# OperettA: Organization-Oriented Development Environment

Virginia Dignum
Delft University of Technology, The Netherlands, m.v.dignum@tudelft.nl

Huib Aldewereld
Utrecht University, The Netherlands, huib@cs.uu.nl

*Abstract*—**The increasing complexity of distributed applications requires new modeling and engineering approaches. Such domains require representing the regulating structures explicitly and independently from the acting components (or agents). Organization computational models, based on Organization Theory, have been advocated to specify such systems. In this paper, we present the organizational modeling approach OperA and a graphical environment for the specification and analysis of organizational models, OperettA. OperA provides an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it.**

## I. Introduction

The engineering of applications for complex and dynamic domains is an increasingly difficult process. Requirements and functionalities are not fixed a priori, components are not designed nor controlled by a common entity, and unplanned and unspecified changes may occur during runtime. There is a need for representing the regulating structures explicitly and independently from the acting components (or agents). Organization computational models, based on Organization Theory, have been advocated to specify such systems.

Traditionally, Multi-Agent Systems (MAS) stress the autonomy and encapsulation characteristics of agents. In such agent-centric view, interactions between agents are mostly seen as speech acts whose meaning may be described in terms of the mental states of an agent. As systems grow to include hundreds or thousands of agents, it is necessary to move from an agent-centric view of coordination and control to an organization-centric one. Organizations provide stable means for coordination that enable the achievement of global goals. In this sense, organization models play a critical role in the development of larger and more complex MAS [5].

Comprehensive analysis of several agent systems has shown that different design approaches are appropriate for different domain characteristics [6]. In particular, agent organization frameworks are suitable to model complex environments where many independent entities coexist witing explicit normative and organizational structures and global specification of control measures is necessary.

Models for agent organizations must, on the one hand, be able to specify global goals and requirements but, on the other hand, cannot assume that participating actors will always act according to the needs and expectations of the system design. Concepts as organizational rules [21], norms and institutions [9], [10], and social structures [15] 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 [22]. These are the rules and global objectives that govern the activity of an organization.

The OperA model [4] proposes an expressive way for defining agent organizations distinguishing explicitly between the organizational aims, and the agents who act in it. 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. OperA has been applied is many domains, including knowledge management, work practice analysis, and serious games and social simulation. Space constraints do not allow for a comparison of OperA with other approaches. For this effect, we refer the reader to [2].

In this paper, we present OperettA, a graphical environment for the specification, validation and analysis of organizational models, based on the OperA formalism [4]. This organization specification tool builds heavily on mechanisms from Model Driven Engineering (MDE), which enables the introduction and combination of different formal methods hence enabling the modeling activity through systematic advices and model design consistency checking.

The paper is organized as follows: first, in section II we introduce and briefly explain the OperA framework. In section III the specification of organizational models in OperA is detailed. In section IV we introduce the OperettA Environment. In section V we provide some design guidelines for organization models. Finally, section VI gives our conclusions.

## II. Organization Modeling: the OperA Framework

Organizational models should enable the explicit representation of structural and strategic concerns and their adaptation to environment changes in a way that is independent from the behavior of the agents. Organization models, combining global requirements and individual initiative, have been advocated to specify open systems in dynamic environments [11], [4]. We take formal processes and requirements as a basis for the modeling of complex systems that regulate the action of the different agents.

The deployment of organizations in dynamic and unpredictable settings brings forth critical issues concerning the design, implementation and validation of their behavior [16], [13], [20], 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 [4] 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. At an abstract level, an OperA model describes the aims and concerns of the organization with respect to the social system. These are described as organization's externally observable *objectives*, that is, the desired states of affairs for the organization.

The OperA framework consists of three interrelated models. The **Organizational 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 specify how to structure groups of agents and constrain their behavior 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 the 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 enable variations to the enactment of interactions between role-enacting agents.

The OperettA framework is developed to specify organization models, according to the OperA OM, which will be described in more detail in section III, using as example the conference organization scenario taken from [8]. In section IV-B we describe the use of MDE principles to implement this framework.

## III. The Organization Model

A common way to express the objectives of an organization is in terms of its expected functionality, that is, what is the organization expected to do or produce.In OperA, the Organization Model (OM) specifies the *means* to achieve such objectives. That is, OM describes the structure and global characteristics of a domain from an organizational perspective, where global goals determine roles and interactions, specified in terms of *Social* and *Interaction Structures*. E.g., how should a conference be organized, its program, submissions, etc.

Moreover, organization specification 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*.

### *A. The Social Structure.*

The social structure of an organization describes the roles holding in the organization. It consists of a list of role definitions, *Roles* (including their objectives, rights and requirements), such as PC-member, program chair, author, etc.; a list of role groups' definitions, *Groups*; and a *Role Dependencies* graph.

Abstract society objectives form the basis for the definition of the objectives of roles. *Roles* are the main element of the *Social Structure*. From the society perspective, role descriptions should identify the activities and services necessary to achieve society objectives and 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 role description is presented in table I.

| | |
|---|---|
| *Id* | PC_member |
| *Objectives* | paper_reviewed(Paper,Report) |
| *Sub-objectives* | {read(P), report_written(P, Rep), review_received(Org, P, Rep)} |
| *Rights* | access-confmanager-program($me$) |
| *Norms & Rules* | PC_member is OBLIGED to understand English<br>IF paper_assigned THEN PC_member is OBLIGED<br>to review paper BEFORE given deadline<br>IF author of paper_assigned is colleague<br>THEN PC_member is OBLIGED to refuse to review asap |

TABLE I
*PC member* ROLE DESCRIPTION.

*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 conference scenario is the organizing team consisting of the roles *program chair*, *local organizer*, and *general chair*.

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 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 the dependency relations between roles. Nodes in the graph are roles in the society.

Arcs are labelled with the objectives for which the parent role depends on the child role. Part of the dependency graph for the conference society is displayed in figure 1. For example, the arc between nodes PC-Chair and PC-member represents the dependency between *PC-Chair* and *PC-member* concerning *paper-reviewed* ($PC - Chair \succeq_{paper\_reviewed} PC - Member$).The way objective $g$ in a dependency relation $r_1 \succeq_g r_2$ is actually passed between $r1$ and $r2$ depends on the coordination type of the society, defined in the Architectural Templates. In OperA, three types of role dependencies are identified: *bidding*, *request* and *delegation*.

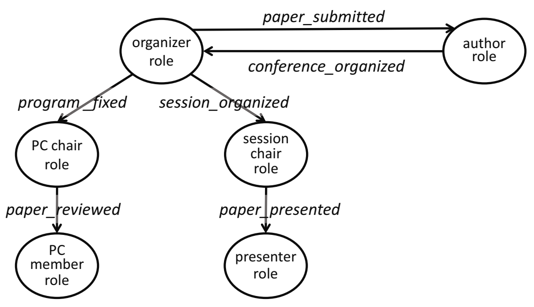


Fig. 1. Role dependencies in a conference.

## B. The Interaction Structure.

Interaction is structured as a set of meaningful scenes that follow pre-defined abstract scene scripts. Examples of scenes are the registration of participants in a conference, which involves a representative of the organization and a potential participant, or paper review, involving program committee members and the PC chair. A *scene script* describes a scene by its players (roles), its desired results and the norms regulating the interaction. In the OM, scene scripts are specified according to the requirements of the society. 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 II gives an example of a scene script.

| | |
|---|---|
| *Scene* | Review Process |
| *Roles* | Program-Chair (1), PC-member(2..Max) |
| *Results* | $r_1$ = ∀ P ∈ Papers: reviews_done(P, rev1, rev2) |
| *Interact. Pattern* | PATTERN($r_1$): *see figure 2* |
| *Norms & Rules* | Program-Chair is PERMITTED to assign papers<br>PC-member is OBLIGED to review papers assigned before deadline |

TABLE II
SCRIPT FOR THE *Review Process* SCENE.

OperA interaction descriptions are declarative, indicating the global aims of the interaction rather than describing exact activities in details. Interaction objectives can be more or less restrictive, giving 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 [17], [14], 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 2 shows the landmark pattern for the *Review Process*. Several different specific actions can bring about the same

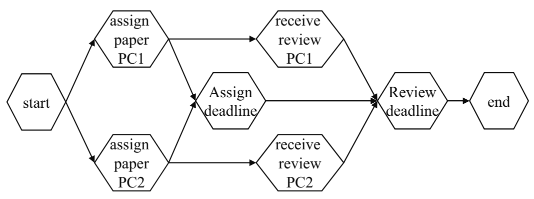


Fig. 2. Landmark pattern for *Review Process*.

state, that is, landmark patterns actually represent families of 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 relation between scenes is

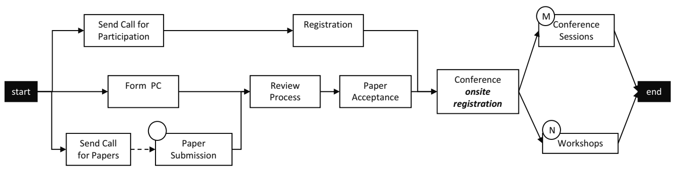


Fig. 3. Interaction Structure in the Conference scenario.

represented by the *Interaction Structure* (see figure 3). In this diagram, *transitions* describe a partial ordering of the scenes, plus eventual synchronization constraints. Note that 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.

## C. 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 conference organization scenario, the desire to share information and uphold scientific quality can be seen as organization values. However, values do not specify *how*, *when* or in *which* conditions individuals should behave appropriately in any given social setup.

In OperA, norms are specified in the Normative Structure using a deontic logic that is temporal, relativized (in terms of roles and groups) and conditional. For instance, the following norm might hold: *"The authors must submit their* *contributions before the deadline"*, which can be formalized as: $O_{author}(submit(paper) \leq Deadline)$

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]. For example, in the context of $Org$, $submit(paper)$ can be concretized as:$send\_mail(organizer, files) \vee send\_post(organizer, hard\_copies) \rightarrow_{Org} submit(paper)$

### D. 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*, such action will most probably not be available to agents acting at the implementation level). 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 performatives and protocols.

## IV. OperettA Environment

In order to support developers designing and maintaining organization models, tools are needed that provide an organization-oriented development environment. The requirements for such a development environment are the following.

1) Organizational Design: The tool should provide means for designing organizational models in an 'intuitive' manner. The tool should allow users to create and represent organizational structures, define the parties involved in an organization, represent organizational and role objectives, and define the pattern of interactions typically used to reach these objectives.
2) Organizational Verification: The tool should provide verification means and assistance in detecting faults in organizational designs as early as possible, to prevent context design issues from being translated to the other levels of system specification.
3) Ontology Design: The tool should to be able to specify, import, and maintain domain ontologies. Domain ontologies specifying the knowledge for a specific domain of interaction should be able to be represented, existing ontologies containing such information should be able to be included (and provide inputs for organizational concepts, such as role or objective names). Ontologies should be maintainable and updatable.
4) Connectivity to System Level: The output of the organizational design tool is intended for use by system level tools, namely MAS environments and agent programming languages. The output of the tool thus needs to provide easy integration and connection between the organization and system level.
5) User-Friendly GUI: A user-friendly graphical interface is to be provided for users to create and maintain organizational models easily. Help and guidelines are useful for beginners to use the tool.
6) Availability: The tool should be available under open source license and for use by other projects.

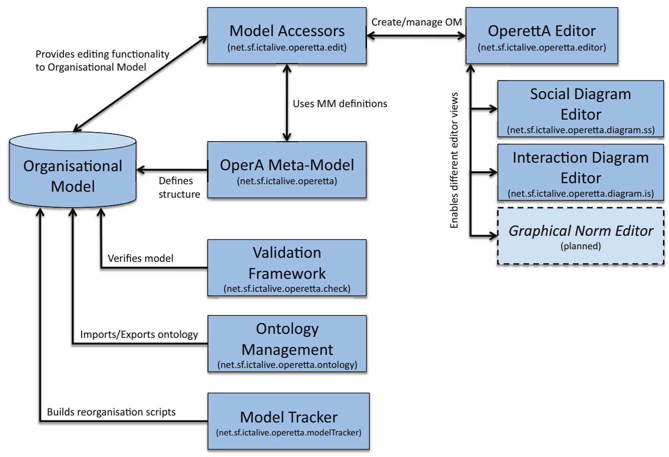
We have developed the OperettA development environment as an open-source solution on the basis of these requirements. OperettA enables the specification and verification of OperA OMs, which satisfies requirements 1 and 2. OperettA combines multiple editors into a single package. It provides separate editors on different components of organizational models; i.e., it has different (graphical) editors for each of the main components of an organizational model as defined in the OperA framework. These specialized editors correspond to the OperA OM structures: social, interaction, normative and communicative. The OperettA Ontology Manager enable the specification and import of domain ontologies, as in requirement 3.

The OperettA tool is a combination of tools based on the Eclipse Modeling Framework (EMF) [18] and tools based on the Graphical Modeling Framework (GMF) integrated into a single editor. Developed as an Eclipse plug-in, OperettA is fully open-source and follows the MDE principles of tool development. In the following we look at the editors provided by OperettA, and how OperettA connects to MAS solutions, thus satisfying requirement 4. A graphical interface (requirement 5) for OperettA has been developed and is currently being user-tested within the ALIVE project [12]. Finally, in accordance to the last requirement, OperettA is available opensource at sourceforge[1].

### A. OperettA Components

The main element of OperettA is the OperA Meta-Model (see figure 4 for an overview of the tools in OperettA and their functionalities). The meta-model, created with the EMF tools, provides the (structural) definition of what organizational models should look like. This meta-model is extended with the default EMF edit and editor plug-ins to provide model accessors and the basic tree-based editor for the creation and management of OperA models. The basic editor has been extended with graphical interfaces for editing parts of the organization model: the social diagram editor, and the

[1]http://ict-alive.svn.sourceforge.net/



Model Accessors
(net.sf.ictalive.operetta.edit)
Create/manage OM
OperettA Editor
(net.sf.ictalive.operetta.editor)
Provides editing functionality to Organisational Model
Uses MM definitions
Enables different editor views
Social Diagram Editor
(net.sf.ictalive.operetta.diagram.ss)
Organisational Model
Defines structure
OperA Meta-Model
(net.sf.ictalive.operetta)
Interaction Diagram Editor
(net.sf.ictalive.operetta.diagram.is)
Graphical Norm Editor
(planned)
Verifies model
Validation Framework
(net.sf.ictalive.operetta.check)
Imports/Exports ontology
Ontology Management
(net.sf.ictalive.operetta.ontology)
Builds reorganisation scripts
Model Tracker
(net.sf.ictalive.operetta.modelTracker)

Fig. 4. OperettA Tool Components.

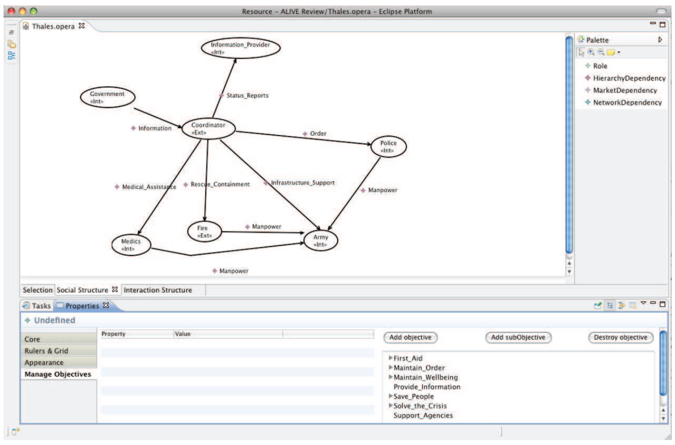


Fig. 5. OperettA Social Diagram Editor.

interaction diagram editor. A third graphical editor is planned for editing and managing formulas and norms.

Next to the graphical editing extensions, OperettA contains three other plug-ins for additional functionality. The Validation Framework provides an improvement over the default validation of EMF-based tools to provide validation of additional restrictions. An ontology managing plug-in is included as well to allow the ontology developed with OperettA to be exported to OWL, as well as allowing for importing existing ontology into the organization to boot-strap the organization design. Finally, OperettA contains a Model Tracker plug-in that can generate re-organization descriptions based on changes made in the OperettA organization editors.

We discuss the graphical editors and additional plug-ins in more details in the following.

*1) Social Diagram Editor:* This graphical editor provides a view of the Social Structure element of OMs. It allows the graphical creation of organizational Roles and Dependencies, thus specifying the social relations between important parties that play a part in the organization. The Social Diagram Editor also provides editing capabilities to specify and manage Role related Objectives, to provide context for the different Roles in an organization. Figure 5 depicts the Social Diagram Editor of OperettA that is used to enter organizational roles and dependencies between roles. Role objectives are created and managed via the objectives editor shown in the bottom part of the figure.

*2) Interaction Diagram Editor:* Similar to the Social Diagram Editor, the Interaction Diagram Editor provides a graphical view of the Interaction Structure element of OMs. This editor allows for the specification and management of the interaction elements of the organization; that is, it is for the specification and management of the different interactions that take place in the organization in order to achieve the different (role) objectives specified in the social part of the OM. The specification of the interaction is done in terms of scenes and transitions (the connection and synchronization points between scenes). Together, these define the order in which objectives are to be reached and how the organization works (though specified on a high level of abstraction). The Interaction Diagram Editor allows for the graphical creation and maintenance of scenes, transitions and arcs (links between scenes and transitions). The graphical editor provides a user-friendly overview of the structural aspect of the organization, defining how different interactions within the organization are supposed to help achieve the organizational objectives. Finally, OperettA allows for the specification and editing of scene properties (like the scene results, the players active in the scene, the landmark pattern, etc).

*3) Ontology Manager:* The Ontology Manager part of the OperettA tool is a plug-in for importing and exporting (domain) ontologies. The creation and maintenance of ontologies is done by external tools (like, for example, Protégé). Parts of the functionality of organizational ontology editing is included in the OperettA editors:

- Automatic creation of organizational ontology while designing the organization. As the designer is inputting the organizational model in OperettA, OperettA maintains an ontology of role names, objective names, and logical atoms that the designer uses to define the organization.
- Using an included (existing) ontology for the naming of organizational model elements; that is, if an (external) ontology is present in the organizational model it can be used to pick concept names for different parts of an organizational model (e.g., the name of a role can be picked from an existing ontology included in the model). The addition of the (external) ontology to an organizational model is done via the ontology manager.

The functionality of ontology editing in the OperettA tools is limited to organizational ontologies. The Ontology manager plug-in extends OperettA with the following capabilities:

- Importing an ontology from a file (e.g., RDF or OWL [19]). Ontologies about the domain or organization that is to provide the context of a system might be already available. These ontologies tend to be stored in some conventional ontology file-format. The Ontology Manager allows OperettA to import and use such ontologies.

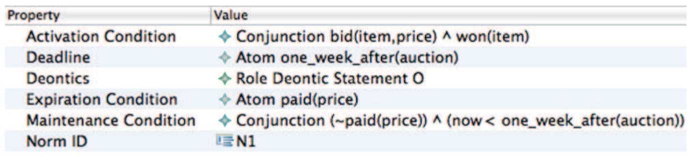
- Exporting (generated) organizational ontologies to file. In order to align an use the organizational ontology created by OperettA, the Ontology Manager extends the OperettA tool with the capability to export the default ontology to an owl file.

The organizational ontology created by OperettA is stored in the Organizational Model. The ontological elements need to be available to the system level of design, and thus need to be included in the domain ontology. The integration of organizational concepts in a domain ontology is not trivial, as it should respect the structure of the domain ontology while adding organizational concepts as roles, objectives, etc. and the instances of these concepts; role names, objective names, etc. The alignment between the exported ontology and the domain ontology will have to be done by hand in an external editor. The inclusion of the ontology manager satisfies requirement 3.

*4) Model Tracker:* To support reorganization, OperettA is extended with a model tracker. This model tracker allows a designer to view the changes made on the organizational model since a last save (but not necessarily the previous one). By storing the changes to the organizational model in a history file, the model tracker can be used to generate scripts that express how an organization is changed. Reorganization scripts capture changes in a precise and concise manner, and can be used to communicate organizational changes to the system level.

*5) Validation Framework:* The validation plug-in of OperettA overwrites the basic validation provided by the EMF framework. Instead of just verifying constraints specified in the OperA meta-model, the validation has been extended with additional verification constraints to minimize organizational design mistakes. The overall purpose of the validation plug-in is to provide OM designers meaningful feedback to eliminate design errors as early as possible (in the design process). The validation plug-in is installed separately from OperettA, but after installation it can be invoked from within each of the different OperettA editing views. The validation plug-in seamlessly overwrites the standard EMF validation, making it the new default manner of validating OperettA models.

The validation plug-in works directly on the model instance to verify various modeling constraints, accessing the model via the meta-model definitions. Some examples of the constraints validated are checking that roles have a name, checking that role names are unique, checking that all roles have an objective, and so on. Less stringent constraints are checked as well, like, for example, whether roles are connected to other roles via dependencies; i.e., while it does not hold for every OM, in most models roles should be connected to other roles (that is, it should be depending upon (an)other role(s) or being depended upon by (an)other role(s)). Such "soft" constraints are presented to designer as a warning, intended to have the designer rethink their model and update if appropriate. The validation plugin fulfills requirement 3.



| Property | Value |
|---|---|
| Activation Condition | Conjunction bid(item,price) ^ won(item) |
| Deadline | Atom one_week_after(auction) |
| Deontics | Role Deontic Statement O |
| Expiration Condition | Atom paid(price) |
| Maintenance Condition | Conjunction (~paid(price)) ^ (now < one_week_after(auction)) |
| Norm ID | N1 |

Fig. 6. A Norm in OperettA.

*6) Norm Editor:* Norms are an important part of the organizational Model, providing lead ways on a high level of abstraction for the agents to follow. Norms can be inputted in the current version of OperettA via the basic EMF generated editor. This editor is not user-friendly. An example norm in OperettA is shown below in figure 6. The norm shown in this figure describes that buyers should have paid for the items they have won in an auction before one week has expired. The norm is input using several logical formulas; one for the activation condition expressing when the norm is active (the item is bid on and won), one for the expiration condition expressing that the norm is no longer active (the item has been paid for), one for the maintenance condition expressing the formula to be checked when the norm is active to see if violations have happened (the buyer has not paid and the week has not yet expired), and one for the deadline expressing a state of affairs before which the norm should have been fulfilled. A user-friendlier (graphical) interface for inputting and managing the norms of an organizational model is planned for a future version of OperettA. This extension, with the graphical editors for the social and interaction structures, fulfills requirement 5.

## B. Connectivity to System Level

The OperettA tool has only off-line functionalities; it is used by designers to create the context of the system and their linked ontologies. It provides design and validation functionalities for the creation and management of OperA organizational models. For the connection to implementations OperettA depends on the Model Driven Engineering (MDE) approach by providing a meta-model of the modeling concepts.

MDE refers to the systematic use of models as primary artifacts throughout the Software Engineering lifecycle. The defining characteristics of MDE is the use of models to represent the important aspect of the system, be it requirements, high-level designs, user data structures, views, interoperability interfaces, test cases, or implementation-level artifacts such as code. The Model Driven Development promotes the automatic transformation of abstracted models into specific implementation technologies, by a series of predefined model transformations.

In essence, this means that the models created with the OperettA tool can be used for automated transformation towards applicable (models of) platforms; e.g, service-based implementations or multiagent systems. The only required step for such transformation is the definition of the transformations based on the OperettA meta-model concepts to the meta-model of the desired platform.

Finally, OperettA is based on the OperA formalism, which assumes that individual agents are designed independently from the organization, to model goals and capabilities of a given entity. Individual agents are the enactors of organizational role(s), as a means to realize their own goals [3]. As such it is necessary that OperettA can connect to such MAS frameworks. As a proof of concept, we have done experiments on the connection towards frameworks like Brahms and Repast, for the simulation of organizations in which normative properties of the organization can be verified for different populations with emergent behavior. As part of the ALIVE project [12] a connection was made with AgentScape to generate MAS from the organizational specification.

## V. Design Guidelines

In the previous we introduced organizational modeling and the OperettA Environment to support this. In this section we present a small overview on how one goes about designing an organization. After identifying that an organization presents the solution to the problem:

1) Identify (functional) requirements: First one determines the global functionalities and objectives of the society.
2) Identify stakeholders: The analysis of the objectives of the stakeholders identifies the operational roles in the society. These first two steps set the basis of the social structure of the OperA model.
3) Set social norms, define normative expectations: The analysis of the requirements and characteristics of the domain results in the specification of the normative characteristics of the society. This results in the norms in the normative structure.
4) Refine behavior: Using *means-end* and *contribution* analysis, a match can be made between what roles should provide and what roles can provide. This aspect contributes to refinement of role objectives and rights.
5) Create interaction scripts: Using the results from steps 3 and 4, one can now specify the patterns of interaction for the organization, resulting in the interaction structure.

More details about the methodological steps taken to create organizational models can be found in [7].

## VI. Conclusions

In this paper, we present an organization-oriented modeling approach for system development. The OperA modeling framework can be used for different types of domains from closed to open environments and takes into consideration the differences between global and individual concerns. The OperettA tool supports software and services engineering based on the OperA modeling framework. It has been used in the European project ALIVE [12] that combines cutting edge coordination technology and organization models to provide flexible, high-level means to model the structure of interactions between services in an environment.

## References

[1] Huib Aldewereld, Sergio Álvarez-Napagao, Frank Dignum, and Javier Vázquez-Salceda. Engineering social reality with inheritance relations. In *Proc. of the 10th Workshop Engineering Societies in the Agents' World (ESAW 2009)*. 2009.

[2] L. Coutinho, J. Sichman, and O. Boissier. Modelling dimensions for agent organizations. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. Information Science Reference, 2009.

[3] M. Dastani, V. Dignum, and F. Dignum. Role assignment in open agent societies. In *AAMAS03*. ACM Press, July 2003.

[4] V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. SIKS Dissertation Series 2004-1. Utrecht University, 2004. PhD Thesis.

[5] V. Dignum. The role of organization in agent systems. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages ??–?? Information Science Reference, 2009.

[6] V. Dignum and F. Dignum. Designing agent systems: State of the practice. *International Journal on Agent-Oriented Software Engineering*, 4(3), 2010.

[7] V. Dignum, F. Dignum, and J.J. Meyer. An agent-mediated approach to the support of knowledge sharing in organizations. *Knowledge Engineering Review*, 19(2):147–174, 2004.

[8] V. Dignum, J. Vazquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems: Second International Workshop ProMAS 2004*, volume 3346 of *LNAI*. Springer, 2005.

[9] Virginia Dignum and Frank Dignum. Modeling agent societies: coordination frameworks and institutions. In A. Jorge P. Brazdil, editor, *Progress in Artificial Intelligence: Proc. of EPIA-2001*, LNAI 2258, pages 191–204. Springer, 2001.

[10] M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In *ATAL-2001*, LNAI 2333, pages 348–366. Springer, 2001.

[11] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *ICMAS'98*, pages 128–135. IEEE Computer Society, 1998.

[12] European Commission FP7-215890. ALIVE, 2009. http://www.ist-alive.eu/.

[13] Davide Grossi, Frank Dignum, Mehdi Dastani, and Lambèr Royakkers. Foundations of organizational structures in multiagent systems. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 690–697, New York, NY, USA, 2005. ACM.

[14] S. Kumar, M. Huber, P. Cohen, and D. McGee. Towards a formalism for conversation protocols using joint intention theory. *Computational Intelligence Journal*, 18(2), 2002.

[15] H.V.D. Parunak and J. Odell. Representing social structures in uml. In M.Wooldridge, G.Weiss, and P. Ciancarini, editors, *Agent-Oriented Software Engineering II*, LNCS 2222. Springer-Verlag, 2002.

[16] L. Penserini, D. Grossi, F. Dignum, V. Dignum, and H. Aldewereld. Evaluating organizational configurations. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2009)*, 2009.

[17] I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback. Designing conversation policies using joint intention theory. In *Proc. ICMAS-98*, pages 269–276. IEEE Press, 1998.

[18] Dave Steinberg, Frank Budinsky, Marcelo Paternostro, and Ed Merks. *EMF: Eclipse Modeling Framework*. Eclipse Series. Addison-Wesley Professional, 2008.

[19] W3C. Owl-s, 2004. http://www.w3c.org/Submission/OWL-S.

[20] H. Weigand and V. Dignum. I am autonomous, you are autonomous. In M. Nickles, M. Rovatsos, and G. Weiss, editors, *Agents and Computational Autonomy*, volume 2969 of *LNCS*, pages 227–236. Springer, 2004.

[21] F. Zambonelli. Abstractions and infrastructures for the design and development of mobile agent organizations. LNAI 2222, pages 245–262. Springer, 2002.

[22] F. Zambonelli, N. Jennings, and M. Wooldridge. Organizational abstractions for the analysis and design of multi agent systems. LNAI 1957, pages 235–251. Springer, 2001.