Chapter 9
OperA/ALIVE/OperettA

Huib Aldewereld, Sergio Álvarez-Napagao, Virginia Dignum, Jie Jiang,
Wamberto Vasconcelos, and Javier Vázquez-Salceda

9.1 Introduction

Comprehensive models for organizations must, on the one hand, be able to specify global goals and requirements but, on the other hand, cannot assume that particular actors will always act according to the needs and expectations of the system design. Concepts as organizational rules [46], norms and institutions [12], [15], and social structures [34] arise from the idea that the effective engineering of organizations needs high-level, actor-independent concepts and abstractions that explicitly define the organization in which agents live [46].

The OperA framework takes these distinctions between organization and individuals, constraining and autonomy, social and selfish, as principal keystones of organizational design. Organizations are something more than the people that act in it; e.g., the organization of the University of Delft has been around since 1842, even though the people (lecturers, deans, students, support staff, etc.) that work in it today are very different from those that worked in it then.

In OperA, we take formal processes and requirements as a basis for the modeling of complex systems that regulate the action of the different agents. Organizational models must enable the explicit representation of structural and strategic concerns and their adaptation to environment changes in a way that is independent from the behaviours of the agents. The deployment of organizations in dynamic and unpredictable settings brings forth critical issues concerning the design, implementation,

Huib Aldewereld, Virginia Dignum, Jie Jiang
Delft University of Technology, Delft, The Netherlands
e-mail: {h.m.aldewereld, m.v.dignum, j.jiang}@tudelft.nl

Sergio Álvarez-Napagao, Javier Vázquez-Salceda
Universitat Politecnica de Catalunya, Barcelona, Spain
e-mail: {salvarez, jvazquez}@lsi.upc.edu

Wamerberto Vasconcelos
University of Aberdeen, Aberdeen, United Kingdom
e-mail: w.w.vasconcelos@abdn.ac.uk

and validation of their behaviour [18, 35, 42], and should be guided by two principles:

- Provide sufficient representation of the institutional requirements so that the overall system complies with the norms.
- Provide enough flexibility to accommodate heterogeneous components.

Therefore, organizational models must provide means to represent concepts and relationships in the domain that are rich enough to cover the necessary contexts of agent interaction while keeping in mind the relevance of those concepts for the global aims of the system. The OperA model [11] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it. That is, OperA enables the specification of organizational structures, requirements and objectives, and at the same time allows participants to have the freedom to act according to their own capabilities and demands.

The explicit distinction between organization and agent, and a clear separation of each of their concerns, permeats also into the extentions of the OperA framework that have been made. Since the organization is considered as separate from the agents, it has to be checked and verified (at design-time and at run-time) whether the coordinated set of agents bring about the objectives of the organization.

The OperA framework consists of three interrelated models. The Organizational Model (OM) is the result of the observations and analysis of the domain and describes the desired behaviour of the organization, as determined by the organizational stakeholders in terms of objectives, norms, roles, interactions, and ontologies.

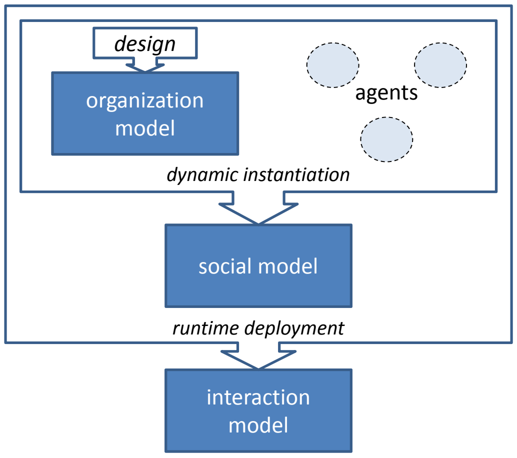


Fig. 9.1 The OperA development process.

The OM provides the overall organization design that fulfills the stakeholders requirements. Objectives of an organization are achieved through the action of agents, which means that, at each moment, an organization should employ the relevant agents that can make its objectives happen. However, the OM does not specify how to structure groups of agents and constrain their behaviour by social rules such that their combined activity will lead to the desired results. The Social Model (SM) maps organizational roles to agents and describes agreements concerning role enactment and other conditions in social contracts. Finally, the Interaction Model (IM) specifies the interaction agreements between role-enacting agents as interaction contracts. IM specification enables variations to the enactment of interactions between role-enacting agents.

9.1.1 Brief History

The OperA modelling framework was developed as a result of the PhD thesis of Virginia Dignum [11]. The aim of the model then was to describe organizational interactions in knowledge management for insurance companies. The model has since evolved through various projects and interactions. In [41], Dignum, Dignum and Vazquez extended the OperA modelling framework with additional normative elements taken from HarmonIA [39]. In [13] Dignum and Dignum formulated a formal logic for organizations based on the principles and ideas of the OperA framework.

In [6] the framework was extended with runtime components. Up to then, most efforts were taken on the modelling aspect of organization design (i.e., the OM). The ALIVE project allowed for the creation of an integrated modelling toolkit (OperettA [4]) and the use of OperA models in the development of flexible, dynamic service-oriented systems. The former created a user-friendly interface for the specification and verification of OperA OM models, the latter explored (runtime) components of the SM and IM models. These runtime components resulted in an integrated service software development kit (ALIVEclipse[1]). Ideas from the ALIVE framework resulted in a number of additional elements of the OperA framework, namely: a Norm Monitoring framework [8] and a Normative Planner [33]. Recently, [26, 28] extended the OperA framework with the conceptualisation of organizations of organizations (OperA+ framework) and organizational compliance verification (CCCP toolkit). Finally, Jenssen and Dignum recently started the development of agent-oriented organizational reasoning mechanisms for the development of OperA-aware agents (AORTA [22]).

[1] ALIVEclipse and accompanying documentation can be downloaded from http://ict-alive.sourceforge.net/.

9.1.2 Applications

OperA has been applied to a number of different domains and applications:

- Support for systems for knowledge management that incorporate the management of knowledge assets with the facilitation and encourgement of interaction between people in an open environment [11];
- Models for scenario development to develop and evaluate organizational/corporate strategy [32];
- Analysis and design of inter- and intr-organizational interaction in the maintenance systems of the Dutch railways [28];
- Simulations of crisis scenarios for the purpose of critical infrastructure and control mechanisms [36, 7];
- Improving flexibility and modularity of control software for warehouse management planning/scheduling systems [21, 2, 3];
- Process and regulation compliance verification for Customs authorities in International Container Trade [27, 26]
- Formalisation and implementation of adaptive serious games using multi-agent organizations [45, 43, 44]
- OperA has also been used in a day-long session with research staff of the project ”International Technology Alliance in Network and Information Sciences”[2]; the aim was to facilitate information and knowledge sharing among coalitions of military forces.
- Formalization of improvisation theatrical performances (formalising dynamic interplay between actors on a high level of abstraction) [24].

9.2 Metamodel

In this section we present the metamodel of OperA / ALIVE, with the aim of comparing to the other framework presented in this book. A formal meta-model, based on OMG MOF [20], of the Organizational Model of OperA, and parts of the ALIVE framework have been created as part of the ALIVE project and can be found in [6] and on the ALIVE website: http://ict-alive.sourceforge.net.

9.2.1 Overview

OperA consists of three interrelated models, OM, SM, IM. Organizational Models (OM) are the main specification of the purposes (aims, objectives, means) of an organization to function. The OM is composed of four main elements, shown in Figure

[2] https://www.usukita.org/.

9.2 below; the social structure (SS), the interaction structure (IS), the communicative structure (CS), and the normative structure (NS). The social structure defines the parties involved in an organization, the relations between these parties, and the objectives of each of these parties (and furthermore, the relations between these objectives). The interaction structure is a second part of organizational models that defines the patterns of interaction between the various parties involved in the organization. These patterns of interaction are grouped in small (on their own standing) bits called scenes. The interaction structure defines the ordering of these scenes and how roles traverse through the scenes to reach their (and the organizations) objectives. The interactions specified in the interaction structure require communication between the roles. The vocabulary of these communications (ontology) and the various different formulas used in other parts of the organizational model are defined in the Communicative Structure. Norms are an important aspect of organizations as they prescribe how the roles are to act within the organization. The norms of an organization are defined in the Normative Structure. The metamodels of the different structures in an Organizational Model can be found in Appendix A.

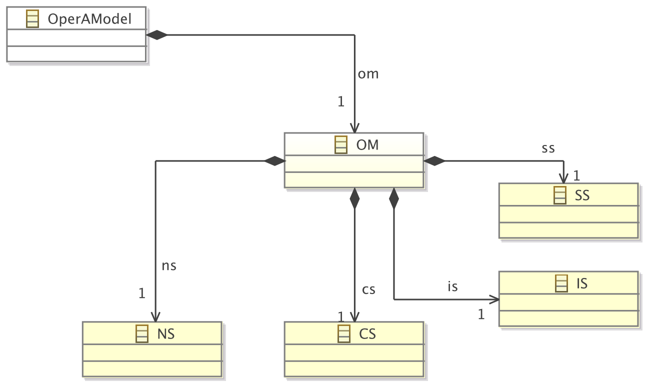


Fig. 9.2 Meta-model: Organisational Model (OM).

9.2.2 *Assumptions*

In OperA, the concept of organization reflects the idea that interactions occur not just by accident, but aim at achieving some desired global goals. That is, there are goals external to each individual participant (or agent) that must be reached through their interaction. Desired behaviour of the organization is therefore external to the participants and must be guaranteed by the organizational structure. However, assuming open environment where neither the internal architecture nor the actual aims of the agents can be verified, such guarantees on desired global behaviour should be achieved without relying on the design of agents nor compromising the agents autonomy [39].

A consequence of this view is that organizational structure is independent from that of the participants, and determined by the designer of the organization. From an organizational perspective, the main function of an individual agent is the enactment of a role that contributes to the global aims of the organization. Organization goals determine which roles and interaction norms must be specified. Agents are then seen as the actors that perform role(s) described by the organization design.

However, the very notion of agent autonomy refers to the ability of individual agents to determine their own actions, plans and beliefs. From an agents perspective, its own capabilities and aims determine the reasons and the specific way an agent will enact its role(s), and the behaviour of individual agents is motivated from their own goals and capabilities [10, 42]. That is, agents bring in their own ways into the society, in that they will follow their own goals and motivations and will bring in their own ways of doing things in the system. In other words, the actual behaviour of the society emerges from the goal-pursuing behaviour of the individual agents within the constraints set by the organization.

These considerations lead to the two pillars of the OperA approach [42]:

Internal autonomy: interaction and structure of the organization must be represented independently from the internal design of the agents.

Collaboration autonomy: activity and interaction in the organization must be specified without completely fixing in advance all interaction possibilities.

The first requirement relates to the fact that since an open organization allows the participation of multiple heterogeneous entities, the number, characteristics and architecture of which are unknown to the designer, the design of the organization cannot be dependent on their individual designs. The second requirement highlights the fundamental tension between the goals of the organization and the autonomy of the participating entities. On the one hand, the more detail about interactions provided by organization design, the more requirements can be checked and guaranteed at design time. This allows, for example, to ensure that certain rules are always followed. On the other hand, there are good reasons to allow the agents some degree of freedom, basically to enable their autonomy to choose their own way of achieving collaboration, and as such increase their flexibility and adaptability.

Taken together, these requirements enable the realization of the objectives of an organization without ignoring the individual aims and personalities of the autonomous participant agents.

Different degrees of abstraction on organization specification have consequences to the level of autonomy required from each agent to accomplish organizational objectives. Intuitively, the more detailed specification given, the less alternative ways (and thus less autonomy) are available for the agents to achieve the organizational objectives. On the other hand, abstract organization models require more autonomy and reasoning capabilities from the agents as they will need to be able to interpret the specification and decide how to coordinate with each others in each situation. This, however, provides higher flexibility of operation. It is then a design decision how to better regulate agent autonomy by deciding on the level of abstractness or concreteness of the organizational description.

9.2.3 Main Constructs

The following details the main elements of OperA with regards to the comparison table, as presented in a later chapter. We discuss each of the elements of that table separately.

Ontology

OperA distinguishes between two different ontologies: the OperA ontology (containing all elements described above as modelling concepts) and the domain ontology (containing all the concepts related to and used by the organizational model). As ontologies do not provide operational semantics for actions, actions in the ontology have an additional operational description in an action language (e.g., PDDL).

Atomic actions

On the specification level, i.e., the OM of OperA, OperA does not represent atomic actions, but rather specifies results (required/desired states of affairs). There is no notion of activity in the Organizational Model, but ordering and importance of states. How the roles (enacted by the agents) move from state to state is not specified in the OM. In the Social Model (SM) and Interaction Model (IM), however, the agents propose, by means of a contract, how they will enact a role, and which actions (or tasks) they will perform to achieve the role objectives.

Activities

The relations between states is represented on two separate levels in the Interaction Structure of the Organizational Model. On the top level, a distinction is made in to pieces of interaction that have a meaning on their own; these are called scenes (see Subspaces and their interrelation below).

The second level of activity specification in OperA is through the use of Landmark Patterns. Each scene has a Landmark Pattern that describes the protocol to follow in the scene on a high level of abstraction. The landmark pattern consists of the important states that should happen in the running of the scene (landmarks) and the order in which these states should follow each other (partial ordering).

Subspaces and their interrelation

As mentioned above, interaction/activity in OperA is divided into pieces of interaction that have a meaning on their own, called scenes. The scenes are related (in order/parallel) through the use of transitions (which serve as flow control operators and access control operators). OperA distinguishes 3 types of transitions:

- AND: all scenes leading into this transition need to be finished before agents can proceed. All scenes after this transition need to be visited in parallel.
- OR: the scenes leading into this transition do not need to finish at the same time (but they can), and a choice can be made in visiting the scenes that follow this transition.
- XOR: only one of the scenes leading to this transition can finish at the same time, and only one of the scenes following this transition can be chosen for a visit.

Interactions within the scenes are described by a Landmark Pattern (see above).

Ubiquity and concurrency

Agents can be in multiple scenes at the same time; access to (and from) the scenes is regulated by the transition norms that are imposed on the transitions that connect the scenes. There are three different types of scenes in OperA:

- single instance scenes: only one of these can be present at a time.
- multiple instance scenes: many can happen at the same time; sometimes bounded by an upper limit.
- persistent scenes: scene cannot be finished, and will exist as long as the organization exists.

Coordination devices

The coordination between the scenes is done via transitions (equipped with transition norms), see above. Coordination between the agents is assumed through the fact that particular roles are required to be together in a scene for it to play out. However, this coordination between the roles/agents is not strictly enforced by the model.

Social and organizational arrangements

OperA abstracts from individual actors, and only describes the social/organizational arrangements. OperA uses roles for this, which can be related (through dependencies). There are three types of dependencies: market, hierarchy, and network.

Incompatibility between roles is handled through the norms. Roles can be related into collective identities through the use of Groups. All roles in the group share the same objective, right, or dependency.

The dynamics of the interaction between roles is implicitly specified in the type of dependency used. Market dependencies between roles typically lead to a Call-for-Proposal type of interaction. Hierarchy dependencies typically leads to a delegation type of interaction. Network dependencies, finally, lead to coordinated action type of interaction.

The organization among the roles is structured through the use of dependencies. Roles can have attached Rights, which are special abilities (capabilities, like marrying people) granted to the agents enacting that role.

Regulatory system

There are three types of norms in OperA:

- **Restricted Norms** norms that cannot be violated. These are incorporated into the structure of the model; e.g., objectives should be achieved, particular orderings have to be followed in the interaction structure, for the organization to advance. Transition norms are a part of this; they specify how to move through the IS, but cannot be violated.
- **Regulative norms** explicit norms represented in the normative structure, specifying soft constraints on the behaviour of the agents. These norms can be violated, which might lead to sanctions (sanction are represented as a second-level norm, activated when the original norm was violated [17]).
- **Constitutive norms** the communicative structure includes the possibility to specify the constitutive elements of the organization through counts-as rules [1].

The first two types of norms in OperA are modelled as CTL* state-logics. Although not the entire complexity of CTL* is used. For constitutive norms, we use production rules, with semantics inspired in the counts-as operator [30, 19].

Social Devices

The use of social devices is left to the designer of the organization. Methodological guidelines are given depending on the chosen organizational structure (market, hierarchy, network).

Dynamics of the system

Changes to the model over time have not been fully covered so far. We have the ability to describe the changes between two occurrences of an organization, but how to move the agent system implementing the organization to incorporate these changes, has not been extensively covered (yet).

Types of agents

OperA allows for the distinction between internal and external agents. Internal agents are assumed to be designed by the designer of the organization and used for enforcement tasks.

No further assumption is made about the nature of the agents, and the roles of an OperA organization can be enacted by agents, organizations, or humans, without requiring a change to the OperA model.

9.2.4 Operations

The OperA framework supports the specification (modelling) of organizations, and provides implementation guidelines (through the ALIVE framework) for implementation. Since the implementation needs additional sources of information (for contextualisation, concretisation and operationalisation of the organization), automatic implementation is not possible (human intervention is required).

9.2.5 Languages

OperA uses a number of languages, but the core of the framework is based on a (simplified) temporal predicate logic. Properties of various model fragments, such as objectives, landmarks, rights, etc, are expressed as Partial State Descriptions, which are, in essence, predicate logic sentences. The ontology component, in the Communicative Structure, is expressed in a (simplified) version of OWL. Norms in OperA are specified using a deontic modal logic (based on CTL*) that is temporal, relativized (in terms of roles and groups) and conditional. For instance, the following norm might hold: Supplier must submit their bids before the deadline, which can be formalized as $O_{supplier}(submit(bid) \leq Deadline)$.

In order to check norms and act on possible violations of the norms by the agents within an organization, abstract norms are translated into actions and concepts that can be handled within such organizations. To do so, the definition of abstract norms are iteratively concretized into more concrete norms, and then translated into specific rules, violations and sanctions.

Concrete norms are related to abstract norms through a mapping function, based on the counts-as operator as developed in [1]. For example, in the context of *Org*, $submit(bid)$ can be concretized as:

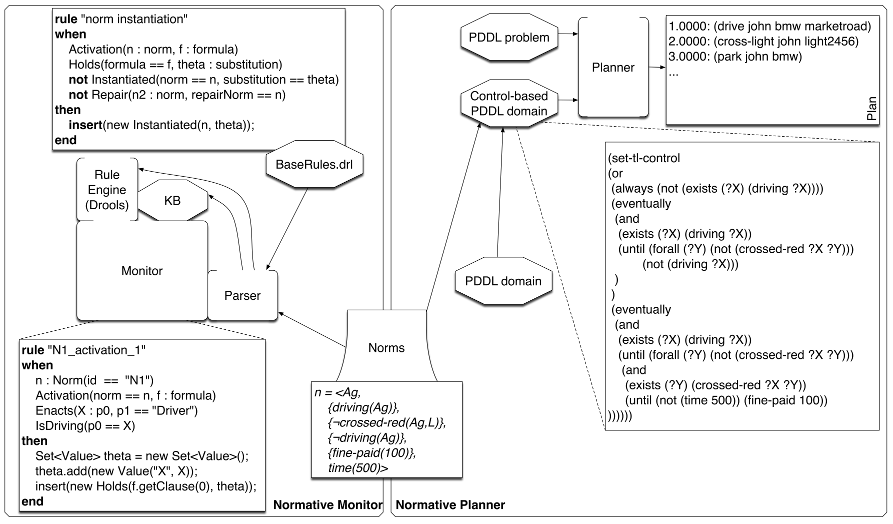
$$send(supplier, docs) \vee send_post(supplier, hard_copies) \rightarrow_{Org} submit(bid)$$

9.3 Tools and Platform

OperA / ALIVE is supported by a range of tools:

OperettA: a specification and validation tool created for the modelling and verification of OperA Organization Models (OM). OperettA [4] has a fully graphical interface, including social structure, interaction structure and partial-state editors. Moreover, OperettA collects all concepts (e.g., role names, objectives, etc.) to automatically build the Organizational Ontology while the designer is inputting the Organizational Model. OperettA stores Organizational Model in a standardised XML-format, which can be used by other components of the OperA-family. Finally, OperettA can perform automatic syntactic (and limited semantic) verification of the designed model, and can generate reorganization scripts to describe the changes made to an organizational model.

Norm Monitor: because of the strict separation between organization and individuals (roles vs. agents), runtime compliance to the organizational model needs to be monitored. In [8] we designed and implemented a norm monitoring framework that takes the norms from an OM (in XML) as input and checks whether the agents in the system comply with the given specification. A prototype of the norm monitor has been implemented in Java and Clojure for the parsing of the norms, and Drools for the rule engine [8].


```
rule "norm instantiation"
when
   Activation(n : norm, f : formula)
   Holds(formula == f, theta : substitution)
   not Instantiated(norm == n, substitution == theta)
   not Repair(n2 : norm, repairNorm == n)
then
   insert(new Instantiated(n, theta));
end
```

```
rule "N1_activation_1"
when
   n : Norm(id  ==  "N1")
   Activation(norm == n, f : formula)
   Enacts(X : p0, p1 == "Driver")
   IsDriving(p0 == X)
then
   Set<Value> theta = new Set<Value>();
   theta.add(new Value("X", X));
   insert(new Holds(f.getClause(0), theta));
end
```

```
n = <Ag,
   {driving(Ag)},
   {¬crossed-red(Ag,L)},
   {¬driving(Ag)},
   {fine-paid(100)},
   time(500)>
```

```
1.0000: (drive john bmw marketroad)
2.0000: (cross-light john light2456)
3.0000: (park john bmw)
...
```

```
(set-tl-control
(or
 (always (not (exists (?X) (driving ?X))))
 (eventually
  (and
   (exists (?X) (driving ?X))
   (until (forall (?Y) (not (crossed-red ?X ?Y)))
          (not (driving ?X)))
  )
 )
 (eventually
  (and
   (exists (?X) (driving ?X))
   (until (forall (?Y) (not (crossed-red ?X ?Y)))
    (and
     (exists (?Y) (crossed-red ?X ?Y))
     (until (not (time 500)) (fine-paid 100))
))))))
```

Fig. 9.3 Normative Monitor and Normative Planner

Normative Planner: another form of operationalisation of OperA models is done through the implementation of a Normative Planner [33] to assist agents enacting roles in an OperA organization. Implemented as a combination of TLPlan [9] and PDDL [16], it provides a complete, robust and rather fast implementation. In order to implement the normative planner, the norms from an OperA specification are simplified into a control statement that is fed into the planner. During execution, the planner will only allow paths where a norm never gets instantiated, or where a norm gets instantiated and never violated, or where a norm violated but repaired before the specified timeout is reached. That is, it discards the paths that do not conform to the norm life cycle. The system allows for multiple instantiations of a norm to be checked simultaneously.

ALIVEclipse: the integrated development framework ALIVEclipse allows designers to design service-oriented applications by using an OperA model to specify the high-level, abstract purposes of the system. Having the system organization explicit allows the ALIVE framework [6] to dynamically and flexibly change services and service invocations at runtime to achieve similar results. The OperA organizational model gives the context in which the system is to run, and what goals/objectives are important (and who is responsible for achieving it). The abstract organizational model is used to specify the task, actions and plans of the intelligent middle-layer (containing AgentScape agents), which on their turn choose the appropriate services and make the necessary service invocations. The framework can also be used in the opposite direction to derive organizational elements (roles, objectives, dependencies) from a running system to give means/purpose to an existing system (and achieve the same level of dynamic flexibility).

Norm Compliance and Conformance Checking Platform: a toolkit [26], based on Colored Petri-Nets [25], that can take the organizational norms (from the Normative Structure of the Organizational Model) and compute whether: a) the norms are consistent with each other (i.e., there are ways to comply with all the norms), and b) whether existing workflow executions (e.g., from a log or from a business process specification) is in conformance with those norms (i.e., testing norm compliance of a protocol). Finally, in [29] we extended the CCCP toolkit with the means to verify both normative compliance with individual preference to enable us to reason about optimality of plans/protocols from not only a normative perspective, but also from an individual agents perspective.

Organization Reasoning for Agents: a toolkit [22, 23], In open environments, agents should be able to reason about the benefits and duties incurred by entering an organisation, so that they can act within the expected boundaries and work towards the objectives of the organization. The AORTA component can be integrated into agents reasoning mechanism, allowing them to reason about (and act upon) regulations specified by an organizational model using simple reasoning rules. The added value is that the organizational model is independent of that of the agents, and that the approach is not tied to a specific organizational model. Organizational reasoning in AORTA is divided into two main parts: organizational option generation and organizational action deliberation. An organizational option is something that the agent should consider, such as an organizational objective or a role. An organizational action is the execution of an organizational option: enacting a role or committing to an organizational objective. AORTA adds organisational facts to the beliefbase of the agent, which can then be used in the deliberation process. Commitment to organisation objectives leads to change in the agents intentions and are incorporated in its plans. The approach is agent-centered, and independent from the organization, allowing agents to join open systems which are regulated by arbitrary organizations.

Other components:

- OperA / ALIVE assumes a strict separation between the organization and the individuals. It makes no assumption about the individuals regarding their nature or, in that sense, the way they are implemented. Agents in OperA can be programmed in any agent-oriented programming languages. OperA organizations have been implemented in AgentScape, JASON, JADE, 3APL, and even lower-level languages such as RePaST.
  To program agents suitable for an OperA organization, a minimal of changes is required:
  - Minimal understanding of the OperA organizational language/structure (i.e., being able to read and understand the OperA XML-specification) [22].
  - Capability to reason about the agents own capabilities to correctly apply for a position in the organization [37].
  - One (or more) agent(s) employed to act as gatekeeper agent; solving role enactment (i.e., accepting agents in organizational positions, based, e.g., on their performance and capabilities) [5].
  - One (or more) agent(s) employed as monitor (see Norm Monitor above) or as enforcer [40].

9.4 OperA in use

In this section we present a brief description of how OperA approaches the tender use-case (see chapter I.2).

9.4.1 Modelling and Implementation

As mentioned, the OperA framework consists of three interrelated models: the organizational model, the social model, and the interaction model. The **Organization Model** (OM) is the result of the observation and analysis of the domain and describes the desired behavior of the organization, as determined by the organizational stakeholders in terms of objectives, norms, roles, interactions and ontologies.

The OM provides the overall organization design that fulfills the stakeholders requirements. Objectives of an organization are achieved through the action of agents, which means that, at each moment, an organization should employ the relevant agents that can make its objectives happen. However, the OM does not enable to specify the individual agents. The **Social Model** (SM) maps organizational roles to (existing) agents and describes agreements concerning the role enactment and other conditions in enactment contracts. Finally, the **Interaction Model** (IM) describes the runtime interactions between role-enacting agents. The overall development process is depicted in figure 9.1.

In this section, we show how to develop an Organization Model (OM) to specify the structure and global characteristics of a case from an organizational perspective. The OM describes the means to achieve global objectives. Components of OM are the *Social* and *Interaction Structures* where global goals are specified in terms of roles and interactions. Moreover, organization specifications should include the description of concepts holding in the domain, and of expected or required behaviors. Therefore, these structures should be linked with the norms, defined in *Normative Structure*, and with the ontologies and communication languages defined in the *Communication Structure*.

The Social Structure

The social structure describes the roles and dependencies holding in the organization. It consists of a list of role definitions, *Roles* (including their objectives, rights and requirements), a list of role groups' definitions, *Groups*, and a *Role Dependency*'s graph. Examples of roles in the Tender Process scenario are Contractor, Bidder, Publication Body, etc.

Global objectives are the basis for the definition of the objectives of roles. From the organization perspective, role descriptions should identify the activities and services necessary to achieve its objectives and also enable to abstract from the individuals that will eventually perform the role. From the agent perspective, roles specify the expectations of the society with respect to the agent's activity in the society. In OperA, the definition of a role consists of an identifier, a set of role objectives, possibly sets of sub-objectives per objective, a set of role rights, a set of norms and the type of role. An example of a role description for Evaluator in the Tender Process scenario is depicted in table 9.1.

Groups provide means to collectively refer to a set of roles and are used to specify norms that hold for all roles in the group. Groups are defined by means of an identifier, a non-empty set of roles, and group norms. An example of a group in the Tender Process scenario is the bid consortium team consisting of the roles *Bidder* and *Partner*.

The distribution of objectives in roles is defined by means of the *Role Hierarchy*. Different criteria can guide the definition of *Role Hierarchy*. In particular, a role can

Id	Evaluator
Objectives	bid_evaluated(Bid,Rep)
Sub-objectives	{read(Bid), report_written(Bid, Rep), review_received(Contractor, Bid, Rep)}
Rights	access_bidregsystem_system(*me*)
Norms & Rules	Evaluator OBLIGED understand_english Evaluator OBLIGED bid_evaluated BEFORE deadline IF conflict_interest THEN Evaluator FORBIDDEN bid_evaluated

Table 9.1 *Evaluator* role description.

be refined by decomposing it in sub-roles that, together, fulfill the objectives of the given role.

This refinement of roles defines *Role Dependencies*. A dependency graph represents dependency relations between roles. Nodes in the graph are roles in the society. Arcs are labeled with the objectives for which the parent role depends on the child role. Part of the dependency graph for the Tender Process scenario is displayed in figure 9.4.

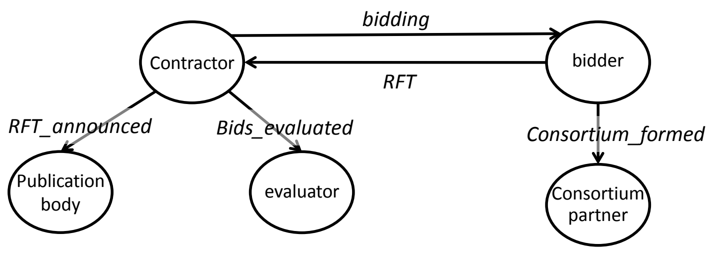


Fig. 9.4 Role dependencies in the Tender Process scenario.

For example, the arc between nodes Contractor and Evaluator represents the dependency concerning *bid-evaluated* ($Contractor \succeq_{bid\_evaluated} Evaluator$). The way objective g of role $r1$ in a dependency relation $r_1 \succeq_g r_2$ is actually passed to $r2$ depends on the coordination type of the system, defined in the Architectural Templates. In OperA, three types of role dependencies are identified: *bidding*, *request* and *delegation*. These dependency types, result in three different interaction possibilities:

Bidding defines market, or auction-like interactions, where the dependent (initiator) of the dependency asks for proposals from the dependees. Typically, the best proposal is selected for the achievement of the objective.

Request leads to networks, where roles interact cooperatively towards the achievement of an objective;

Delegation gives raise to hierarchies, where the dependent of the dependency delegates the responsibility of the achievement of the objective to the dependees (i.e., subordinates).

The Interaction Structure

Interaction is structured as a set of meaningful scenes that follow pre-defined scene scripts. Examples of scenes are the 'bid submission', which involves Bidder and Contractor, or 'Bid Evaluation Process', involving Contractor and the Evaluators. A *scene script* describes the players (roles), desired results and the norms regulating the interaction. The results of an interaction scene are achieved by the joint activity of the participating roles, through the realization of (sub-)objectives of those roles. A scene script establishes also the desired *interaction patterns* between roles, that is, a desired combination of the (sub-) objectives of the roles. Table 9.2 gives an example of a scene script for the review process involving two Evaluators and the Contractor.

Scene	Bid Evaluation Process
Roles	Contractor (1), Evaluator(2)
Results	r_1 = ∀ Bid ∈ Bids: evaluation_done(Bid, eval1, eval2)
Interaction Pattern	PATTERN(r_1): *see figure 9.5*
Norms & Rules	Contractor PERMITTEDbid_assigned IF bid_assigned THEN Evaluator OBLIGED bid_evaluated BEFORE deadline

Table 9.2 Script for the *Bid Evaluation Process* scene.

Interaction scene descriptions are declarative, indicating the global aims of the interaction rather than describing exact activities in details. Interaction patterns can be more or less restrictive, which will give the agent enacting the role more or less freedom to decide how to achieve the role objectives and interpret its norms. Following the ideas of [38, 31], we call such expressions *landmarks*, defined as conjunctions of logical expressions that are true in a state. Landmarks combined with a partial ordering to indicate the order in which the landmarks are to be achieved are called a *landmark pattern*. Figure 9.5 shows the landmark pattern for the *Bid Evaluation Process*. Several different specific actions can bring about the same state, that

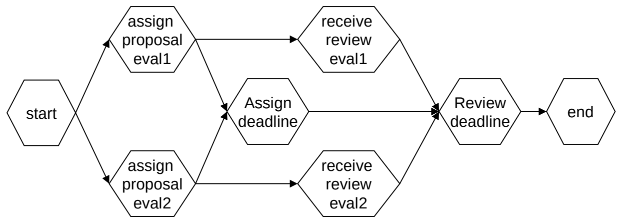


Fig. 9.5 Landmark pattern for *Bid Evaluation Process*.

is, landmark patterns actually represent families of actual interaction protocols. The use of landmarks to describe activity enables the actors to choose the best applicable actions, according to their own goals and capabilities. The ordering relation

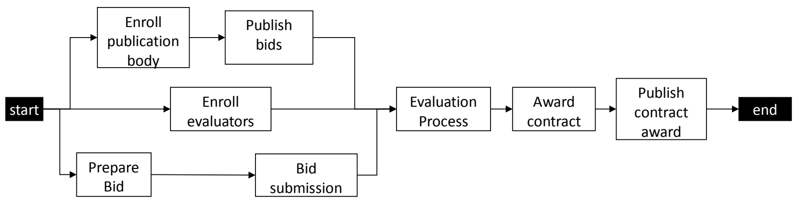


Fig. 9.6 Interaction Structure in the Tender Process scenario.

between scenes is given in the *Interaction Structure* (see figure 9.6). In this diagram, *transitions* describe a partial ordering of the scenes, plus eventual synchronization constraints. Note that, at runtime, several scenes can be happening at the same time and one agent can participate in different scenes simultaneously. Transitions also describe the conditions for the creation of a new instance of the scene, and specify the maximum number of scene instances that are allowed simultaneously. Furthermore, the enactment of a role in a scene may have consequences in following scenes. Role *evolution relations* describe the constraints that hold for the role-enacting agents as they move from scene to scene, e.g., in the transition between paper acceptance and conference registration authors will became participants.

The Normative Structure.

At the highest level of abstraction, norms are the *values* of a society, in the sense that they define the concepts that are used to determine the value or utility of situations. For the tender scenario, the desire to write out appropriate calls for tender and do a fair allocation can be seen as values. However, the values do not specify *how*, *when* or in *which* conditions individuals should behave appropriately in any given social setup. In OperA, these aspects are defined in the Normative Structure.

In OperA, norms are specified using a deontic logic that is temporal, relativized (in terms of roles and groups) and conditional [14]. For instance, the norm *"The bidders should submit their proposals before the submission deadline"* is formalized as, e.g.: $O_{bidder}(submit(proposal) \leq Submission\_deadline)$

Furthermore, in order to check norms and act on possible violations of the norms by the agents within an organization, abstract norms have to be translated into actions and concepts that can be handled within such organizations. To do so, the definition of the abstract norms are iteratively concretized into more concrete norms, and then translated into specific rules, violations and sanctions. Concrete norms are related to abstract norms through a mapping function, based on the "counts-as" operator as developed in [1].

The Communication Structure.

Communication mechanisms include both the representation of domain knowledge (*what* are we talking about) and protocols for communication (*how* are we talking). Both content and protocol have different meanings at the different levels of abstraction. E.g. while at the abstract level one might talk of *disseminate knowledge*, such action will most probably not be available to agents acting at the implementation level, where such abstract objective will be translated into concrete actions, such as *publish proceedings*. Specification of communication content is usually realized using ontologies, which are shared conceptualizations of the terms and predicates in a domain. Agent communication languages (ACLs) are the usual means in MAS to describe communicative actions. ACLs are wrapper languages in the sense that they abstract from the content of communication.

In OperA, the Communication Structure describes both the content and the language for communication. The content aspects of communication, or domain knowledge, are specified by *Domain Ontologies* and *Communication Acts* define the language for communication, including the performatives and the protocols.

9.4.2 Discussion

The main structures in the tender scenario fit well with OperA concepts. Given that the current tool support for OperA focus on the design of organisational structure, we have limited the application of OperA to the case study to these aspects. As such, the work described in the previous section concerns the specification of the Organisational Model for the RFT scenario. We have particularly focussed on the specification of roles, dependencies and interaction scenes as these are the main and most distinctive concepts of OperA OM. The use of landmarks enables to specify a sufficient rich organisation such that all organisational requirements and goals are taken into account and enable the verification of global properties, but the resulting model is still sufficiently open for adaptation to an individual agents needs. Using AORTA [22, 23], it is possible to provide an agent-readable specification of the organisational model to a BDI agent. We did not illustrated this aspect here, because that would require the introduction of specific agent platforms, which is outside the scope of this volume and would unduly extend the length of the chapter.

9.5 Critical Assessment

A key feature of the OperA approach is the strict separation between the agents and the organization. The difference of interests of the agents individually and the organization as a whole are a cornerstone of the framework.

The strongest point is the modelling framework (with accompanying tools support) for modelling the organizational model. The fact that the organizational model abstracts from the actual agents implemeting the organization, means that it is much easier for non-computer scientists to model in OperA (no implementation details are required to make the organisational model).

This abstraction, however, comes with a cost. The transition from organizational model to implementation is more difficult to make, due to the fact that there is a gap in knowledge and/or ontology between the organization and the implementation. OperA speaks only about organizational aspects (in an almost declarative manner), which make operationalisation difficult.

While we have tried Model-driven architectures to assist in the automatic generation of implementations, these still require a designer present to enhance/complete the aspects of the model that are missing on the organizational abstraction level. More recent work on organization-aware agents circumvents some of the problems, as the link between the implementation and specification is done directly within the agent program.

9.6 Key references

For a more detailed reading about the OperA framework and its use, we recommend the following:

[11] Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. SIKS Dissertation Series 2004-1. Utrecht University (2004)

PhD-thesis covering the main concepts of OperA, the initial assumptions for the model and motivations for choices made.

[4] Aldewereld, H., Dignum, V.: OperettA: Organization-oriented development environment. In: Proceedings of the 3rd International workshop on Languages, Methodologies and Development Tools for Multi-agent Systems (LADS2010@Mallow) (2011)

Introduction and explanation of the OperettA-tool, as support tool for building and maintaining OperA OM models.

[23] Jensen, A.S., Dignum, V., Villadsen, J.: A framework for organisation-aware agents. JAAMAS (2015). Submitted.

Introduces and elaborates on the use of organization-aware agents.

[6] Aldewereld, H., Padget, J., Vasconcelos, W., V azquez-Salceda, J., Sergeant, P., Staikopoulos, A.: Adaptable, organization-aware, service-oriented computing. Intelligent Systems, IEEE 25(4), 2635 (2010)

Shows the role and place of organizational modelling in service-oriented architecture-based software development.

References

1. Aldewereld, H., Álvarez-Napagao, S., Dignum, F., Vázquez-Salceda, J.: Making norms concrete. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 807–814. International Foundation for Autonomous Agents and Multiagent Systems (2010)
2. Aldewereld, H., Dignum, F., Hiel, M.: Re-organization in warehouse management systems. In: Proceedings of the IJCAI 2011 workshop on artificial intelligence and logistics (AILog-2011), pp. 67–72 (2011)
3. Aldewereld, H., Dignum, F., Hiel, M.: Decentralised warehouse control through agent organisations. In: Automation in Warehouse Development, pp. 33–44. Springer (2012)
4. Aldewereld, H., Dignum, V.: OperettA: Organization-oriented development environment. In: Proceedings of the 3rd International workshop on Languages, Methodologies and Development Tools for Multi-agent Systems (LADS2010@Mallow) (2011)
5. Aldewereld, H., Dignum, V., Jonker, C.M., van Riemsdijk, M.B.: Agreeing on role adoption in open organisations. KI-Künstliche Intelligenz **26**(1), 37–45 (2012)
6. Aldewereld, H., Padget, J., Vasconcelos, W., Vázquez-Salceda, J., Sergeant, P., Staikopoulos, A.: Adaptable, organization-aware, service-oriented computing. Intelligent Systems, IEEE **25**(4), 26–35 (2010)
7. Aldewereld, H., Tranier, J., Dignum, F., Dignum, V.: Agent-based crisis management. In: Collaborative Agents-Research and Development, pp. 31–43. Springer (2011)
8. Alvarez-Napagao, S., Aldewereld, H., Vázquez-Salceda, J., Dignum, F.: Normative Monitoring: Semantics and Implementation. Coordination, Organizations, Institutions, and Norms in Agent Systems VI **6541**, 321–336 (2011)
9. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artificial Intelligence **116**(1-2), 123–191 (2000)
10. Dastani, M., Dignum, V., Dignum, F.: Role-assignment in open agent societies. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 489–496. ACM (2003)
11. Dignum, V.: A Model for Organizational Interaction: based on Agents, founded in Logic. SIKS Dissertation Series 2004-1. Utrecht University (2004)
12. Dignum, V., Dignum, F.: Modelling agent societies: Co-ordination frameworks and institutions. In: Progress in artificial intelligence, pp. 191–204. Springer (2001)
13. Dignum, V., Dignum, F.: A logic of agent organizations. Logic Journal of IGPL **20**(1), 283–316 (2012)
14. Dignum, V., Meyer, J.J.C., Dignum, F., Weigand, H.: Formal specification of interaction in agent societies. In: Formal approaches to agent-based systems, pp. 37–52. Springer (2003)
15. Esteva, M., Padget, J., Sierra, C.: Formalizing a language for institutions and norms. In: Intelligent agents VIII, pp. 348–366. Springer (2002)
16. Gerevini, A., Long, D.: Plan constraints and preferences in PDDL3: The Language of the Fifth International Planning Competition. Tech. Rep. R.T. 2005-08-07 (2005)
17. Grossi, D., Aldewereld, H., Dignum, F.: Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In: Coordination, organizations, institutions, and norms in agent systems II, pp. 101–114. Springer (2007)
18. Grossi, D., Dignum, F., Dastani, M., Royakkers, L.: Foundations of organizational structures in multiagent systems. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 690–697. ACM (2005)
19. Grossi, D., Meyer, J.J.C., Dignum, F.: Classificatory aspects of counts-as: An analysis in modal logic. Journal of Logic and Computation **16**(5), 613–643 (2006)

20. Group, O.M.: Meta Object Facility (MOF) Specification (2003)
21. Hiel, M., Aldewereld, H., Dignum, F.: Modeling warehouse logistics using agent organizations. In: Collaborative Agents-Research and Development, pp. 14–30. Springer (2011)
22. Jensen, A.S., Dignum, V., Villadsen, J.: The aorta architecture: Integrating organizational reasoning in jason. In: Engineering Multi-Agent Systems, pp. 127–145. Springer (2014)
23. Jensen, A.S., Dignum, V., Villadsen, J.: A framework for organisation-aware agents. JAAMAS (2015). Submitted
24. Jensen, A.S., Spurkeland, J.S., Villadsen, J.: Formalizing theatrical performances using multi-agent organizations. In: SCAI, pp. 135–144 (2013)
25. Jensen, K.: Coloured petri nets. In: Petri nets: central models and their properties, pp. 248–299. Springer (1987)
26. Jiang, J., Aldewereld, H., Dignum, V., Tan, Y.H.: Compliance checking of organizational interactions. ACM Transactions on Management Information Systems (TMIS) **5**(4), 23 (2014)
27. Jiang, J., Dignum, V., Aldewereld, H., Dignum, F., Tan, Y.H.: Norm compliance checking. In: Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, pp. 1121–1122. International Foundation for Autonomous Agents and Multiagent Systems (2013)
28. Jiang, J., Dignum, V., Tan, Y.H.: An agent-based inter-organizational collaboration framework: Opera+. In: Coordination, Organizations, Institutions, and Norms in Agent System VII, pp. 58–74. Springer (2012)
29. Jiang, J., Thangarajah, J., Aldewereld, H., Dignum, V.: Reasoning with agent preferences in normative multi-agent systems. In: Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, pp. 1373–1374. International Foundation for Autonomous Agents and Multiagent Systems (2014)
30. Jones, A.J., Sergot, M.: A formal characterisation of institutionalised power. Logic Journal of IGPL **4**(3), 427–443 (1996)
31. Kumar, S., Huber, M.J., Cohen, P.R., McGee, D.R.: Toward a formalism for conversation protocols using joint intention theory. Computational Intelligence **18**(2), 174–228 (2002)
32. Mensonides, M., Huisman, B., Dignum, V.: Towards agent-based scenario development for strategic decision support. In: Agent-oriented information systems IV, pp. 53–72. Springer (2008)
33. Panagiotidi, S., Vázquez-Salceda, J., Dignum, F.: Reasoning over norm compliance via planning. In: Coordination, Organizations, Institutions, and Norms in Agent Systems VIII, pp. 35–52. Springer (2013)
34. Parunak, H.V.D., Odell, J.J.: Representing social structures in uml. In: Agent-Oriented Software Engineering II, pp. 1–16. Springer (2002)
35. Penserini, L., Dignum, F., Dignum, V., Aldewereld, H., Grossi, D.: Evaluating organizational configurations. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02, pp. 153–160. IEEE Computer Society (2009)
36. Quillinan, T.B., Brazier, F., Aldewereld, H., Dignum, F., Dignum, V., Penserini, L., Wijngaards, N.: Developing agent-based organizational models for crisis management. In: Proc. of the 8th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2009), pp. 45–51 (2009)
37. van Riemsdijk, M.B., Dignum, V., Jonker, C.M., Aldewereld, H.: Reflection about capabilities for role enactment. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3, pp. 1231–1232. International Foundation for Autonomous Agents and Multiagent Systems (2011)
38. Smith, I.A., Cohen, P.R., Bradshaw, J.M., Greaves, M., Holmback, H.: Designing conversation policies using joint intention theory. In: Multi Agent Systems, 1998. Proceedings. International Conference on, pp. 269–276. IEEE (1998)
39. Vázquez-Salceda, J.: The role of norms and electronic institutions in multi-agent systems applied to complex domains. the harmonia framework. Ai Communications **16**(3), 209–212 (2003)

40. Vázquez-Salceda, J., Aldewereld, H., Grossi, D., Dignum, F.: From human regulations to regulated software agents behavior. Artificial Intelligence and Law **16**(1), 73–87 (2008)
41. Vázquez-Salceda, J., Dignum, V., Dignum, F.: Organizing multiagent systems. Autonomous Agents and Multi-Agent Systems **11**(3), 307–360 (2005)
42. Weigand, H., Dignum, V.: I am autonomous, you are autonomous. In: Agents and Computational Autonomy, pp. 227–236. Springer (2004)
43. Westra, J., Dignum, F., Dignum, V.: Modeling agent adaptation in games. In: BNAIC 2008 Belgian-Dutch Conference on Artificial Intelligence, p. 381 (2008)
44. Westra, J., Dignum, F., Dignum, V.: Keeping the trainee on track. In: Computational Intelligence and Games (CIG), 2010 IEEE Symposium on, pp. 450–457. IEEE (2010)
45. Westra, J., van Hasselt, H., Dignum, F., Dignum, V.: Adaptive serious games using agent organizations. In: Agents for Games and Simulations, pp. 206–220. Springer (2009)
46. Zambonelli, F.: Abstractions and infrastructures for the design and development of mobile agent organizations. In: Agent-Oriented Software Engineering II, pp. 245–262. Springer (2002)

Appendix A

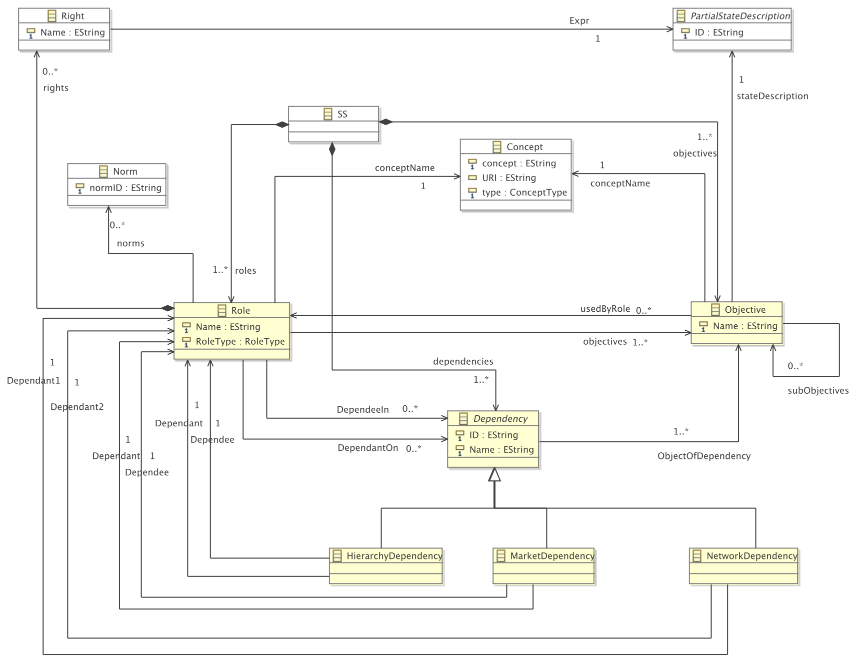


Fig. 9.7 OperA metamodel: Social Structure (SS).

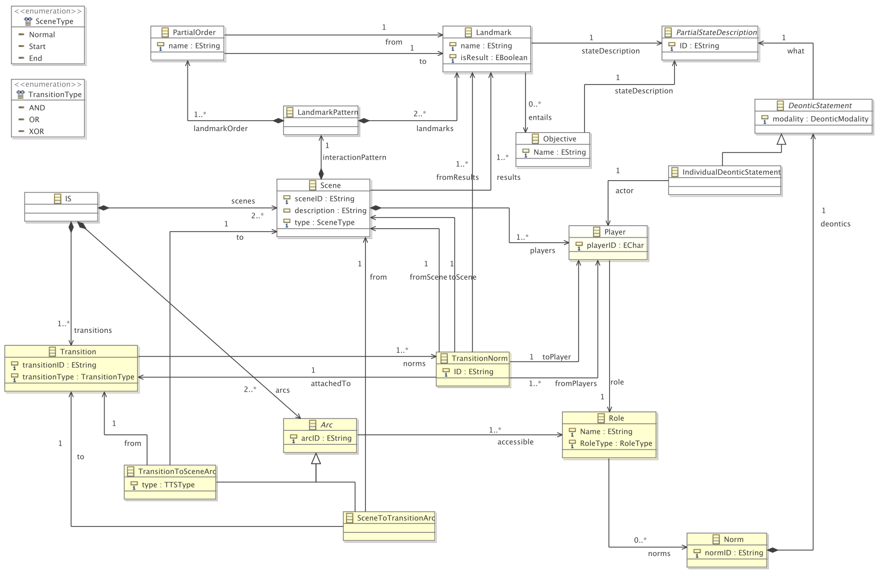


Fig. 9.8 OperA metamodel: Interaction Structure (IS).

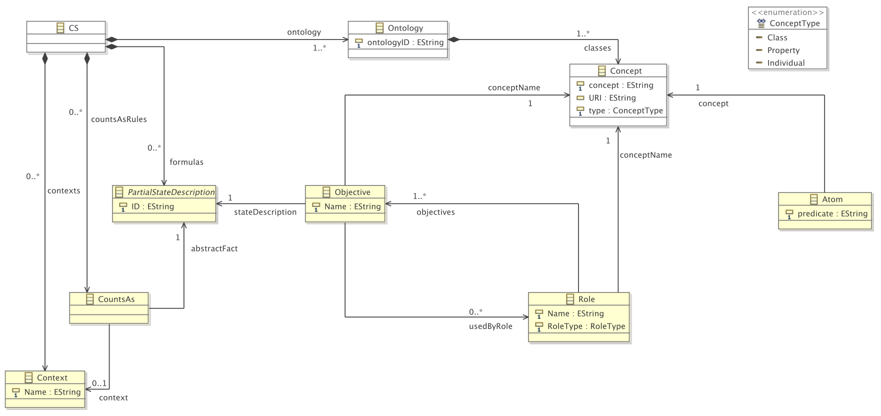


Fig. 9.9 OperA metamodel: Communicative Structure (CS).

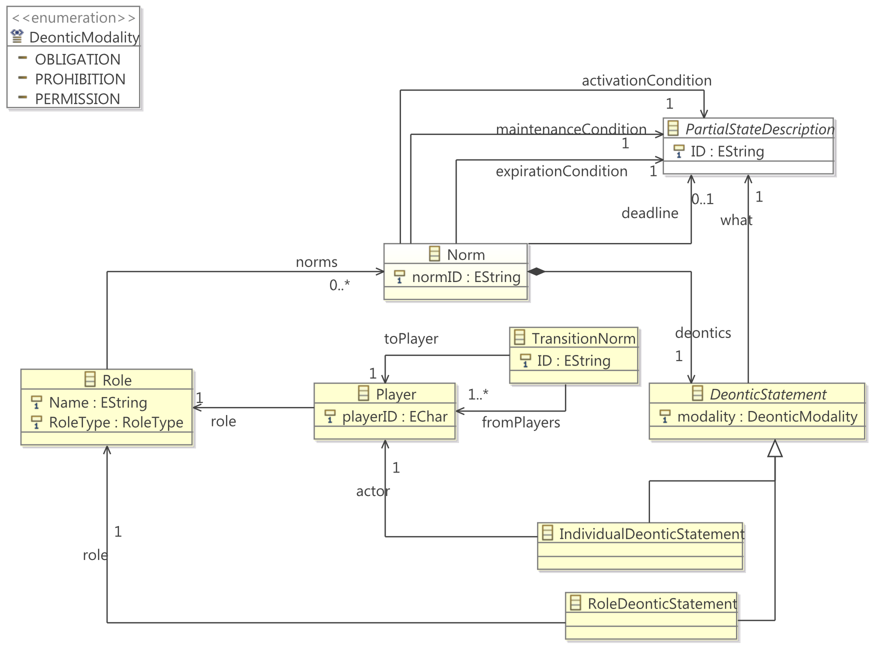


Fig. 9.10 OperA metamodel: Normative Structure (NS).