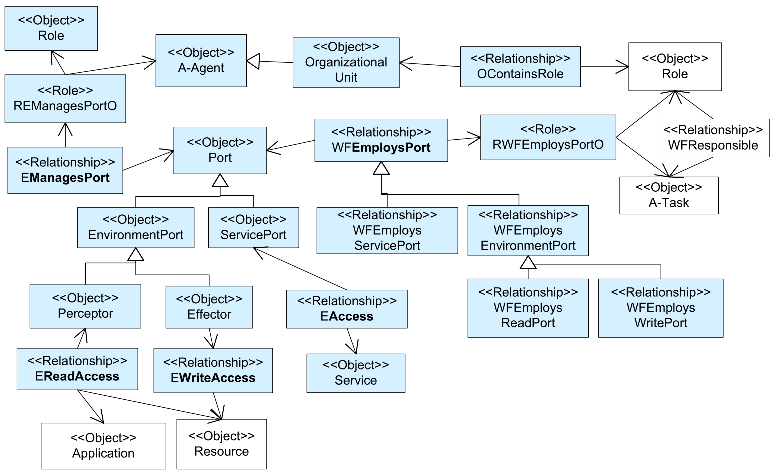
the environment and how agents are connected to other types of entities such as objects, applications or resources.

The proposed *environment meta-model* (figure 13) focuses on the description of its elements (resources, applications and mental entities), perceptions and actions over the environment, and permission accesses for using these elements.

Resources represent environment objects that do not provide a specific functionality, but are indispensable for task execution [5]. They can be consumable or not, have an initial state, a lower and upper threshold and a capacity granularity. As regards applications, they represent functional interfaces that are described with a name, several parameters, preconditions, postconditions and results.

Agent perceptions and actions are described using the *EnvironmentPort* concept, which is a specialization of the *Port* entity. This concept has been extracted from AML language modeling [8], in which a port represents an interaction point between an entity and other model elements. Two kinds of ports have been defined: environment and service ports. The environment port allows lecture and/or write access to resources or applications. The *Perceptor* port establishes how agents can obtain information from resources and applications. The *Effector* port allows agents to modify resource data. The *EManagesPort* relationship indicates who manages and controls the environment port access. The *WFEmploysPort* relationship represents which roles are allowed to use the port and in which way (*WFEmploysReadPort* for obtaining information, *WFEmploysWritePort* for creating or modifying environment information).

The *ServicePort* concept represents the publishing feature of the service, i.e. the contact point or grounding mechanism for service access. The entity in charge of publishing it (in a service directory, for example) is represented with the *EManagesPort* relationship.



<<Object>> Role
<<Object>> A-Agent
<<Object>> Organizational Unit
<<Relationship>> OContainsRole
<<Object>> Role
<<Role>> REManagesPortO
<<Relationship>> WFEmploysPort
<<Role>> RWFEmploysPortO
<<Relationship>> WFResponsible
<<Relationship>> EManagesPort
<<Object>> Port
<<Object>> A-Task
<<Object>> EnvironmentPort
<<Object>> ServicePort
<<Relationship>> WFEmploys ServicePort
<<Relationship>> WFEmploys EnvironmentPort
<<Object>> Perceptor
<<Object>> Effector
<<Relationship>> EAccess
<<Relationship>> WFEmploys ReadPort
<<Relationship>> WFEmploys WritePort
<<Relationship>> EReadAccess
<<Relationship>> EWriteAccess
<<Object>> Service
<<Object>> Application
<<Object>> Resource

Fig. 13. Environment meta-model. Port access.

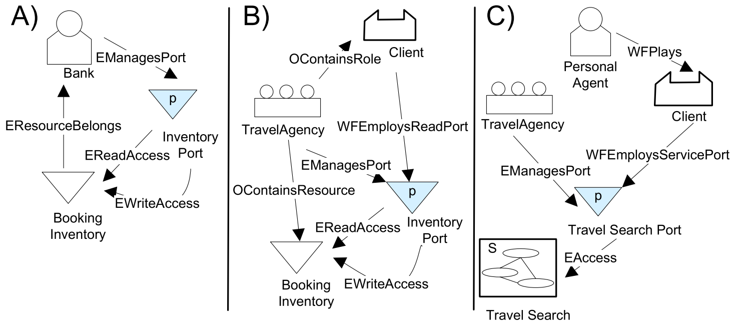


Fig. 14. Case-study example: **A)** the *bank* agent contains the *Booking Inventory* and manages its access; **(B)**The *TravelAgency* unit contains the *Booking Inventory* and manages its access; **(C)** The *TravelAgency* unit publishes *TravelSearch* service, which is used by agents playing the *client* role

For the travel case-study, an example of a resource belonging to a specific agent is shown in figure 14.A, in which the *Bank* agent controls access to the *Booking Inventory* by means of the *Inventory Port*. However, in many problems the resource does not belong to a specific agent, but to the environment of a group of agents. In this case, the organizational unit that represents this group contains this resource and manages its access through a resource port. For example, in figure 14.B, the *TravelAgency* unit contains the *Booking Inventory* resource, which can be read or modified, but the *client* role defined in this unit is only empowered to read access. Finally, an example of a service port access is shown in figure 14.C, in which a *PersonalAgent* playing the *client* role is allowed to make use of the *TravelSearch* service. The *TravelAgency* unit is in charge of publishing this service (represented by the *EManagesPort* relationship).

5 Modeling MAS Norms

Norms have been widely used as mechanisms to limit human autonomy inside societies, in order to solve problems of coordination, specially when total and direct social control cannot be exerted. In open multi-agent systems, norms have been considered as a key issue for managing the heterogeneity, autonomy and diversity of interests of agents [17].

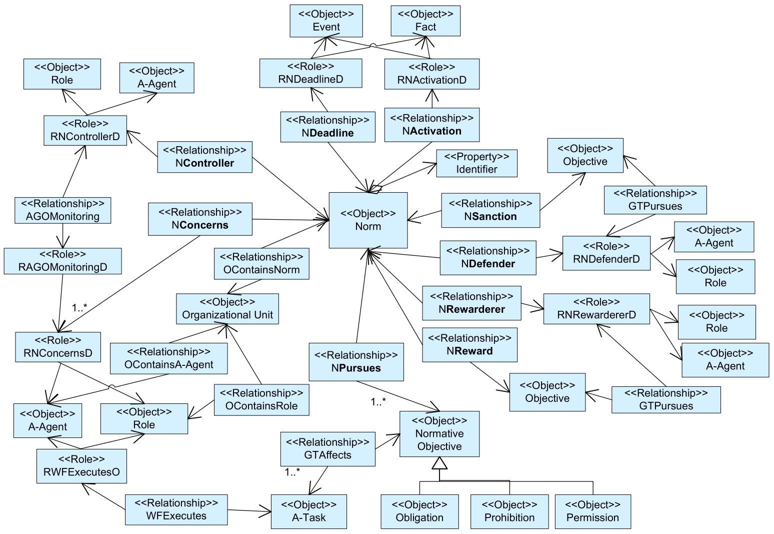


Fig. 15. Normative meta-model

The proposed **normative meta-model** (figure 15) describes the **Norm** concept, which represents a specific regulation, expressed by means of a *Normative Objective* (Obligation, Permission or Prohibition). This regulation affects or concerns A-Agents or Roles (*NConcerns* relationship), whose actions are controlled by the normative objective (*WFExecutes* and *GTAffects* relationships). The norm also indicates who is in charge of monitoring that the norm is satisfied (*NController* relationship) and who is responsible for punishments (*NSanction* and *NDefender* relationships) and/or rewards (*NReward* and *NRewarderer* relationships). Finally, *NActivation* relationship specifies all facts and events of the environment that provoke the activation of the norm. Its deactivation (*NDeadline*) is produced when the normative objective or the deadline is satisfied.

In figure 16, the pattern design for an obligation norm is shown. A sanction is created when the deadline has been reached and compulsory tasks have not been satisfactorily executed yet.

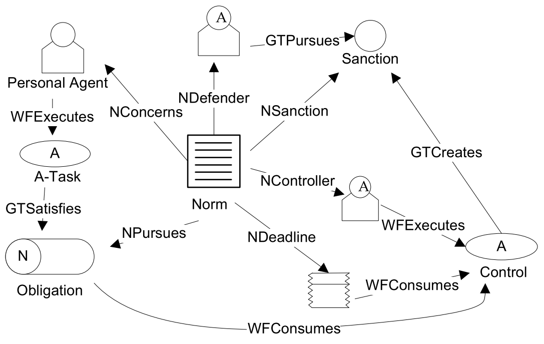


Fig. 16. Pattern design of an obligation

6 Discussion

An extension of ANEMONA meta-models has been proposed, in order to include concepts of organizational unit, service and norm. These concepts have been extracted from human organizational approaches, from multiagent systems works and from service oriented architectures, being integrated in a framework for modeling organizations. In this way, the main features of an organization can be described: its structure, functionality, dynamics, environment and norms. Thus, the *organization meta-model* describes its components, relationships and connections with its environment. The *activity meta-model* details offered and required services, their tasks and objectives. The *environment meta-model* captures system resources and applications, agent perceptions and effects and port accesses permissions. Moreover, organization rules are expressed with the *normative meta-model*. Finally, the *agent meta-model* details concrete responsibilities of agents and their internal functions, and the *interaction meta-model* defines specific interactions between agents and service invocation (using service ports). These two last meta-models have not been included in this paper due to lack of space, but they are mainly the same as in ANEMONA.

Regarding related work on MAS organizational modelling, there are different interesting approaches, standing out AGR [18], MOISE [7], ODML [20], AML [8], OMNI [21] and OMACS [23]. The AGR model [18] is based on agent, groups and role concepts. It was lately extended in the AGRE[19] work (E for environment).

MOISE^{Inst} model [7] includes structural, functional and deontic views. Its structural view is related with our organization meta-model, detailing roles, groups and relationships. In our proposal, the environment is also modelled and the internal topology of groups is considered as well. Its functional view describes plans and missions to achieve goals, similarly to our A-Objectives. In our approach, services required and offered are modelled too, and agent interactions are deeply described in the interaction meta-model. Finally, MOISE^{Inst} deontic view describes permissions and obligations of roles, including sanctions. Our normative meta-model also incorporates rewards.

ODML [20] uses a basic underlying model of organizations for performance prediction of the multiagent organization. Their existing organizational models [12] have served as a basis for our topological analysis [14]. AML [8] extends UML with agent concepts, including resources, environment, organizational units and services, but it lacks of a normative modeling. Our proposal has adopted AML environment perspective, using ports for accessing services and resources. Moreover, our meta-models are integrated in an iterative process, such as in INGENIAS or ANEMONA methodologies.

OMNI [21] offers Normative, Organizational and Ontological Dimensions. It makes use of ISLANDER [1] and AMELI [22] framework for MAS implementation. The mission of the organization, its norms and rules, roles, groups and concrete ontological concepts are detailed. It is also based on contracts, used for acquiring roles and controlling agent interactions. In our proposal, these contract specifications can be employed to better define the organizational services in the dynamic view of the organization meta-model.

Finally, OMACS [23] defines a meta-model for MAS that allows the system to design its own organization at runtime. It is based on agent capabilities (similar to our agent meta-model, in which tasks and services that an agent is responsible for are defined), role assignments (described in our organization meta-model) and policies, which include behavioral and reorganization policies (defined in our normative meta-model) and assignment policies (described in our organization and environment meta-models using access restrictions on resources and services).

The Organizational MAS modeling approach presented in this paper has been integrated in an iterative process of system development, in which several methodological guidelines are employed for describing the mission of the organization, its productive tasks and processes, its organizational dimensions and topological structure, its decision and information processes, its dynamics and normative behavior and its reward system. Moreover, a BNF language for describing norms has been developed. It allows defining restrictions on service usage, registration and provision. Furthermore, a graphical development tool is currently being implemented, that helps designers with diagram model construction and automatic code generation.

References

1. Esteva, M., de la Cruz, D., Sierra, C.: ISLANDER: an Electronic Institution Editor. In: Proc. AAMAS 2002, pp. 1045–1052 (2002)
2. Dignum, V., Meyer, J., Wiegand, H., Dignum, F.: An organization-oriented model for agent societies. In: Proc. RASTA 2002 (2002)
3. Dignum, V., Dignum, F.: A landscape of agent systems for the real world. Tech. Report Inst. Information and Computer Sciences, Utrecht. Univ. (2006)
4. van Gigch, J.P.: System Design Modeling and Metamodeling. Plenum Press, New York (1991)
5. Gomez, J., Fuentes, R., Pavon, J.: The INGENIAS Methodology and Tools. In: Agent-oriented Methodologies, pp. 236–276. Idea Publishing Group, USA (2005)
6. Botti, V., Giret, A.: ANEMONA: A multi-agent methodology for Holonic Manufacturing Systems. Series in Advanced Manufacturing. Springer, Heidelberg (2008)

7. Gateau, B., Boissier, O., Khadraoui, D., Dubois, E.: Moise-inst: An organizational model for specifying rights and duties of autonomous agents. In: Proc. CoORG (2005)
8. Cervenka, R., Trencansky, I.: AML. The Agent Modelling Language. Whitestein Series in Soft. Agent Tech. and Autonomic Computing. Birkhäuser, Basel (2007)
9. Robbins, S.: Organizational Behavior. Pearson Prentice Hall (2007)
10. Moreno-Luzon, M., Peris, F., Gonzalez, T.: Gestión de la Calidad y Diseño de Organizaciones. Prentice Hall, Pearson Education (2001)
11. Hodge, B.J., Anthony, W., Gales, L.: Organization Theory: A Strategic Approach. Prentice Hall, Englewood Cliffs (2002)
12. Horling, B., Lesser, V.: A survey of multiagent organizational paradigms. The Knowledge Engineering Review 19, 281–316 (2004)
13. Kelly, S., Lyytinen, K., Rossi, M.: MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) CAiSE 1996. LNCS, vol. 1080, pp. 1–21. Springer, Heidelberg (1996)
14. Argente, E., Palanca, J., Aranda, G., Julian, V., Botti, V., Garcia-Fornes, A., Espinosa, A.: Supporting Agent Organizations. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS, vol. 4696, pp. 236–245. Springer, Heidelberg (2007)
15. Grossi, D., Dignum, F., Dastani, M., Royakkers, L.: Fundations of organizational structures in multiagent systems. In: Proc. AAMAS 2005, pp. 690–697 (2005)
16. Weyns, D., Parunak, H., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 1–47. Springer, Heidelberg (2005)
17. López, F., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. Computational and Mathematical Organization Theory 12, 227–250 (2006)
18. Ferber, J., Gutkenecht, O., Michel, F.: From agents to organizations: an organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
19. Ferber, J., Michel, F., Baez, J.: AGRE: Integrating Environments with Organizations. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS, vol. 3374, pp. 48–56. Springer, Heidelberg (2005)
20. Horling, B., Lesser, V.: Using ODML to model multi-agent organizations. In: Proc. IEEE/WIN/ACM INt. Conf. on Intelligent Agent Technology, pp. 72–80 (2005)
21. Dignum, V., Vazquez, J., Dignum, F.: OMNI: Introd. social structure, norms and ontologies into agent organizations. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS, vol. 3346, pp. 181–198. Springer, Heidelberg (2005)
22. Esteva, M., Rosell, B., Rodriguez-Aguilar, J.A., Arcos, J.: AMELI: An Agent-based Middleware for Electronic Institutions. In: AAMAS 2004, pp. 236–243 (2004)
23. DeLoach, S., Oyenan, W., Matson, E.: A capabilities-based model for adaptive organizations. Auton. Agent Multi-Agent Syst. 16, 13–56 (2008)