applied sciences

MDPI

Article

Towards Agent Organizations Interoperability: A Model Driven Engineering Approach

Luciano R. Coutinho [1], Anarosa A. F. Brandão [2], Olivier Boissier [3] and Jaime S. Sichman [2,*]

[1] Departamento de Informática (DEINF), Universidade Federal do Maranhão (UFMA), Av. dos Portugueses, 1966, Bacanga, São Luís 65080-805, Brazil; luciano.rc@ufma.br
[2] Laboratório de Técnicas Inteligentes (LTI), Escola Politécnica (EP), Universidade de São Paulo (USP), Av. Prof. Luciano Gualberto, 158, travessa 3, São Paulo 05508-010, Brazil; anarosa.brandao@usp.br
[3] Department of Computer Science and Intelligent Systems, ENS Mines Saint-Etienne, 158, cours Fauriel, CEDEX 2, 42023 Saint-Étienne, France; Olivier.Boissier@emse.fr
* Correspondence: jaime.sichman@usp.br; Tel.: +55-11-3091-0625

Received: 4 May 2019; Accepted: 7 June 2019; Published: 13 June 2019

Abstract: In the research and development of *multiagent systems* (MAS), one of the central issues is how to conciliate the autonomy of the agents with a desirable and stable behavior of the MAS as a whole. *Agent organizations* have been proposed as a suitable metaphor for engineering social order in MAS. However, this emphasis has led to several proposals of *organizational models* for MAS design, thus creating an *organizational interoperability problem*: How to ensure that agents, possibly designed to work with different organizational models, could interact and collectively solve problems? In this paper, we have adopted techniques from *Model Driven Engineering* to handle this problem. In particular, we propose an abstract and integrated view of the main concepts that have been used to specify agent organizations, based on several organizational models present in the literature. We apply this integrated view to design MAORI, a model-based architecture for organizational interoperability. We present a MAORI application example that has shown that our approach is computationally feasible, enabling agents endowed with heterogeneous organizational models to cooperatively solve a problem.

Keywords: interoperability; multiagent systems; organizational models

1. Introduction

In the research and development of *multiagent systems* (MAS), one of the central issues is how to conciliate the autonomy of the agents with a desirable and stable behavior of the MAS as a whole. Borrowing ideas from the Social Sciences, some authors have named this issue the *problem of social order*: "How to obtain from local design and programming, and from local actions, interests, and views, some desirable and relatively predictable/stable emergent result" [1]. A closely related issue is the *problem of social consensus* often characterized as how to reach agreement with regard to some aspect or quantity of interest in a network of agents by combining the local preferences or states of individual agents [2,3]. Both social order and social consensus are fundamental problems in the design of MAS. While social order stresses the idea that agent behaviors must be coherent with the MAS global purpose, social consensus highlights the need of agreement among agents working together for a global purpose.

Faced with these problems, especially in the context of open MAS (i.e., systems formed by a dynamical population of agents provided by different developers), several researchers have argued in favour of using the *human organizations* as a proper metaphor for engineering MAS [4–7]. Human organizations, whose typical examples are firms, clubs, corporations, etc., are *collectivities* pursuing *specific goals* and exhibiting *formalized social structures* [8]. Goals are specific to the extent that they

are explicitly and clearly defined. Social structures are formalized in such a way that patterns of structuring and behaving (such as *roles*, *role relations*, *procedures*, *protocols*, *norms*, etc.) are precisely specified regardless of personal traits and relations of any individual part of the organization. Thus, by conceiving a MAS as an organization—or more generally as a bigger system formed by several organizations, hereafter called *agent organizations*—the basic idea is to promote social order and consensus in a top-down fashion. The idea is to have the agents' actions and interactions governed by formalized "social structures", defined above the agents level, in order to enable the MAS (seen as a collective entity) to achieve definite global goals.

This emphasis on organizations as a suitable metaphor for engineering social order has led to several proposals of *organizational models* for MAS design [4]. From the perspective of software development, most of these organizational models can be characterized as *domain specific modeling languages* [9]. That is, they provide a specialized *conceptual structure* (metamodel) embodied in some *concrete syntax* (notation) by means of which the designer can write formal representations of the social structure of agent organizations. Such representations, called *organizational specifications*, are then used as specification artifacts driving the development of agents and MAS.

On the one hand, the existence of many organizational models favours the organizational design of MAS since, with various proposals, experience and best practices are accumulated. On the other hand, a great variety of organizational models introduces heterogeneity in the development of MAS. As a direct consequence of this heterogeneity, mainly in the case of open MAS, a new and important interoperability issue arises: If to enter and fully work in an organization the agents should be designed to "understand" and comply with an organizational specification of a given kind (i.e., conforming to some organizational model), then, in addition to the communication language and the domain ontology, the organizational model is something that the agents are supposed to share in order to properly work together. In other words, how can we provide means for a set of agents, immersed in a common environment, to evolve, reason, decide and interact with each other based on organizational concepts, since their organizational models may differ? In this paper, we call this issue the *organizational interoperability problem*.

We can think about four basic approaches to solve the organizational interoperability problem: *Standardization*, *universal agents*, *delegation* and *adaptation*. Standardization consists in providing interoperability by eliminating the root of the problem, the diversity, by means of a *standard model* that has to be accepted and used by all developers [10]. The universal agents approach implies the creation of agents which are able to deal with several different organizational models [11]. Delegation means creating specialized services in middleware layers (like proxies [12] and governors [13]) to whom agents may delegate reasoning and decision mechanisms related to organizational issues. Adaptation, by its turn, is a solution based on the possibility of defining mappings between models [14]. From these mappings, an *adapter* is created, a component that converts specifications from a model to another model [15,16].

Each of these approaches have their pros and cons. Standardization fully eliminates the problem but it is politically and economically difficult to achieve and lets aside legacy systems. Universal agents must be updated every time a new model is created or changed. Delegation practically vanishes the agents organizational autonomy. Adaptation deals with legacy systems but it is technically difficult to achieve, if not impossible, when there are no meaningful mappings between the models.

Motivated by the organizational interoperability problem and the basic approaches to it, all of which presuppose an integrated knowledge of the organizational models used to engineer agent organizations, the objective of this paper is to analyze the conceptual structures of several organizational models present in the literature and, based on this analysis, to propose an abstract and integrated view of the main concepts that have been used to specify agent organizations. We believe that the abstract view of organizational models we put forward can be used both as basis for defining essential mappings and for future standardization efforts of organizational models.

This work is based on a previous work [17] in which we did a review of several prominent organizational models to answer the questions: How are the conceptual structures of the models related? Are there basic similarities? What are they? The answers we have given to these questions, in terms of *modeling dimensions*, are summarized in Section 2. The idea of modeling dimensions describes the organizational models basic similarities in broad terms. It characterizes the macrostructure of organizational models.

In this paper we deepen our analysis by further exploring the conceptual structure within each modeling dimension. In this sense, we seek to characterize the common concepts and their relations found in existing organizational models along the modeling dimensions. The specific questions we propose to answer in this paper are: Inside each modeling dimension, what are the recurring modeling concepts? Is it possible to combine these recurring concepts into a coherent whole? How this can be used in a solution to the organizational interoperability problem? In approaching these questions, we have used techniques from Model Driven Engineering (MDE) [18]. Specifically, we think of organizational models as domain specific modeling languages whose conceptual structures are represented by means of metamodels. Thus, to address the questions systematically, we propose an iterative integration method, described in Section 3, aiming at building an integrated metamodel out of particular metamodels.

A central step in the integration method is the identification of correspondences between the conceptual structure of organizational models represented by metamodels. To assist this identification, in Section 4, we analyze the recurring concepts of existing organizational models along the modeling dimensions. The result is an *abstract conceptual structure* formed by the union of *conceptual patterns* found by comparing the organizational models. Relying on this abstract conceptual structure and using the proposed integration method, in Section 5 we show how to effectively integrate (part of) three existing organizational models. To put into perspective the integration of organizational models, we then discuss in Section 6 a solution based on adaptation for the organizational interoperability problem. In this solution, named MAORI (*Model-based Architecture for ORganizational Interoperability*) [19], the mappings between organizational models are defined indirectly by using the integration we have proposed.

In Section 7, we compare our proposal to related work in the literature. To the best of our knowledge, the systematic integrated analysis of organizational models we propose is novel, constituting the main contribution of the paper. Its importance, as already hinted, lies on serving as a common ground for aligning organizational models and as a starting point towards standardization. The MDE approach we apply is also a contribution and advancement in the state of the art. Looking at the literature, we found that few organizational models are defined in terms of explicit metamodels. Then, our representation of existing organizational models and their integration by means of formal metamodels helps in further the understanding of their features and limitations. Finally, in Section 8, we present our conclusions and future work.

2. Organizational Models for MAS

This section presents organizational models for multiagent systems by classifying their content in modeling dimensions that were adopted to define a method for the integration of organizational models. As a result of using the method, we build an *abstract conceptual structure* to deal with the organizational interoperability problem within MAS.

2.1. Modeling Dimensions

After a detailed analysis of a significant part of the existing organizational models, we have noted a lot of similarities and complementary issues regarding their conceptual structures. The common points identified were classified into some recurring themes we have called *modeling dimensions* for agent organizations [17]. In what follows, we discuss the modeling dimensions identified. Then we

move to a short overview of the various organizational models analyzed. Ending the section, we show a comparative table summarizing the models analyzed along the modeling dimensions identified.

2.1.1. Fundamental Aspects of Systems

In general, designed systems exhibit some fundamental aspects that, from an engineering standpoint, are natural candidates for modeling. Firstly, there is the *functional behavior* of the system—the input (stimulus) to output (response) relations that couple the system to external elements composing the environment in which the system is situated. In modeling this aspect, the system is commonly depicted as a black box whose internal constitution, at first, does not matter. What really matters is that the environment imposes functional requirements, such as operations or tasks, that the system as a whole is supposed to perform. Further, these functional requirements may be subdivided in a recursive way until reaching atomic operations or tasks arranged in a given ordering (dependency graph). Later, when the innards are determined, the actual execution of the atomic operations or tasks can be associated with specific components of the system and their interactions. Modeling the functional behavior is a common practice both in developing computer systems and in representing organizational processes. Modeling techniques such as DFD (Data Flow Diagram) [20], and the activity diagrams of UML (Unified Modeling Language) [21] are typical examples of that.

Another fundamental aspect is the *internal structure* of the system. In contrast, to model the internal structure of a system means to represent it as a transparent box. It means to represent the break down of the system in its constituent parts (components and subsystems) and the relations interconnecting these parts. Like functional modeling, modeling the internal structure of a system is a recurring theme in system design. In software development, for example, the class diagrams and the component diagrams of UML serve to this purpose. In the case of human organizations, a traditional form of structural modeling is the creation of organograms describing the divisions, roles and hierarchical relationships existing inside an organization.

A third candidate for modeling is the *structural behavior* of the system. Roughly, it consists of the "movement" of the internal structure of a system towards the realization of some desired functional behavior. Thus, when modeling the structural behavior, we also see the system as a transparent box. What we try to represent is the ordering of interactions occurring over time among the constituent parts of a system. These interactions make the system work, i.e., perform some expected task or operation in its environment. Examples of this type of modeling are the sequence and collaboration diagrams of UML.

2.1.2. Primary Modeling Dimensions

From the premise that designed systems in general, not only agent organizations, exhibit these three fundamental aspects as natural modeling concerns, we have used them as a first classification scheme for separating the modeling concepts of organizational models into cohesive categories. Consequently, we define:

- The *functional dimension*, in which we place the modeling concepts used to represent the functional behavior of agent organizations;
- the *structural dimension*, composed by modeling concepts used to represent the internal structure of agent organizations; and
- the *interactive* or *dialogical dimension*, grouping the modeling concepts relative to the representation of the structural behavior of agent organizations.

2.1.3. Social Systems and the Normative Dimension

While the functional, structural and interactive dimensions can be justified by analysing the modeling of systems in general, they are not sufficient to classify all modeling concepts appearing in organizational models.

According to [22], three basic types of systems and corresponding models can be identified: *Deterministic*, i.e., systems and models in which neither the parts nor the whole are purposeful; *animated*, i.e., systems and models in which the whole is purposeful but the parts are not; and *social*, i.e., systems and models in which both the whole and the parts are purposeful. A fourth type is also considered, *ecological*, i.e., systems and models in which the parts are purposeful but the whole is not. Given this classification, we can say that traditional software systems are deterministic, autonomous agents are animated, and agent organizations are social. Bigger and more encompassing MAS aggregating agents and agent organizations form ecological systems.

Being deterministic, traditional software systems tend to have an architecture in which the functional behavior, the internal structure and the structural behavior are foreseen in detail. Their components are not conceived as purposeful entities with autonomous behaviors. On the contrary, they are designed to obey rigidly what is fixed in the architectural specification of the system.

Regarding agent organizations characterized as social systems, the idea of agents obeying rigidly the prescriptions of functional, structural and interactive specifications is not realistic. Agents are conceived as self interested components, especially in open MAS. Therefore, neither the functional, structural and interactive specifications can be very detailed to the point of precisely determining the minutiae of the joint structuring and behaving of the agents, nor one can assume benevolence from the agents with respect to the organizational goals.

In this context our analysis is that the specification of *norms* (permissions, prohibitions, obligations, etc.), as occurs in human organizations design, are also expected to show up in organizational models. They will work as a complementary mechanism helping to couple more flexibly the agents to the organization. On the one hand, norms provide explicit means to capture interdependencies among the functional, structural and interactive aspects (e.g., agent playing a given role in the internal structure is obliged to behave functionally or structurally in a given way). On the other hand, norms can be used to explicitly regulate sanctions or penalties to deviant behavior.

Accordingly we define a fourth and last category for the analysis of organizational modeling concepts:

- The *normative dimension*, characterized by modeling concepts to further restrict, regulate and interrelate elements from the other modeling dimensions, given the expected autonomous behavior of the agents.

2.2. Models Review

Now we pass to a quick description of concrete organizational models taking into account how they cover the four dimensions of modeling identified. We describe six models—TAEMS [23], AGR [5], STEAM [24], MOISE+ [25], ISLANDER [26] and OPERA [27]. We think of these as good exemplars showing how models have evolved towards a full coverage of the organizational modeling dimensions. Other models are mentioned at the end of the review.

2.2.1. TAEMS

In TAEMS (Task Analysis, Environment Modeling, and Simulation) the basic modeling concept is the notion of *task*. In essence, by using TAEMS we can specify *tasks structures* composed by the definition of *tasks*, *resources*, *tasks relationships*, and *task groups*. A task group is an independent collection of interrelated tasks. There are two kinds of task relationships: *Subtask* and *non-local effects* relationships. The subtask relationship links a parent task to child task explicitly defining a task decomposition tree. Individual tasks that do not have child tasks are called *methods*. Methods are primitive tasks that agents should be able to perform. Non-local effects are task relationships that have positive or negative effects in the quality, costs or duration of the related tasks. Examples of possible non-local effects are: *Facilitates*, *enables*, *hinders*, *limits*, etc.

TAEMS is a model specialized exclusively in the specification of the functional behavior of agent organizations. A TEAMS specification, the task structure, only represents what should be done by

the agents alone (method definitions) or in groups (task groups). It tells nothing about the internal structuring or explicit interactions to realize the specified tasks.

2.2.2. ARG

AGR (Agent, Group, Role) is the evolution of the AALAADIN model [28]. In AGR *agent*, *group* and *role* are the primitive modeling concepts. An agent is an active, communicating entity playing one or more roles within one or more groups. No constraints are placed upon the architecture of an agent or about its mental capabilities. A group is a set of agents sharing some common characteristics. A group is the context for a pattern of activities and is used for partitioning organizations. An agent can participate at the same time in one or more groups. Agents may communicate if and only if they belong to the same group. A role is the abstract representation of a functional position of an agent in a group. Roles are local to groups, and a role must be requested by an agent.

AGR is a model providing a minimalist structural view of organizations. There is no concepts for modeling functional behavior. The specification of an organization, called *organizational structure*, is in essence the depiction of the internal structure of the organization in terms of roles, roles constraints and group structures. AGR also says that agents can have their joint behavior orchestrated by interaction protocols, but the nature and the primitives to describe such protocols are left open.

2.2.3. STEAM

STEAM (a Shell for TEAMwork) is a model whose focus is teamwork. In STEAM an agent organization is conceived as an *agent team*. Two separate hierarchies are used to specify the internal structure and functional behavior of a team: A *subteam* and *roles* hierarchy (or *organization hierarchy*), and a hierarchy of joint activities (or *operator hierarchy*). The subteam and roles hierarchy is a tree in which the root represents a team, the internal nodes the possible subteams and the leaves the individual agent roles. The joint activity hierarchy is also a tree whose nodes are called *operators*. Leaf operators represent atomic activities. Internal operators represent a *reactive plan*, i.e., the decomposition of an activity into interrelated subactivities. For each individual role or subteam, it is assigned one or more operators from the activity hierarchy.

With STEAM we see a first model that combines the structural (subteams and role hierarchy) and functional modeling dimensions (operator hierarchy).

2.2.4. MOISE+

MOISE+ (Model of Organization for multI-agent SystEms) is a model that explicitly divides the specification of an agent organization in three parts: The *structural*, the *functional* and the *deontic* specifications. The structural specification defines the internal structuring of agents through the notions of *roles*, *roles relations* and *group specifications*. A role defines a set of constraints the agent has to accept to enter in a group. Role relations are *links* (*communication*, *acquaintance* and *authority*) and *compatibilities* from a source role to a target role. A group specification consists in role definitions, subgroup definitions (group decomposition), links and compatibilities definitions, role cardinalities and subgroup cardinalities. The functional specification describes how an agent organization usually achieves its *global goals*, i.e., how these goals are decomposed (by *plans*) and distributed to the agents (by *missions*). Global goals, plans and missions are specified by means of a *social scheme*. A social scheme can be seen as a goal decomposition tree where the root is a global goal, the internal nodes are *plan operators* (sequence, choice, parallel) to decompose goals into subgoals, and the leaves are atomic goals that can be achieved by an individual agent. Missions are coherent sets of goals; hence, an agent that is committed to a mission is responsible for the satisfaction of all its component goals. Finally, the deontic specification associates roles to missions by means of *permissions* and *obligations*.

Like STEAM, MOISE+ addresses both the functional and structural dimensions of modeling. However, MOISE+ goes further and provides concepts for modeling normative aspects (deontic

specification). The deontic concepts allow a flexible coupling between the functional and structural specifications that is not seen in STEAM.

2.2.5. ISLANDER

ISLANDER is a declarative language for specifying *electronic institutions*. According to ([26] p. 348), "Institutions establish how interactions of a certain sort will and must be structured within an organization". In ISLANDER, an electronic institution is composed of four basic elements: A *dialogic framework*, *scenes* definitions, a *performative structure*, and *norms* definitions. In the dialogic framework it is defined the participating *roles* and their *relationships*. Each role defines a pattern of behavior within the institution and any agent within an institution is required to adopt some of them. A scene is a collection of agents playing different roles in interaction with each other in order to realize a given activity. Every scene follows a well-defined *communication protocol*. The performative structure establishes relationships among scenes. The idea is to specify a network of scenes that characterizes more complex activities. The norms component of an electronic institution defines the *commitments*, *obligations* and *rights* of participating roles.

With ISLANDER we perceive a change of focus from functional to structural behavior. Unlike the previous models, there is no concepts for explicitly modeling goals and plans (goal decompositions). All behavior is specified by means of direct interactions between roles (dialogs) and regulated by the definitions of norms.

2.2.6. OPERA

In OperA (Organizations per Agents) an agent organization is specified in terms of four structures: The social, the interaction, the normative and the communicative structures. In the *social structure* are defined *roles*, *objectives*, *groups* and *role dependencies*. Roles identify activities and services necessary to achieve social objectives. Groups provide means to collectively refer to a set of roles. Role dependencies describe how the roles are related in terms of objective realization. The *interaction structure* defines how the main activity of an agent organization is supposed to happen. This definition is done in terms of *scenes*, *scene scripts*, *scene transitions* and *role evolution relations*. Scenes are representations of specific interactions. A scene script is described by its players (roles or groups), scene norms (expected behavior of actors in a scene) and a desired interaction pattern. Scene transitions are used to coordinate scenes by defining the ordering and synchronization of the scenes. Role evolution relations specify the constraints that hold for the role-enacting agents as they move from scene to scene respecting the defined transitions. The normative structure gathers all the norms that are defined during the specification of roles, groups, and scene scripts. Norms are specified as formal logical expressions. Finally, the communicative structure describes the set of performatives and the domain concepts used in the interaction structure by the role enacting agents.

OPERA is a model that addresses all the identified modeling dimensions. Nevertheless, we note that the functional and structural modeling of OPERA is less developed than the others models, the interactive modeling is comparable to what is found in ISLANDER, and the normative modeling is the most elaborated of all models analysed. In OPERA norms are expressed in a formalism called LCR (Logic for Contract Representation).

2.2.7. Other Models

The literature on organizational models is vast. For reasons of space and scope, we briefly mention other models below:

- ODML—*Organizational Design Modeling Language* [29]—a minimalist organizational model that provides elements to model and evaluate structural aspects of organizations.

- AGRE—*Agent, Group, Role, Environment* [30]—an extension of AGR that takes into account physical and social environments. The main idea is that agents are situated in domains called *spaces*. The spaces can be physical (*areas*) or social (*groups*).
- MOISE^{Inst} [31]—an extension of MOISE+ that allows for *Contextual Specifications* (*contexts* and transitions between contexts) and *Normative Specifications* (*norms* set) in the modeling of an MAS organization.
- OMNI—*Organizational Model for Normative Institutions* [32]—an unification of two other models: the OperA and the HarmonIA framework [33].
- MAS-ML—*MAS Modeling Language* [34]—a modeling language that extends UML with elements of the TAO conceptual framework [35]. Regarding organizational modeling, one of the distinguishing features of MAS-ML is that organization technically are an extension of an *AgentClass* classifier. This means that organizations are conceived as kinds of agents.
- MACODO [36] is an organizational model for context-driven dynamic agent organizations where organization, agent, role and context are abstract concepts related to the system structure; capabilities, role positions and role contracts are related to the system functioning and there are laws for governing inter-organizational (merge law) or intra-organizational (join law) interactions. MACODO also provide a middleware that allows the implementation and execution of systems modeled following the MACODO model.

In addition to these, we also find in the literature on agent oriented software engineering (AOSE) methodologies a strong concern about organization modeling during the analysis and design of MAS. The Gaia [37] and Tropos [38] are some examples of AOSE methodologies that incorporate the concept of organization in their metamodels.

2.3. Models Comparison

More than large differences, we perceive several similarities and complementarity among the conceptual structures of the organizational models analysed, as we show in Table 1. The commonalities occur in two levels. In a first macro level, they occur as the dimensions of organizational modeling that were identified. For example, all models except TAEMS present concepts to represent the internal structure of organizations (structural dimension). All models except AGR and ISLANDER promote the functional behavior modeling (functional dimension). ISLANDER and OPERA present very similar concepts for representing the structural behavior of organizations (dialogical dimension). Normative concepts appear in MOISE+, ISLANDER and OPERA (normative dimension).

Table 1. Organizational models comparison.

Model	Functional	Structural	Dialogical	Normative
TAEMS	method, task, subtask relation, non-local effect	none	none	none
AGR	none	role, group, role relation	interaction protocol	none
STEAM	operator, plan, dependency	team, individual role	none	none
MOISE+	goal, plan, mission	role, group, role relations	none	deontic relations
ISLANDER	none	roles, role relations	scene, transition, interaction protocols	obligation
OPERA	objective, subobjective	role, group, dependency	scene, transition, interaction patterns	obligation

On a more detailed level of analyses, one can still identify various modeling patterns within each dimension. In the next section, we propose an iterative integration method that rely on these patterns, whose formal description is presented in Section 4.

3. Method for the Integration of Organizational Models

We advocate that modeling dimensions for agent organizations are useful not only to analyse and compare organizational models, but also they serve as a starting point for the *conceptual integration* of organizational models. As we have mentioned in the Introduction, when the objective is to provide organizational interoperability, a consistent conceptual integration of organizational models is a fundamental and necessary element. In order to systematically perform such integration (having in mind the problem of organizational interoperability), we have defined an iterative integration method that is discussed in this section.

3.1. General Process

Let OM_1, OM_2, ..., OM_n be n organizational models to be integrated. In broad lines, we understand by a *conceptual integration* of $OM_1, \ldots, OM_n$ the process of representing, correlating and joining the *conceptual structure* (i.e., the modeling concepts and their interrelationships) of each OM_i obtaining as the final result an *integrated metamodel* MM^{int} whose conceptual structure subsumes the structure of all OM_i. This idea of conceptual integration, as an iterative process, is depicted in Figure 1.

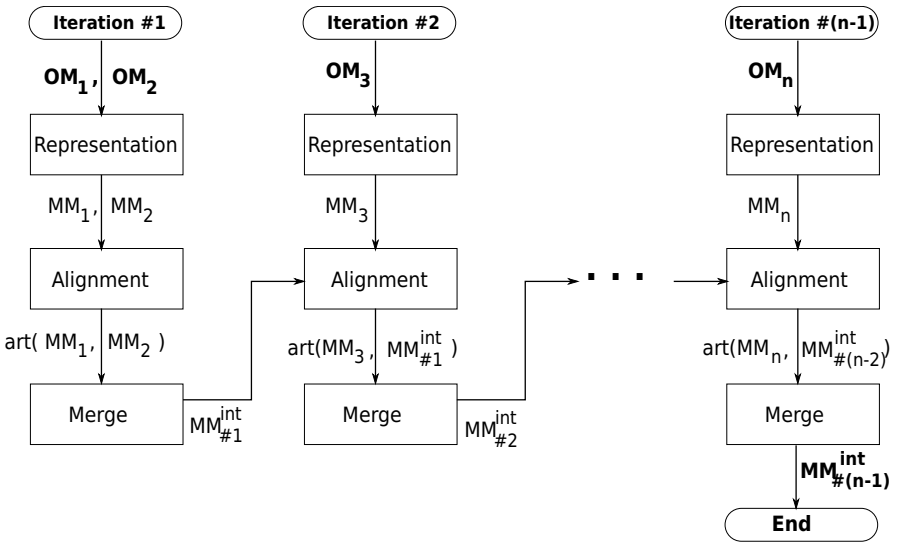


Figure 1. General conceptual integration method.

More specifically, for n models we have $n-1$ iterations. In the first one, three sequencial steps are performed for OM_1 and OM_2:

1. Analysis and representation of the conceptual structure of OM_1 and OM_2 as *metamodels* MM_1 and MM_2, respectively. In the representation it is used a common *metamodeling language*.
2. Comparison of MM_1 and MM_2, and identification of *correspondences* between the conceptual structure expressed in both metamodels. Such correspondences represent semantic overlapping areas between MM_1 and MM_2, i.e., modeling concepts and concept relationships that are assumed to have equivalent or similar interpretations. In general, the identified correspondences are explicitly expressed as *articulations* $art(MM_1, MM_2)$ between MM_1 and MM_2, as described next.
3. From MM_1, MM_2 and $art(MM_1, MM_2)$, we produce an integrated metamodel $MM^{int}_{\#1}$ that (*i*) is as expressive as both MM_1 and MM_2 (in the sense of retaining all concepts, relations and restrictions

found in both MM_1 and MM_2); and (*ii*) avoids unnecessary replication of elements which were declared equivalent or similar by means of the correspondences between MM_1 and MM_2.

From the second iteration onward, the same three steps are performed for each $OM_{3 \leq i \leq n}$ with the following differences: (1) Instead of MM_1 and MM_2 we have MM_i and $MM^{int}_{\#(i-2)}$, respectively; MM_i is the metamodel representing OM_i and $MM^{int}_{\#(i-2)}$ is the integrated metamodel from the previous iteration; (2) in the place of $art(MM_1, MM_2)$ we have $art(MM_i, MM^{int}_{\#(i-2)})$ which is the articulation between MM_i and $MM^{int}_{\#(i-2)}$; and (3) instead of $MM^{int}_{\#1}$ we produce $MM^{int}_{\#(i-1)}$, i.e., the integrated metamodel resulting from joining MM_i to $MM^{int}_{\#(i-2)}$.

3.2. Metamodel Representation

In the first step, we assume a metamodel based representation of the organizational models. In this sense, our method adopts the way by which special purpose modeling languages are defined in the area of Model-Driven Engineering. Given this assumption, there are several metamodeling languages available for expressing the conceptual structure of the organizational models. Some of these languages are KM3 [39], MOF/OCL [40,41], XMF [42] and Ecore [43]; this last is used in this work, as illustrated in Sections 4 and 5.

3.3. Metamodel Alignment

In the second step, the definition of correspondences between modeling concepts is an inherently heuristic process. One possible heuristic is to use the modeling dimensions identified in Section 2. The basic idea is to divide the work along the modeling dimensions. For each model, we start by classifying its modeling constructs in one or more dimensions. Then, for each dimension covered by the models, we identify the corresponding modeling constructs. In this way, the functional modeling concepts of one model are put into correspondence with the functional concepts of the other model, the structural concepts of one model with the structural concepts of the other, and so on.

Another heuristic we put forward for aligning the conceptual structure of organizational models is to take into account some basic conceptual patterns found in the models. These patterns are described in Section 4 in the form of an abstract organizational model.

3.4. Metamodel Merging

Unlike the alignment, the merging of metamodels is a more deterministic process that can be fully automated by using several algorithms reported in the literature [44–46]. In general, these proposals for (meta)model merging can be described as merging based on *graphs* and *morphisms*. In this case, the metamodels are abstractly conceived as graphs and the correspondences between two metamodels assume the form of an articulation between graphs. If MM_1 and MM_2 are two metamodels viewed as graphs, then an articulation $art(MM_1, MM_2)$ between them is a triple composed of a graph G^{art} and two morphisms $m_1 : G^{art} \rightarrow MM_1$ and $m_2 : G^{art} \rightarrow MM_2$:

$$art(MM_1, MM_2) =< G^{art}, m_1, m_2 >$$

Intuitively, the idea is that G^{art} is a representation of the common concepts and relations found in M_1 and M_2, and the morphisms m_1 and m_2 are the links mapping this common concepts and relations to their counterpart in both MM_1 and MM_2.

Once characterized as graphs, the merging of two metamodels MM_1 and MM_2 is in essence an *amalgamated sum* (or *pushout*) of MM_1 and MM_2, modulo $art(MM_1, MM_2)$:

$$merge(MM_1, MM_2) = MM_1 \oplus_{art(MM_1, MM_2)} MM_2 =< MM^{int}, m'_1, m'_2 >$$

where MM^{int} is the resulting integrated metamodel at the end of the iteration and m'_1 and m'_2 are morphisms $m'_1 : MM_1 \rightarrow G^{int}$, $m'_2 : MM_2 \rightarrow G^{int}$. The integrated metamodel MM^{int} consists of the union of the nodes (concepts) and edges (concept relationships) of MM_1 and MM_2, where correspondent elements as described via $art(MM_1, MM_2)$ are treated as only one element [44,46]. In this way MM^{int} retains all non-duplicate information in MM_1 and MM_2 collapsing the elements that $art(MM_1, MM_2)$ declares redundant. The morphisms $m'_1 : MM_1 \rightarrow G^{int}$ and $m'_2 : MM_2 \rightarrow G^{int}$ describe how translating from the particular to the integrated metamodel. The inverse, translating from the integrated to the particular metamodels, is performed via G^{art}, m_1 and m_2.

4. Abstract Conceptual Structure of Organizational Models

In this section we compare in detail the conceptual structure of the organization models discussed in Section 2. With this comparison, we intend to explicitly show two main findings: (*i*) We can identify patterns in the conceptual structure of organizational models inside each modeling dimension, if we homogenize the terminology used and abstract some particularities of each model; and (*ii*) the patterns identified can be consistently combined into a single conceptual structure (metamodel) that represents in an essential and integrated way the conceptual structures of organizational models. In the next subsections, we detail the basic patterns that emerge when one look more closely to the conceptual structures of the organizational models proposed in the literature. Each subsection focus on a modeling dimension previously discussed in Section 2. At the end, we combine the conceptual patterns obtaining in this way the abstract organizational metamodel.

4.1. Functional Dimension

In the functional dimension, we found concepts for the specification of the functional behavior of an agent organization, i.e., the collective behavior of agents when the internal structure of their organization is not taken into account. Looking at Table 1, we can see that this dimension occurs in TAEMS, STEAM, MOISE+ and OPERA. In these models, the functional specifications follow a general pattern which is illustrated in Figure 2.

4.1.1. Graphs of Hierarchical Plans and Goal Relationships

In essence, the general pattern can be characterized as directed graphs where:

- The nodes correspond to *goals* (in MOISE+), *operators* (in STEAM), *objectives* (in OPERA) or *tasks* (in TAEMS), to be achieved or done by the agents in an organization;
- the edges represent:
  - Either the acyclic decomposition of a *goal* (*operator*, *objective* or *task*) into *subgoals* (*suboperators*, *subobjectives* or *subtasks*), giving rise to the notion of *hierarchical plans*,
  - or binary relationships between *goals* (*operators*, *objectives* or *tasks*), like the *depends* relation in STEAM or the *non-local effects* in TAEMS.

Further, in each graph there is one *root node* that corresponds to a primary goal whose planning and future achievement is prescribed by the structure of the graph. Such graphs of hierarchical plans and goal relationships receive the names of "task group" in TAEMS, "operator hierarchy" in STEAM, "social scheme" in MOISE+ and "role objective definition" in OPERA.

In Figure 2, the conceptual pattern identified in the functional dimension is represented by means of an Ecore metamodel. The classes and references composing the metamodel are described in Table 2. Additional contextual constraints are presented in Table 3. These are written in OCL and formalize the static semantics of the metamodel (conceptual pattern).

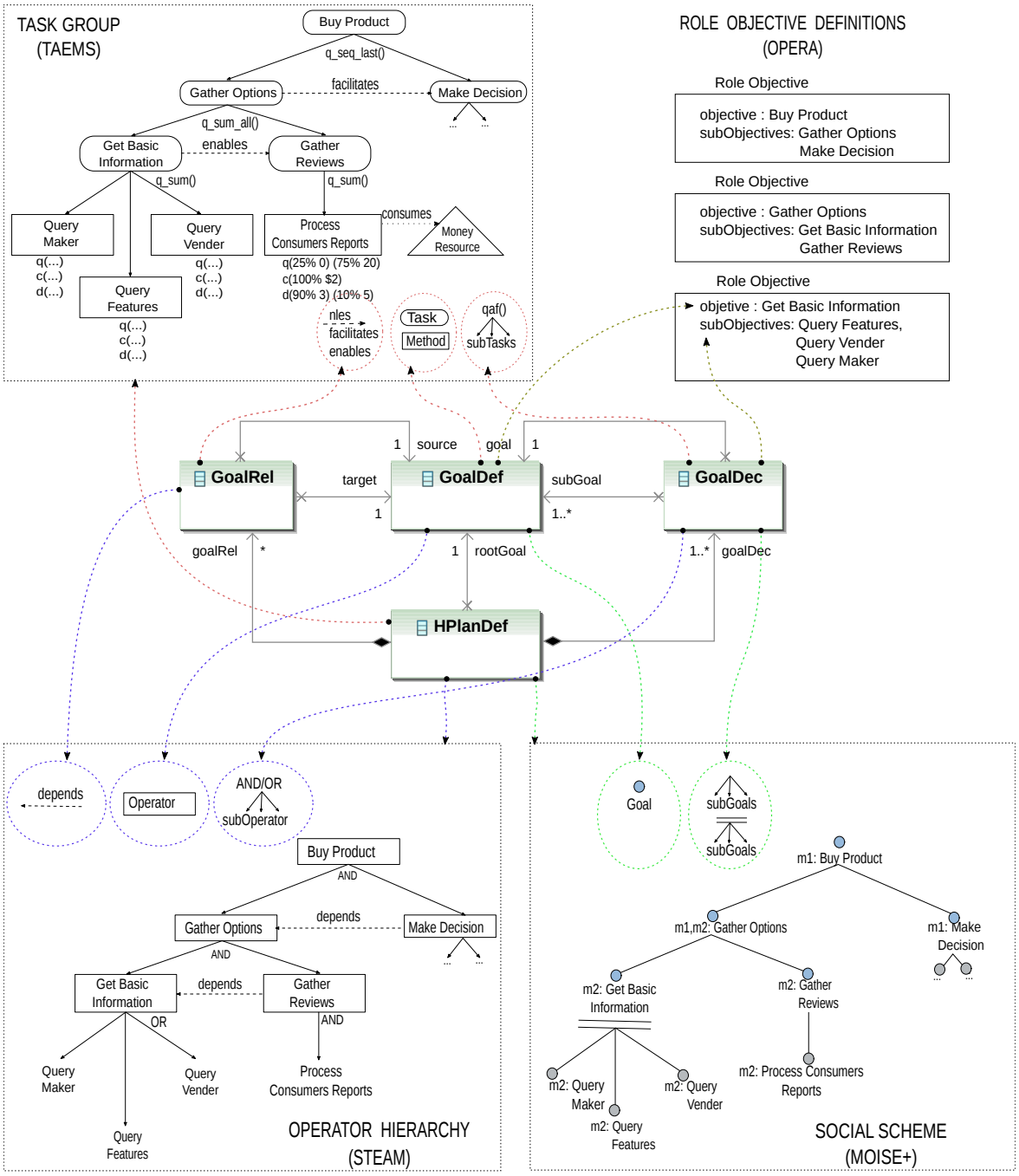
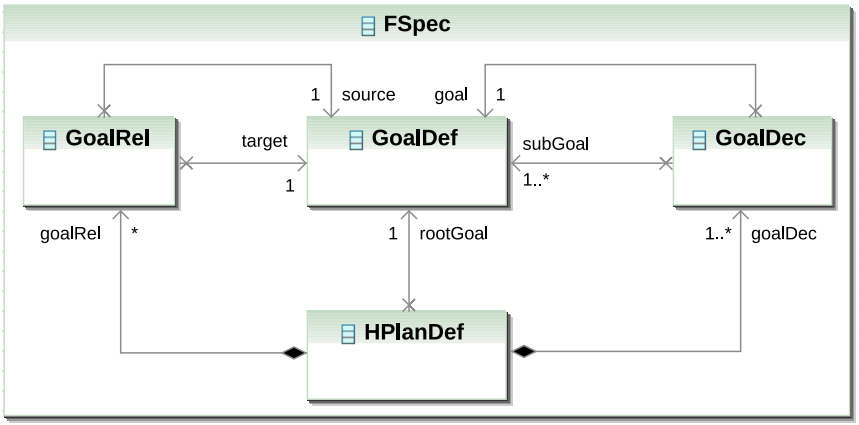


Figure 2. Similar functional specifications written in TAEMS, OPERA, STEAM and MOISE+, which prescribe how an agent is supposed to proceed for buying a product in an electronic market (example taken from [47]). The class diagram (Ecore metamodel) in the center of the figure represents the conceptual pattern identifiable in the various approaches of functional modeling. The dotted arrows detail what concepts are captured by what classes of the metamodel.

Table 2. Functional specification pattern: Classes description.



Class	Description
`FSpec`	Represents the concept of *functional specification*, i.e., an organizational specification restricted to the functional dimension.
`GoalDef`	Part of a functional specification that represents the definition of a *goal*. Abstracts the concepts of *task* in TAEMS, *operator* in STEAM, *goal* in MOISE+, or *objective* in OPERA.
`HPlanDef`	Part of a functional specification that represents the definition of a graph of *hierarchical plans* and *goal relationships*. `HPlanDef` is characterized by a unique *root goal* (the `GoalDef` referenced by `HPlanDef::rootGoal`), one or more *goal decompositions* (referenced by `HPlanDef::goalDec`) and zero or more *goal relationships* (referenced by `HPlanDef::goalRel`). Abstracts the concepts of *task group* in TAEMS, *operator hierarchy* in STEAM, *social scheme* in MOISE+ and *role objective definition* in OPERA.
`GoalDec`	Part of a `HPlanDef` graph that represents the general concept of *goal decomposition*, i.e., the decomposition of one major *goal*(`GoalDec::goal` reference) into one or more minor direct *subgoals*(`GoalDec::subGoal` reference). Abstracts the relationship of *subtasks* or *local effects* in TAEMS, the concept of *plan* in STEAM and MOISE+, and the *subobjective definitions* in OPERA.
`GoalRel`	Part of a `HPlanDef` graph that represents *goal relationships* directed from one *source goal*) `GoalRel::source` reference) to one *target goal* (`GoalRel::target` reference. Abstracts the concept of *non-local effects* (*facilitates*, *enables*, etc.) in TAEMS and the *depends* relation in STEAM.

Table 3. Functional specification pattern: Contextual constraints.

Class	Auxiliary Definitions
HPlanDef	::getAllSubGoal(goal:GoalDef) : Set(GoalDef), a query that returns the set of all subgoals in which a given goal is direct or indirectly decomposed in the context of a HPlanDef graph, via goal decomposition referenced by HPlanDef::goalDec In OCL: **context** HPlanDef **def:** getAllSubGoal(goal : GoalDef) : Set(GoalDef) = getSubGoal(goal) -> union(getSubGoal(goal) -> collect (sg | getAllSubGoal(sg))); **def:** getSubGoal(goal : GoalDef) : Set(GoalDef) = getGoalDec(goal) -> collect (gd | gd.subGoal); **def:** getGoalDec(goal : GoalDef) : Set(GoalDec) = goalDec(goal) -> serlect (gd | gd.goal = goal);
	Contextual Constraints
	(1) In the context of a HPlanDef graph, all decompositions of a goal into subgoals must be reachable from the root goal. In other words, each goal decomposed in subgoals in a HPlanDef graph is either the root goal itself or a direct or indirect subgoal of the root goal. In OCL: **context** HPlanDef **inv:** goalDec -> forAll (gd | root.Goal = gd.goal or getAllSubGoal(rootGoal) -> includes(gd.goal))
	(2) In the context of a HPlanDef graph, the source and target goals of all goal relationships must pertain to the collection of subgoals of the root goal of the graph. In other words, each goal relationship must connect only subgoals of the root goal of a HPlanDef graph. In OCL: **context** HPlanDef **inv:** getAllSubGoal(rootGoal) -> includesAll(goalRel -> collect (gr | Set(gr.source, gr.target)))
	(3) Restricted to goal decomposition, all HPlanDef graphs must be acyclic. In other words, in the context of a HPlanDef graph, no goal can be, direct or indirectly, a subgoal of itself. In OCL: **context** HPlanDef **inv:** not getAllSubGoal(rootGoal) -> includes(rootGoal) and getAllSubGoal(rootGoal) -> forallAll(sg | not getAllSubGoal(sg) -> includes(sg))

4.1.2. Particularities

Besides the common aspects represented in the functional specification pattern (Tables 2 and 3), there are some particular aspects in the organizational models that should be highlighted.

In TAEMS, there are the concepts of *resources* and *non-local effects between tasks and resources*. These concepts were not considered as part of the functional specification pattern presented because they occur only in TAEMS.

In MOISE+, there is no notion of *binary relationships between goals*. There is, however, the particular notions of *mission* and *preferences among missions*, which do not occur in the other organization models analysed, but only in MOISE+. Thus, like the concepts of *resources* and *resource non-local effects* of TAEMS, the notions of *mission* and *mission preferences* also do not appear in the functional specification pattern presented.

In OPERA, the functional modeling is done implicitly as part of a *role definition* (structural modeling). In this way, we have a functional modeling with little resources when compared to what can be found in TAEMS, STEAM and MOISE+. Quoting [27], a *role definition* is done specifying one or more *role objectives* γ and

> "[each] role objective γ can be further described by specifying a set of subobjectives that must hold in order to achieve objective γ. Subobjectives give an indication of how an objective should be achieved, that is, describe the states that must be part of any plan that an agent enacting the role will specify to achieve that objective. However, subobjectives abstract from any temporal issues that must be present in a plan, and as such must not be equated with plans." (pp. 60–61)

From this passage, we conclude that there is no explicit notion of *hierarchical plans* in OPERA. Even so, the general notion of *goal decomposition* can be identified in OPERA, as we have done in the previous subsection, by making it to correspond to the notion of *subobjectives specification* in OPERA (see Figure 2).

Lastly, we note that the abstract concepts of *goal decomposition* and *goal relationships* admit concrete subtypes of various kinds in the organizational models analysed. For instance, in TAEMS, the kinds of goal decomposition are denominated *quality accumulation functions* (*qafs*) and the kinds of goal relationships are named *non-local effects* (*nles*): Examples of *qafs* are `q_seq_last` (all subtasks must be completed in order, and overall quality is the quality of last task), `q_sum_all` (all subtasks must be completed in no specific order, and overall quality is the aggregate quality of all subtasks) and `q_exactly_one` (only one subtask may be performed, and overall quality is the quality of the single task performed); regarding specific *nles*, two of them are *facilitates* (when information from one task reduces or changes the search space making some other task easier to solve) and *enables* (when information from one task is a prerequisite for doing another task). In STEAM, there are two basic concrete subtypes of goal decomposition: AND (when all suboperators must be done to realize a given operator), and OR (when at least one suboperator must be done to realize a given operator), which are complemented by the *depends* relationship (that establishes a partial order for doing operators). Finally, in MOISE+, there is not the concept of *goal relationship*, only three subtypes of goal decomposition: *Sequence* (when subgoals must be achieved in some order to achieve a given goal), *choice* (when only one subgoal must be achieve to fulfill a given goal), and *parallelism* (when all subgoals can be pursued at the same time in order to achieve a given goal).

4.2. Structural Dimension

Forming the structural dimension, we have modeling concepts used to specify the internal structure of an agent organization in which the agents must engage to become an active member of the organization. From Table 1, five organizational models provide concepts for creating structural specifications. They are: AGR, TAEMS, MOISE+, ISLANDER and OPERA. Looking carefully at the structural specifications one is able to produce using these models, like the ones shown in Figure 3, we realize that the structural dimension of organizational modeling can also be characterized by an abstract conceptual pattern.

4.2.1. Graphs of Roles and Groups

In the structural dimension, the organizational models present three fundamental concepts: Role, group and role relationships. The instantiation of these three interrelated concepts forms the specification of the internal structure of agent organizations.

In essence, structural specifications can be characterized directed graphs where:

- The nodes correspond
  - either to the definition of *groups* (in AGR, MOISE+ and OPERA) or *teams* (in STEAMS),
  - or to the definition of *roles* (in all models);
- the edges represent
  - either the *decomposition of a group in subgroups* (or a team in subteams), forming a *group (team) hierarchy*,
  - or binary *relationships between roles*,

– or *links from a role to a group* (or *subteam*) in which the role can be played by agents.

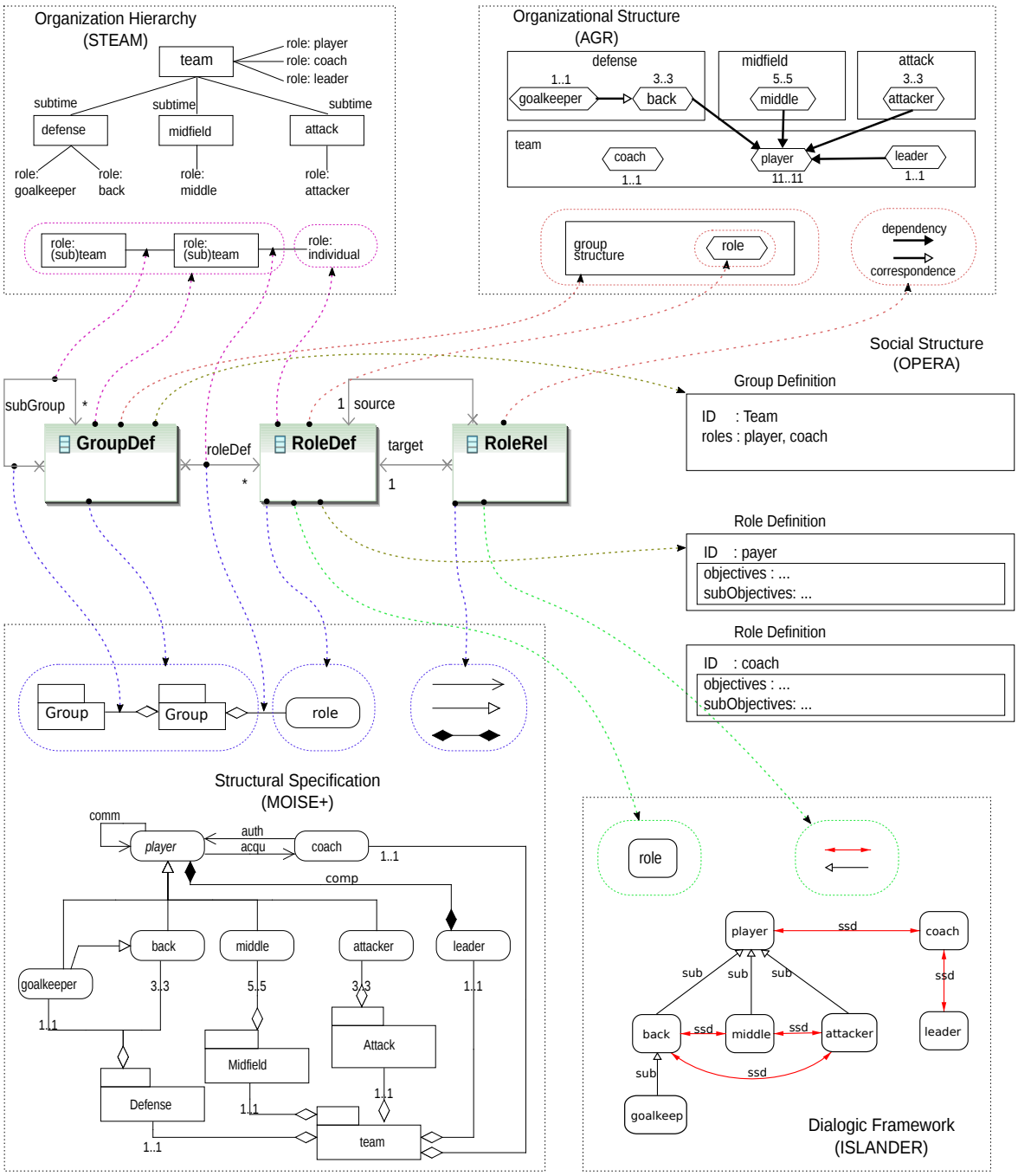


Figure 3. Similar structural specifications written in STEAM, AGR, OPERA, MOISE+ and ISLANDER, for a team of agents part of a simulated soccer game (example taken from [24,25]). The team is composed of eleven *players* (one *goalkeeper*, three *backs*, five *mid-filders* and three *attackers*) and one *coach*, and is divided in three groups: *Defense*, *midfield* and *attack*. The class diagram (Ecore metamodel) in the center of the figure represents the conceptual pattern identifiable in the various approaches of structural modeling. The dotted arrows detail what concepts are captured by what classes of the metamodel.

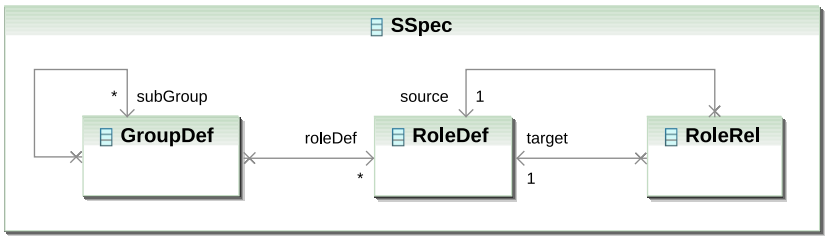
With regard to the *group* (*team*) *hierarchy*, there are two situations present in the models. On the one hand, we have a unique *root group* that represents the organization as a whole (the hierarchy is a rooted tree), what occurs in STEAM, MOISE+ and OPERA. On the other hand, there is no explicit *root group*, or even the decomposition of groups in subgroups, what is the case of AGR. Regarding the

binary relationships between roles, in general they are directed and, in the models AGR, MOISE+ and ISLANDER they are subdivided in various kinds with diverse interpretations.

Such graphs of roles and groups receive the names of "organization hierarchy" in STEAM, "organizational structure" in AGR, "structural specification" in MOISE+. In ISLANDER, they are part of the definition of a "dialogic framework". In OPERA, they form the "social structure".

In Figure 3, the conceptual pattern identified in the structural dimension is represented by means of an Ecore metamodel. This metamodel is presented in more detail in Table 4.

Table 4. Structural specification pattern.



Class	Description
`SSpec`	Represents the concept of *structural specification*, i.e., organizational specifications restricted to the structural dimension.
`GroupDef`	Part of a structural specification that represents a *group definition*. Each group definition is characterized by the declaration of the roles that agents can play in the group (the `RoleDef` referenced by `GroupDef::rootDef`), and by possible subgroup definitions (referenced by `GroupDef::subGroup`). Abstracts the concepts of *group structure* in AGR, *subteam role* in STEAM, *group specification* in MOISE+, and *group definition* in OPERA.
`RoleDef`	Part of a structural specification that represents a *role definition*. Abstracts the concepts of *role* in AGR, MOISE+, ISLANDER and OPERA, and *individual role* in STAEMS.
`RoleRel`	Part of a structural specification that represents a *direct relationship* between two roles: a *source role* referenced by `RoleRel::source`, and a *target role* referenced by `RoleRel::target`. Abstracts the concept of *role constraints* in AGR, *role relations* and *inheritance* in MOISE+, *static separation of duties* (*ssd*) and *subroles* in ISLANDER.
	Auxiliary definitions

`GroupDef`	`::getAllSubGroup() : Set(GroupDef)`, a query that returns the set of all group definitions that, direct or indirectly, are subgroups of a given `GroupDef` via the `GroupDef::subGroup` reference. In OCL:
	context `GroupDef`
	def: `getAllSubGroup( ) : Set(GroupDef)`
	`= subGroup(goal) -> union(`
	`subGroup -> collect (sg | sg.getAllSubGroup()));`

	Contextual Constraints
	(1) All group definition shall reference at least one role definition or one subgroup definition, or both. In OCL: **context** `GroupDef` **inv:** `roleDef -> notEmpty() or subGroup -> notEmpty()`
	(2) No group definition can be, direct or indirectly, a subgroup of itself, i.e., the group definitions shall form an acyclic directed graph. In OCL: **context** `GroupDef` **inv:** `not self.getAllSubGroup() -> includes(self)`

4.2.2. Particularities

Besides the similarities that give rise to the structural specification pattern, the organizational models also differ in some particular points. Four of these points deserve mention.

The first one is related to the definition of subgroups. In STEAM and MOISE+, the definition of subgroups forms a real hierarchy, i.e., an non cyclic graph. On the other hand, in AGR and OPERA, there is no explicit subgroup relationships between group definitions. From structural specification pattern perspective, this fact can be expressed in the following way: In AGR and OPERA, for each group definition `g` (instance of `GroupDef`), the collection of its subgroups is empty, i.e., `g.subGroup = {}`. On the other hand, in STEAM and MOISE+, there exist group definitions `g` such that `g.subGroup` $\neq$ `{}`.

The second point is the notion of *cardinality* that is found in AGR and MOISE+ but not in the other models. Cardinalities can be defined for roles or subgroups. In the case of roles, cardinalities indicate a maximum and a minimum number of agents allowed per role in the context of a group. Regarding cardinalities of groups, they determine how many subgroups of a given type can be created in the context of a group. In AGR, cardinalities are attributes of role and group. In MOISE+, they are attributes of the association between role and group, or group and subgroup. For this reason, and observing that they are not an explicitly feature of the majority of the models, we have chosen not to explicitly represent cardinalities in the structural specification pattern.

As third point, we observe that the abstract notion of *structural relationships between roles* (class `RoleRel`, Table 4) admits diverse concrete subtypes, analogously to what happens with the notion of *goal relationships* (class `RoleRel`, Table 2). In AGR, for instance, there exist two subtypes: *Correspondence* (which states that agents playing one role will automatically play another one) and *dependency* (which rules out the possibility of an agent to play one role if it is not playing another role). In MOISE+, three subtypes: *Links* (which declare the possible relationships of *communication*, *authority* and *acquaintance* between roles), *compatibility* (which determines that two roles can be played at the same time by the same agent) and *inheritance* (which states that one role, besides its own features, also has all the features, like links and compatibilities, of another role). In ISLANDER, two subtypes: The concept of *subroles* (which is similar to the concept of inheritance in MOISE+), and the concept of *static separation of duties* (which means the opposite of the concept of *compatibility* in MOISE+).

In OPERA, there is only one type of binary directed relationship between roles: The *dependency*. Nevertheless, differently from the other organizational models, the concept of *dependency between roles* in OPERA is not properly a structural but rather a functional relationship. In other words, in OPERA, the dependency relationship reflects directly the decomposition of a goal into subgoals, elements of the functional dimension. When one of the subgoals defined in the scope of a role is a goal of another role, then there exist the dependency relationship between the two roles in OPERA. Such idea is different from the structural dependency present in AGR, which indicates that the fact of playing a given role is a prerequisite for playing another role.

The last point that should be mentioned concerns the nature of the group definitions. In all analysed models, except MOISE+, agents playing any roles in the same group may, in principle, exchange messages. Further, in the absence of explicit constraints, such as incompatibilities or dependencies, the agents are free to play the roles they wish in a given group. In MOISE+, we have the opposite situation. If there is no explicitly stated communications links or compatibility relations between roles, the agents are not allowed to exchange messages or play more than one role in the same group. Moreover, in MOISE+, communication links or compatibility relations are not limited to a single group, but can be specified between roles defined in different groups leading to possible inter-group collaborations. In other models, such as AGR, this inter-group collaboration can also be achieved by means explicitly defined correspondence links between two roles in different groups. In this way, an agent playing one of the roles automatically plays the other role and can participate in more than one group at the same time.

4.3. Dialogical Dimension

The dialogical dimension is characterized by concepts to prescribe (or describe) the direct interaction by means of message exchanging that occurs between role playing agents in order to achieve organizational goals. Among the organizational models considered in this work, only ISLANDER and OPERA offer explicit concepts for dialogical modeling (see Table 1). In these two models, the dialogical specifications are written according to approximate conceptual structures, as can be seen in Figure 4.

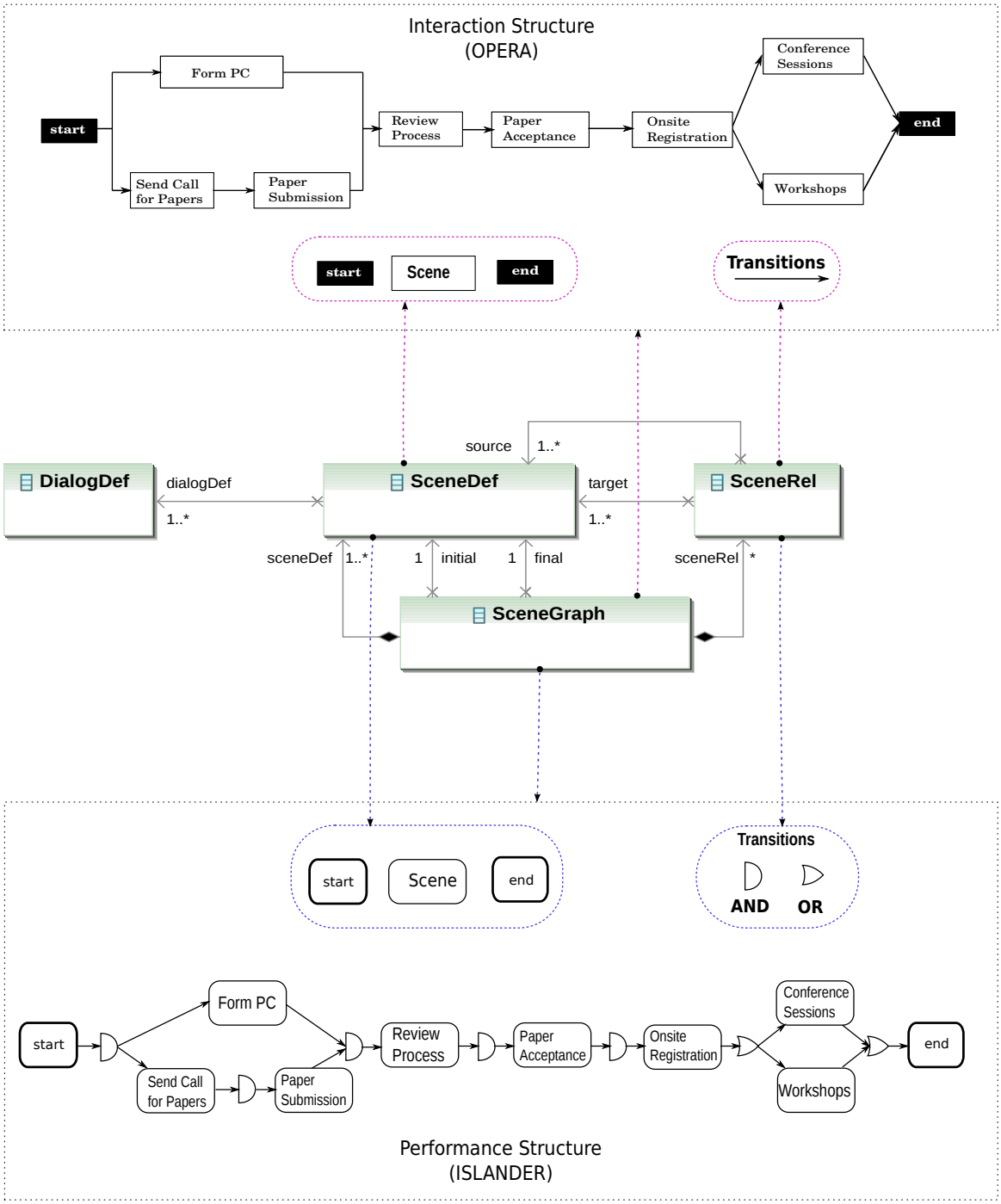


Figure 4. Similar dialogical specifications written in OPERA and ISLANDER for an agent based conference management system (example taken from [27] chapter 3). The class diagram (Ecore metamodel) in the center represents the conceptual pattern identifiable in the two approaches of dialogical modeling. The dotted arrows detail what concepts are captured by what classes of the metamodel.

4.3.1. Hypergraphs of Scenes

On a macro level, both in ISLANDER and in OPERA, the direct interactions by message exchanging are partitioned into *scenes*. Scenes are structured and coordinated by means of directed *hypergraphs* (directed graphs where some edges, called *hyperedges*, can connect any number of sources and target nodes)in which:

- Nodes correspond to the definitions of *scenes*;
- hiperedges represent partial ordering and/or synchronization relationships from many source scenes to many target scenes, giving rise to the concept of *scene transitions*.

In a well formed hypergraph of scenes, there is:

- An *initial scene*, i.e., the node from which agents playing roles have access to the other scenes of the hypergraph. Starting in the initial scene and following the transitions, all the scenes that make up a hypergraph should be achievable;
- a *final scene*, i.e., the node in which the dialogic participation of agents within an organization ends successfully. As the dual of the initial scene, the final scene has to be achievable from any scene in a hypergraph, otherwise the hypergraph of scenes is not well formed.

In ISLANDER, the hypergraphs of scenes are named "performance structures", and in OPERA they are called "interaction structures".

On a micro level, the interactions within each scene are governed by one or more predefined *dialogue scripts*. These scripts correspond to the concepts of *scene protocol* in ISLANDER and *interaction pattern* in OPERA. Dialogue scripts are not detailed in Figure 4. The reason is that the intra-scene (micro level) dialogical specifications have distinct natures both in ISLANDER and in OPERA, as will be discussed in the sequel.

In Figure 4, the conceptual pattern identified in the dialogical specifications of ISLANDER and OPERA is captured by means of an Ecore metamodel. This metamodel is described in details in Tables 5 and 6.

4.3.2. Particularities

In both ISLANDER and OPERA, the dialogical specification consists in a network of scenes in which all possible or desirable episodes of direct interaction within an organization are planned and orchestrated. As mentioned earlier, this common structure takes place at the macro level. This means that the joint activity characteristic of agent organization, under a broad point of view, is ruled by the presented hypergraphs of scenes.

The main difference between ISLANDER and OPERA occurs at the micro level. In other words, restricting the point of view to each particular scene, instead of the network of scenes, the models analyzed have different ways to specify how agents can or should interact.

On the one hand, in ISLANDER, there is the notion of *scene protocol*. In a scene protocol, one represents in detail a communication protocol in which are specified all the involved roles, and the sequencing of all possible message exchanges (*illocution schemes*), in On the other hand, in OPERA, there is the notion of interaction pattern. Unlike scene protocols, an interaction pattern does not determine in detail the exchange of messages in a given scene. Instead, it delimits a partial order between scene states (*landmarks*) towards achieving the objectives related to the scene. Any detailed communication protocol used in a scene should respect the established interaction pattern.

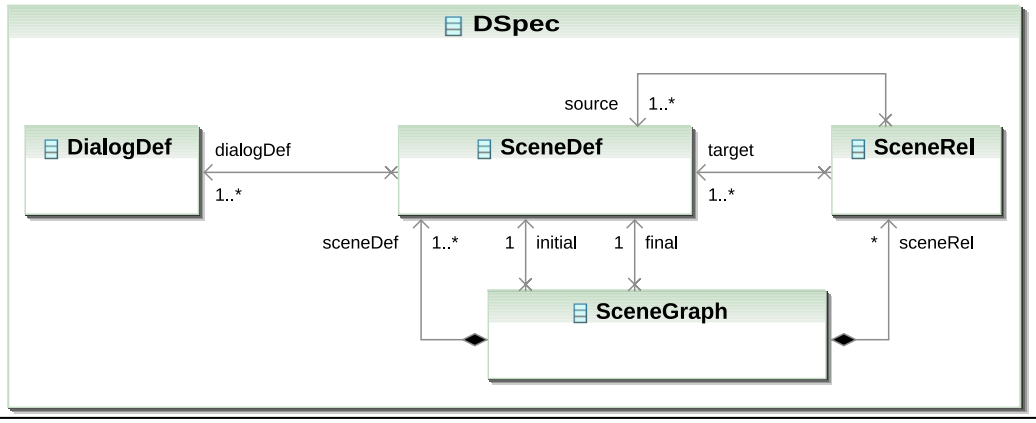
4.4. Normative Dimension

The normative modeling dimension is characterized by the general concept of *norm* (*permissions*, *obligations*, etc.). Norms occur in organizational specifications as a mechanism that interrelates and complements the functional, structural and dialogical specifications. Three organizational models we have analyzed present concepts to create normative specifications. They are: MOISE+, ISLANDER and OPERA (see Table 1).

4.4.1. The Concept of Norm

In the organizational models analyzed, unlike the other dimensions, the normative specifications are not created as graph like structures. Instead, they assume the form of *textual normative expressions*. This general pattern is expressed in Table 7.

Table 5. Dialogical specification pattern: Classes description.



Class	Description
`DSpec`	Represents the concept of *dialogical specification*, i.e., an organizational specification restricted to the dialogical dimension.
`SceneGraph`	Part of a dialogical specification that represents the definition of a *hypergraph of scenes*. `SceneGraph` is characterized by one or more *scene definitions* (referenced by `SceneGraph::sceneDef`), and several *scene relationships* (referenced by `SceneGraph::sceneRel`). Among the scenes, there are one *initial scene* (referenced by `SceneGraph::initial`) and one *final scene* (referenced by `SceneGraph::final`). Abstracts the concepts of *performance structure* in ISLANDER and *interaction structure* in OPERA.
`SceneDef`	Part of a hypergraph of scenes that represents the definition of a *scene*. Each scene definition is characterized by the declaration of a *dialogue script*, via reference `SceneDef::dialogDef`. Abstracts the concepts of *scene definition* in ISLANDER and in OPERA.
`SceneRel`	Part of a hypergraph scene that represents a directed relationship from one or more source scenes (referenced by `SceneRel::source`), to one or more target scenes (referenced by `SceneRel::target`). Abstracts the concepts of *scene transitions* in OPERA and in ISLANDER.
`DialogDef`	Part of of a dialogical specification that represents the definition of a *dialogue script*. Abstracts the concepts of *scene protocol* in ISLANDER, and *interaction pattern* in OPERA.

Table 6. Dialogical specification pattern: Contextual constraints.

Class	Auxiliary Definitions
`SceneGraph`	OCL queries that return immediate predecessors and successors of a scene definition in the context of a hypergraph of scenes. **context** `SceneGraph` **def:** `getSourceSceneDef(sceneDef : SceneDef): Set(SceneDef) = getSceneRelHavingTarget(sceneDef) -> collect(sr | sr.source);` **def:** `getTargetSceneDef(sceneDef : SceneDef): Set(SceneDef) = getSceneRelHavingSource(sceneDef) -> collect(sr | sr.target);` **def:** `getSceneRelHavingTarget(sceneDef : SceneDef): Set(SceneRel) = sceneRel) -> select(sr | sr.target -> includes(sceneDef));` **def:** `getSceneRelHavingSource(sceneDef : SceneDef): Set(SceneRel) = sceneRel) -> select(sr | sr.source -> includes(sceneDef));`

Contextual Constraints

(1) The initial and final scenes are scenes defined in the context of the same `SceneGraph`. In OCL:

context `SceneGraph`
inv: `sceneDef-> includes(initial) and`
inv: `sceneDef-> includes(final)`

(2) In a `SceneGraph`, there should be no relationships arriving at the initial scene or departing from the final scene. In OCL:

inv: `getSceneRelHavingTarget(initial)-> isEmpty() and`
inv: `getSceneRelHavingSource(final)-> isEmpty()`

(3) In a `SceneGraph`, all scene definitions must directly or indirectly be reachable from the initial scene as well as lead to the final scene, via the scene relationships. In OCL:

```
context SceneGraph
inv:    sceneDef-> forAll(sd |
                  getTargetClosure(Set{initial} -> includes(sd) and
                  getSourceClosure(Set{final} -> includes(sd))

def:    getTargetClosure(sceneSet : Set(SceneDef)): Set(SceneDef)
        = let newSceneSet = sceneSet-> collect(scene |
                  getTargetSceneDef(scene) -> including(scene)
                  ) in
        if sceneSet -> includesAll(newSceneSet)
             sceneSet
        else
        getTargetClosure(newSceneSet)
        endif;
def:    getSourceClosure( sceneSet : Set(SceneDef)): Set(SceneDef)
        = let newSceneSet = sceneSet-> collect(scene |
                  getSourceSceneDef(scene) -> including(scene)
                  ) in
        if sceneSet -> includesAll(newSceneSet)
             sceneSet
        else
        getSourceClosure(newSceneSet)
        endif;
```

(4) Every scene relationship must only involve scenes defined within the context of the same `SceneGraph`. In OCL:

```
context SceneGraph
inv:    sceneRel-> forAll(sr |
                  sceneDef -> includesAll(sr.source) and
                  sceneDef -> includesAll(sr.target))
```

4.4.2. Particularities

In ISLANDER, norms are written as logical expressions in accordance with the format:

$(s_1, \gamma_1) \wedge ... \wedge (s_m, \gamma_m) \ \wedge \ e_1 \wedge ... \wedge e_n \ \wedge \neg((s_{m+1}, \gamma_{m+1}) \wedge ... \wedge (s_{m+n}, \gamma_{m+n})) \rightarrow \ obl_1 \wedge ... \wedge obl_p$

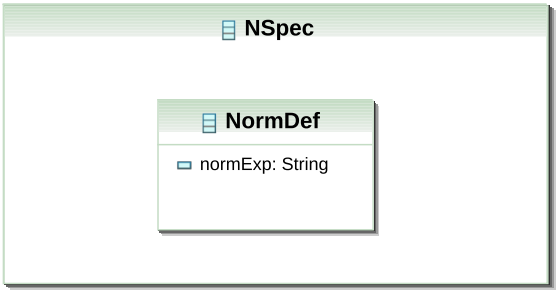
where (s_1, γ_1), ..., (s_{m+n}, γ_{m+n}) are pairs of *scenes* and *illocution schemes*, e_1, ..., e_n are *boolean expressions* over illocution scheme variables, $\neg$ is a *defeasible negation*, and obl_1, ..., obl_p are *obligations*. "The meaning of these rules is that if the illocutions (s_1, γ_1), ..., (s_{m+n}, γ_{m+n}) have been uttered, the expressions e_1, ..., e_n are satisfied and the illocutions (s_{m+1}, γ_{m+1}), ..., (s_{m+n}, γ_{m+n}) have *not* been uttered, the obligations obl_1, ..., obl_p hold" ([48] p. 38).

In OPERA, norms are specified using logical expressions written in a formalism called LCR (*Logic for Contract Representation*). There are three types of norms: *Obligations*, *permissions* and *prohibitions*. The following excerpt from ([27] p. 149) summarizes the syntax for writing these modalities.

`<Norm>::= OBLIGED(<id>,<Norm-Form>)|PERMITTED(<id>,<Norm-Form>)|`
`FORBIDDEN(<id>,<Norm-Form>)`

`OBLIGED(<id >,<Norm-Form>)` represents an obligation of the agent playing the role referenced by `<id>` in achieving the state `<Norm-Form>` described as an LCR formula ([27] chapter 4). Based on the notion of *obligation*, the concepts of *permission* and *prohibition* are defined. A permission `PERMITTED<id>,<Norm-Form>)` is an abbreviation for `¬OBLIGED(<id>,¬<Norm-Form>)`. In turn, a prohibition `FORBIDDEN(<id>,<Norm-Form>)` means the same as `OBLIGED(<id>, ¬<Norm-Form>)`.

Table 7. Normative specification pattern: Class description.



Class	Description
`NSpec`	Represents the concept of *normative specification*, i.e., organizational specifications restricted to the normative dimension.
`NormDef`	Central part of a normative specification. Represents the definition of a *norm*. In general, norm definitions are characterized by a *normative expression* (attribute `NormDef::normExp`) that refers to elements found in the structural, functional and dialogical dimensions. The form and meaning of the normative expression vary considerably in the organizational models analyzed. Abstracts the concepts of *deontic relation* found in MOISE +, and the particular concepts of *norm* found in ISLANDER and OPERA.

Finally, in MOISE+, the general concept of *norm* is translated into the notion of *deontic relations* that link *roles* to *missions*. There are two types of deontic relations, *permissions* and *obligations*:

> "A permission $per(\rho, m, tc)$ states that an agent within the role ρ may be committed to the mission m. Temporal constraints (tc) are established for the permission, that is, they determine a set of time periods when the permission is valid ... An obligation $obl(\rho, m, tc)$ states that an agent within the role ρ is required to commit to the mission m in the time periods determined by tc." ([49] pp. 46–47)

In this case, the normative expressions $per(\rho, m, tc)$ and $obl(\rho, m, tc)$ are less comprehensive than what is found in OPERA and ISLANDER.

4.5. Abstract Organizational Metamodel

All the patterns identified in the organizational modeling dimensions, and previously discussed in Sections 4.1–4.4, can be combined to form an *abstract organizational metamodel*. This abstract metamodel, as shown in Figure 5, characterizes the common conceptual structure of the organizational models analyzed.

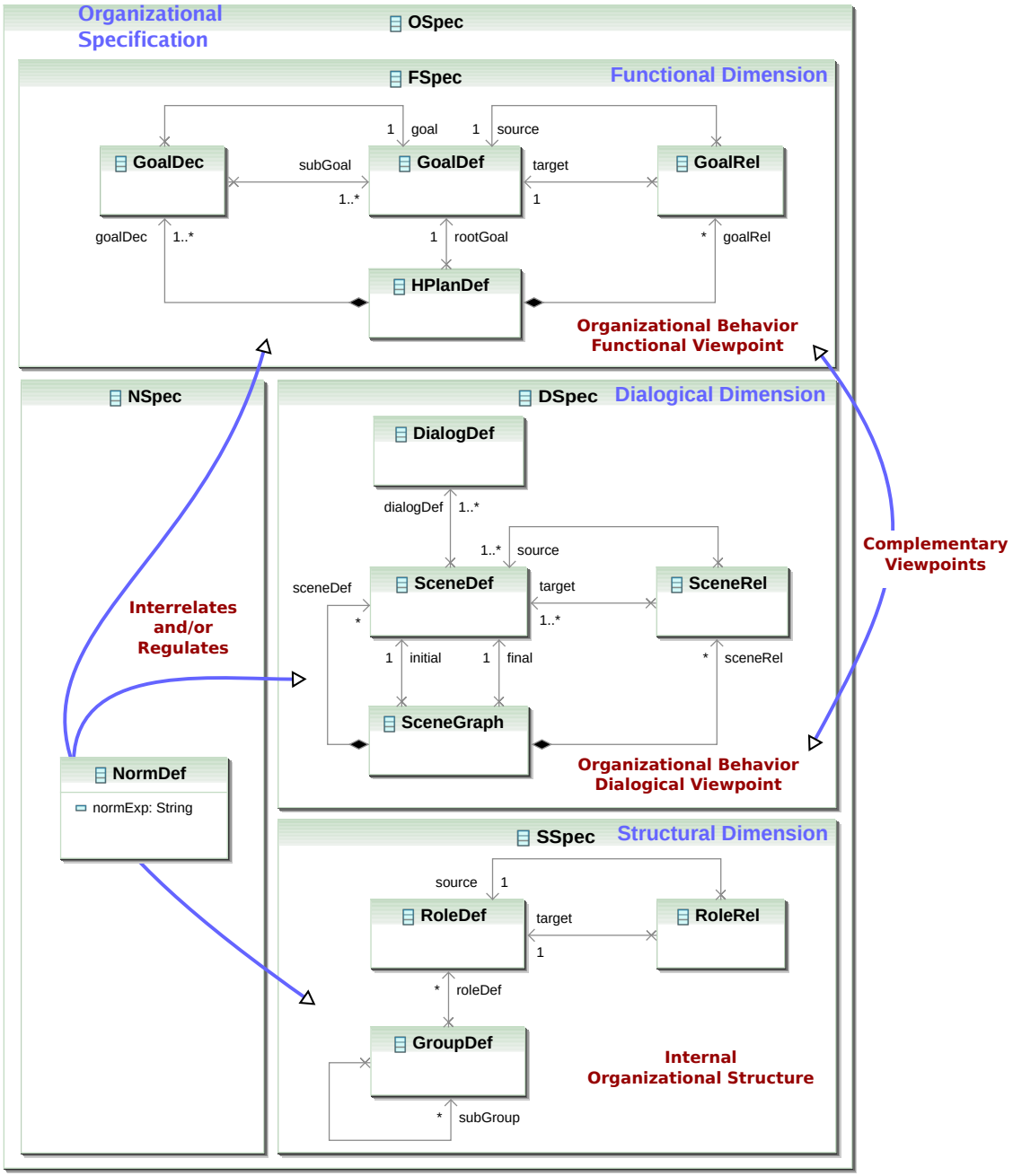


Figure 5. Abstract organizational metamodel.

By means of this abstract organization metamodel, we can see that the normative dimension works as a glue among the three others. It interrelates and/or regulates the organization behavior (be it functional or dialogical) and the organizational internal structure (in the sense of allowing or forcing the association of certain functional and/or dialogical elements with certain structural elements). Last, but not least, it makes it clear that structuring of organizational modeling dimensions greatly helps in making the notions independent and self-contained, while linked via normative bonds.

5. Integration Method Application

In this section we present an application of the integration method described in Section 3, guided by conceptual patterns identified in Section 4. We show how to apply the method to integrate the AGR, STEAM and MOISE+ models. Since a complete description of the integration process involve many details, we will restrict the discussion to the structural dimension.

As shown in Figure 1, we need to perform two iterations of the method to integrate three models. First we merge AGR and STEAM. Then, we merge MOISE+ to the result of the previous iteration.

5.1. First Iteration

The representation of AGR and STEAM as Ecore/OCL metamodels is shown in Figure 6. Below, on the left side, we have the AGR metamodel; on the right side, the STEAM metamodel (both restricted to the structural dimension). Above, mediating the alignment of the metamodels, we see the conceptual pattern of Section 4.2.

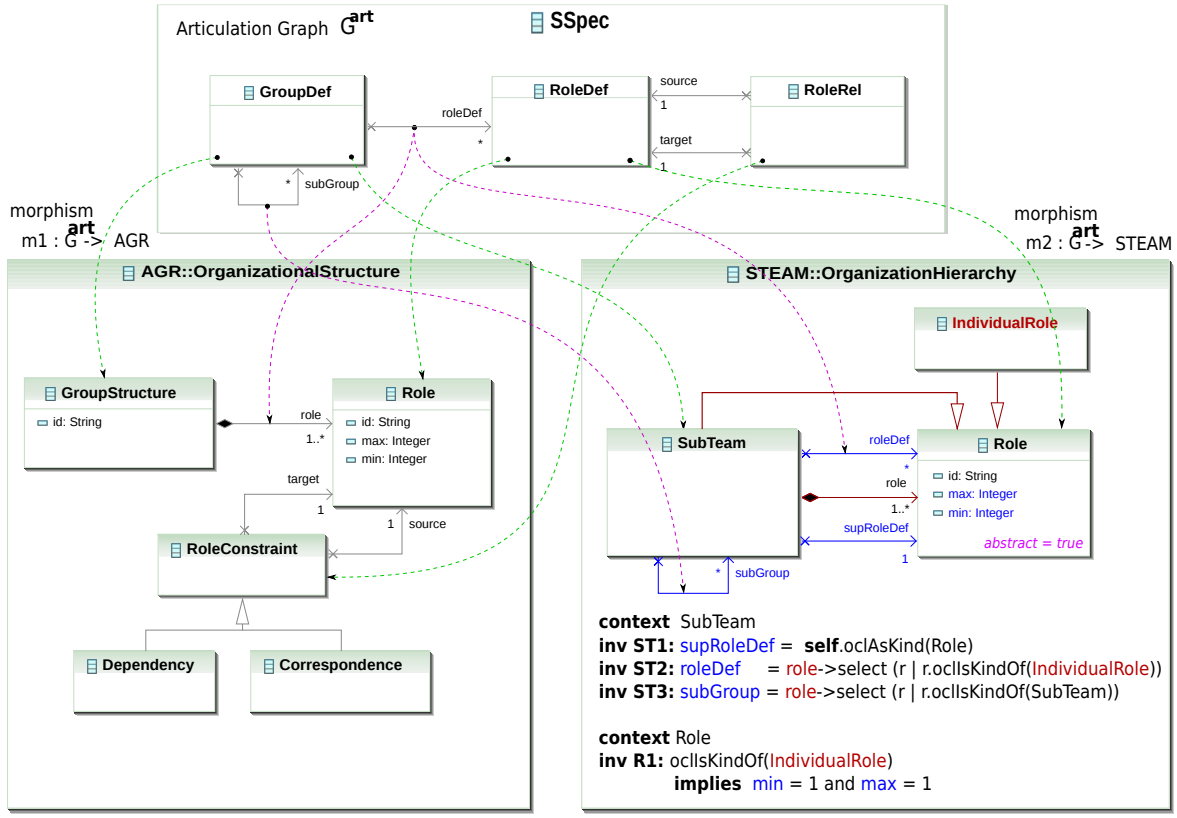


Figure 6. Alignment between AGR and STEAM.

In the alignment, the *organizational structure* of AGR and the *organization hierarchy* of STEAM are identified as similar specifications. The concepts of *group* (AGR) and *subteam* (STEAM) are declared similar concepts, both identified with the general concept of *group definition*. The same happens with the concepts of *role* (AGR) and *role* (STEAM), both identified with the general concept of *role definition*.

Intuitively, when we take into account only the terms used, these basic correspondences between AGR and STEAM are reasonable. However, when we look more closely to the specific relationships among the concepts, it is possible to see that there is a stronger coupling between *subteam* and *role* in STEAM than the one that exists between the correspondent concepts of *group* and *role* in AGR. In STEAM, the notion of *role* is abstract, being materialized both in the specification of activities for groups of agents as a whole and in the specification of activities for individual agents (*individual role*). This notion is represented in the metamodel as an abstract class `STEAM::Role` with two

concrete subtypes `STEAM::SubTeam` and `STEAM::IndividualRole`. As a consequence, every instance of `STEAM::SubTeam` besides corresponding to a *group definition* is also a kind of *role definition*.

This coupling between the concepts of *group* and *role definitions* is absent in AGR and is not foreseen in the structural pattern of Section 4.2. To easy the merging of these different views regarding the nature of the concepts, one possibility is to interpret the generalization relation between `STEAM::SubTeam` and `STEAM::Role` as a composition relationship, similar to the application of the "replace inheritance with delegation" refactoring, as proposed in [50]. When this is done we posit a derived reference from `STEAM::SubTeam` to `STEAM::Role` named `supRoleDef`. This derived reference when used in the place of the original generalization decouples the concepts of *group definition* and *role definition* while permitting to represent the same information in a slightly different way.

In Figure 6 there are two other derived references: `roleDef` and `subGroup`; both extracted from the original reference `role` between `STEAM::SubTeam` and `STEAM::Role`. The rationale for these is to make explicit that, in reality, not only *role definitions* but also *group definitions* can be associated with a *subteam* via the reference `role`. Once these derived references become explicit, we can do a more fine grained matching between the STEAM metamodel and the corresponding abstract concepts of *group* and *role definition*.

Ending the comparison between the metamodels, we note that in AGR the *group definitions* cannot be decomposed into *sub groups* and there is the concept of *role relation* materialized as *role constraints* (*dependencies* and *correspondences*). In STEAM, there is no explicit *role relations* and no explicit *role cardinality*. In the case of *individual role* there is an implicit cardinality of *min=1* and *max=1*.

Finally, concluding the iteration, we merge the metamodels taking into account the correspondences identified. The resultant integrated metamodel $MM^{int}_{\#1}$ is shown in Figure 7, where we retain the terminology of the abstract structural pattern of Section 4.2. See online version for colors. The elements added to the structural pattern from the specific metamodels are depicted in blue. The elements marked in red in the STEAM metamodel (Figure 6) are not included in the integrated metamodel, being replaced by derived references aforementioned. Essentially, $MM^{int}_{\#1}$ is an *almagamated sum* of the AGR and STEAM metamodels (viewed as graphs) modulo the alignment (articulation) between AGR and STEAM, as describe in Section 3.4. For simplicity, the morphisms from $MM^{int}_{\#1}$ to the AGR and STEAM metamodels are omitted.

5.2. Second Iteration

In the second iteration, we integrate the MOISE+ metamodel (structural dimension) to the resulting metamodel $MM^{int}_{\#1}$ obtained in the previous iteration. The MOISE+ metamodel is shown on the left side of Figure 8. On the right side, we have $MM^{int}_{\#1}$ (from Figure 7) augmented with derived classes and relationships. On the middle, there is the articulation graph between MOISE+ and $MM^{int}_{\#1}$ metamodels. Since the articulation graph preserves the class and reference names from $MM^{int}_{\#1}$, for simplicity we have omitted the morphism $m_2 : G^{art} \rightarrow MM^{int}_{\#1}$.

In $MM^{int}_{\#1}$ there are three classes `GroupDef`, `RoleDef` and `RoleRel` which represent the main concepts for the structural specification of agent organizations. In MOISE+, the correspondent classes are `GroupSpecification`, `Role` and `RoleRelation`, respectively. Similar to class $MM^{int}_{\#1}$`::GroupDef`, class `MOISE+::GroupSpecification` represents the definition of a *group* in which it is possible to specify *roles* and *subgroups*. Like $MM^{int}_{\#1}$`::RoleDef`, the class `MOISE+::Role` denotes a *role definition* associated with *group definitions*. Both $MM^{int}_{\#1}$`::RoleRel` and `MOISE+::RoleRelation` characterize *role relationships* from a target to a source *role definition*.

Apart from this basic agreement, there are some particularities regarding how these concepts occur in MOISE+ that leads to an extension of $MM^{int}_{\#1}$. One first particularity is the way in which *group definitions* are linked to *role definitions* and *subgroups*. In the integration of AGR and STEAM, *group definitions* are linked to *role definitions* and to *subgroups* by means of the `roleDef` and `subGroup` references, respectively. In the MOISE+ metamodel the correspondent links are not represented by references but by the classes `GroupRole` and `SubGroup`, respectively.

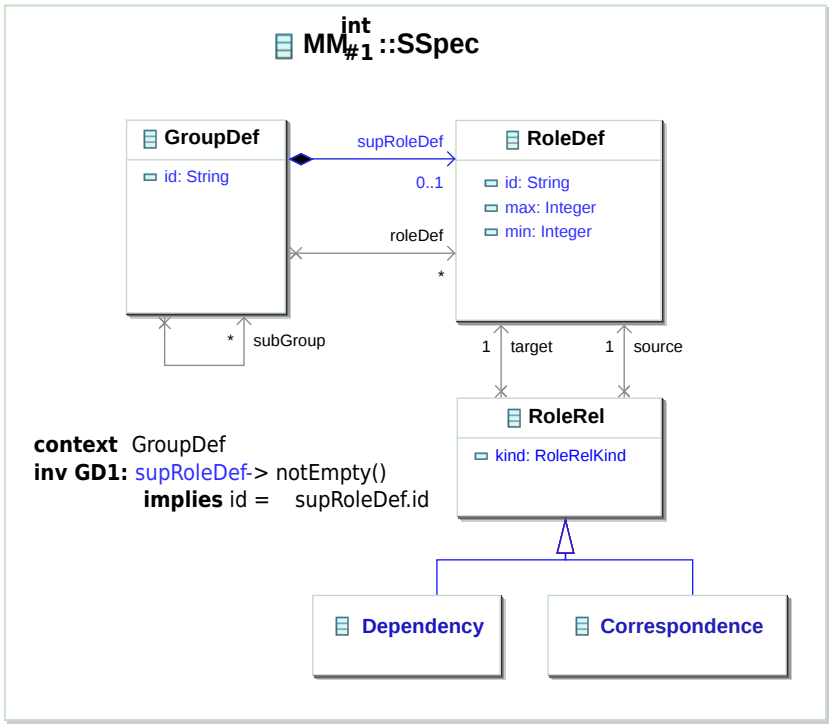


Figure 7. Integrated metamodel $MM^{int}_{\#1}$.

By means of `MOISE+::GroupRole` and `MOISE+::SubGroup`, the same *role definition* or *subgroup* can have different cardinalities, one for each *group definition* in which the definition is referenced. The cardinalities are represented by the attributes `max` (for the maximum number of agents per role, or subgroups in a group) and `min` (for the minimum number).

In $MM^{int}_{\#1}$ this flexibility is not possible as long as the cardinalities are declared directly as attributes of the *role definition*, and not as attributes of the relation between a *group definition* and *role* or *subgroup definition*. In this way, we note that the information about role and group cardinalities of MOISE+ can not always be expressed in the current integration of ARG and STEAM. However, the converse is always possible, as it can be shown by means of the derived classes $MM^{int}_{\#1}$`::RoleRef` and $MM^{int}_{\#1}$`::GroupRef` in the upper right of Figure 8.

The derived class $MM^{int}_{\#1}$`::RoleRef` is the implict correspondent of `MOISE+::GroupRole`. Similar to class `MOISE+::GroupRole`, class $MM^{int}_{\#1}$`::RoleRef` has attributes `max` and `min` and makes reference a single *role definition*. In the context of $MM^{int}_{\#1}$`::GroupDef` the derivation of $MM^{int}_{\#1}$`::RoleRef` is specified by the invariant `GD2` shown in the bottom right of Figure 8. This invariant establishes that for each instance `rd` of $MM^{int}_{\#1}$`::RoleDef` (referenced by `roleDef`), there must exist (be created) an instance `rr` of $MM^{int}_{\#1}$`::RoleRef` that points to `rd` and has the attributes `rr.min = rd.min` and `rr.max = rd.max`.

By its turn, the class $MM^{int}_{\#1}$`::GroupRef` is the derived correspondent of `MOISE+::SubGroup`. In the context of $MM^{int}_{\#1}$`::GroupDef` the derivation of $MM^{int}_{\#1}$`::GroupRef` is specified by the invariant `GD3` shown in the bottom right of Figure 8. The invariant establishes that for each instance `sg` of $MM^{int}_{\#1}$`::GroupDef` (referenced by `subGroup`), there must exist (be created) an instance `gr` of $MM^{int}_{\#1}$`::GroupRef` pointing at `sg` and having the attributes `gr.min = sg.supRoleDef.min` and `gr.max = sg.supRoleDef.max`.

As long as they are more expressive, the classes `RoleRef` and `GroupRef` are used in the articulation graph replacing the references `roleDef` and `subGroup` present in $MM^{int}_{\#1}$. Therefore the replaced references are marked to be left out during the merge step at the end of the iteration. In addition, the attributes `max` and `min` in the class $MM^{int}_{\#1}$`::RoleDef` are marked once the same information is now represented as attributes of `RoleRef` and `GroupRef` in the articulation graph.

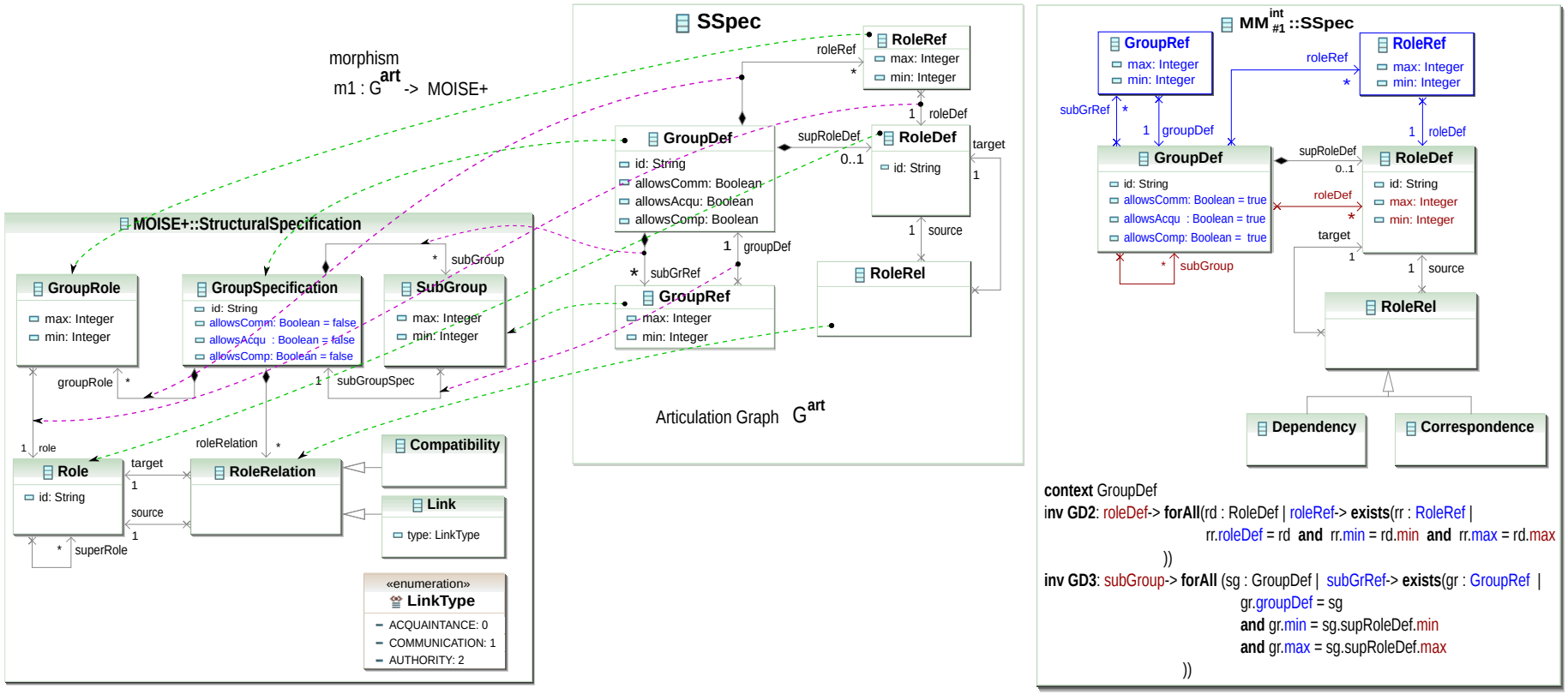


Figure 8. Alignment between MOISE+ and $MM^{int}_{\#1}$.

A second peculiarity of MOISE+ concerns the possibility of defining *links* and *compatibilities* between *roles* as part of a *group specification*. In this regard, there are two observation to be made. Firstly, *links* and *compatibilities* are new kinds of *role relation*, not present in $MM^{int}_{\#1}$. In fact, we observe that the *link* and *compatibility* concepts, in essence, differ from the *dependency* and *correspondence* concepts found in AGR. On the one hand, in MOISE+, a *link* enable the *acquaintance*, *communication*, or *authority* between roles; and a *compatibility* indicates that an agent playing a role can also play another role. On the other hand, in AGR, a *dependency* indicates that an agent only can play a role if it previously commits itself to another one; and a *correspondence* means that to play a role automatically implies to play another role.

The second observation is about the nature of the *group definition* concept behind the notions of *links* and *compatibilities*. By default, a *group definition* in MOISE+ does not enable the *compatibility* or any *link* between roles. In other words, if not stated explicitly, an agent playing a role do not have permission to play another role, or even to interact with agents playing another role, neither in the same nor in other groups. If *compatibilities* and *links* are needed, this must be explicitly specified in the *group definition*.

Conversely, in AGR and STEAM a *group definition* does not implies *a priori* any restriction regarding *compatibility* and *link* among the roles. With the exception of explicit *dependencies* relationships and cardinality restrictions, in AGR and STEAM specifications the agents are free to play the roles they want and are not blocked with respect to interacting with any agent playing some other role.

In the articulation presented in Figure 8, these observations are made explicit by means of the derived attributes `allowsComm`, `allowsAcqu` and `allowsComp` in the context *group definitions*. For `MOISE+::GroupSpecification`, these attributes have the value `false`. This represents respectively the *communication*, *acquaintance* and *compatibility* restrictions existing in MOISE+. On the other side, for $MM^{int}_{\#1}$`::GroupDef` the three attributes assume the value `true` indicating the absence of the respective restrictions in AGR and STEAM.

Despite their opposite nature, we note that AGR and STEAM *group definitions* can be expressed in MOISE+. To this end, one has to explicitly define *communication*, *acquaintance*, and *compatibilities* relationships between all roles in a *group definition*. However, the converse is not always possible without losing information.

Ending the comparison, in MOISE+ there is a third form of *role relationship*: the *inheritance*. In MOISE+ metamodel this relationship is represented by the reference `superRole` involving instances of the class `MOISE+::Role`. In the articulation between MOISE+ and $MM^{int}_{\#1}$, an alternative form of representing inheritance could be as a subclass of `RoleRel`. As the inheritance relation does not have a direct effect on the behavior of the agents as the other role relations, being only a way of simplifying role definitions in MOISE+, we have opted to preserve the representation of this relation as the reference `superRole` rather than defining a new subclass for `RoleRel`.

Finally, concluding the iteration, we merge the metamodels taking into account the identified correspondences. The result is show in Figure 9.

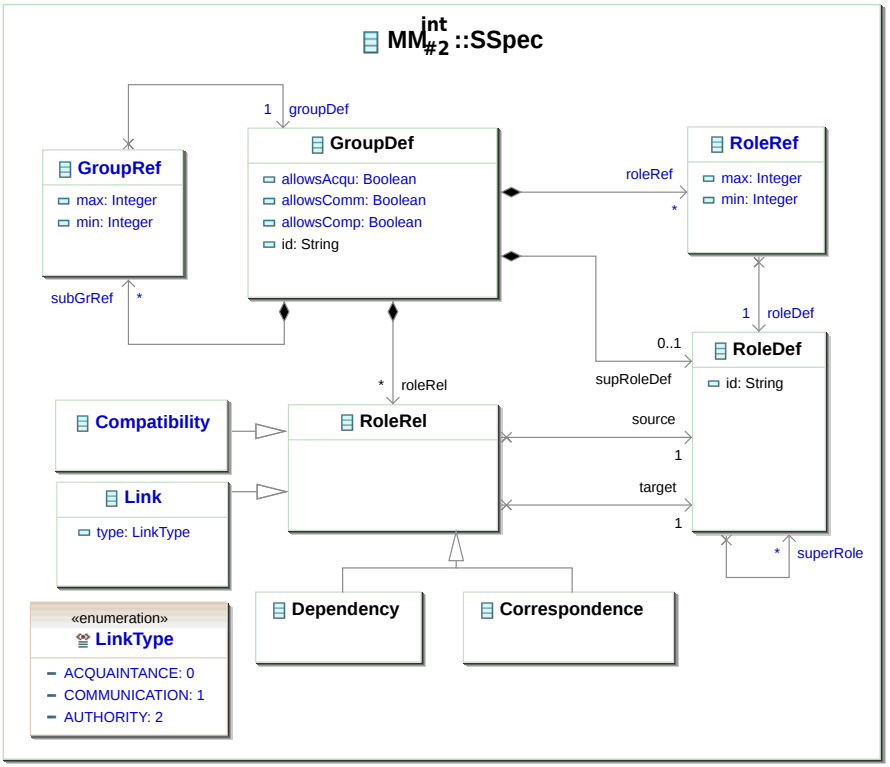


Figure 9. Integrated metamodel $MM^{int}_{\#2}$.

6. Organizational Interoperability Approach

Adopting an organization-centered perspective [4], the engineering of MAS can be described as a process that starts with the creation of an *organizational specification* written in conformance to an *organizational model*. This specification is a prescription of the desired patterns of joint activity that should occur inside the MAS towards some desired purpose. Once the organizational specification is done, it is used as the input to an *organizational infrastructure*. In general, what we mean by organizational infrastructure is some kind of middleware supposed to interpret the specification and reify the *organization* of the MAS outside the agents. In this respect, it should maintain an internal *organizational state* and offer to the agents an interface for accessing and modifying this state. The information maintained in the organizational state contains a list of the members of the organization, what roles they are playing, what groups are active in the organization, among others. Finally, with the organizational infrastructure materializing the desired agent organization, it is time to develop application domain agents (not necessarily by the same designer of the organization specification) that can enter and interact inside it by accessing the available organizational interface.

Regarding *organizational infrastructures* there are several approaches for the engineering of (open) agent organizations [12,13,36,47,51–54]. On the one hand, the availability of a wide range of diverse models and infrastructures has made the development of agent organizations feasible. On the other hand, such a diversity introduced an important new interoperability challenge for agent designers: How to deal with heterogeneous organizational models and infrastructures? Whenever an agent is build to enter some MAS it has to be able to interact with the other participants using a particular agent communication language as well as to understand received messages against a given domain ontology. Besides this, if the MAS was designed as an agent organization, the entering agent has also to be able to access a particular organizational infrastructure and to interpret its underlying organizational model. In this way, the agent design can become tailored to a particular organizational approach.

For instance, suppose that several e-business applications designed as open agent organizations are available on the Internet. In addition, assume that these applications are heterogeneous regarding the organizational technology applied to build them. To put it in more concrete terms, let us suppose

two agents organizations: One built upon the S-MOISE+ [53] organizational middleware, based on the MOISE+ model, and the other by using MADKIT [54] platform, based on ARG model. In this setup (and assuming a shared common agent communication language and domain ontology), the agent designers face the following problem: The native S-MOISE+ agents do not interoperate with the MADKIT platform, and vice-versa. Thus, it is not directly possible, for instance, to write an agent code that enter both e-business agent organizations in the search of products and/or services on behalf of its users. Such fact limits the range of applicability of S-MOISE+ and MADKIT agents which, in turn, limits the idea of open MAS.

As mentioned in the Introduction, four basic approaches can be envisioned for this organizational interoperability problem. One of them is to bridge the interface between the external agents and the agent organization by means of model mapping (Adaptation). By using such mappings it is possible to provide adapted copies of the specification and state of a given organizational model/infrastructure "understood" by the external agents.

In what follows we describe MAORI—a *Model-based Architecture for ORganizational Interoperability* [19]. MAORI is an experimental framework for providing organizational interoperability following the line of adaptation. Its main objective is to show how the integration of organizational models presented in this paper can possibly be used in a solution for the problem of organizational interoperability.

6.1. MAORI Overview

MAORI is structured along three layers, as it may be seen in Figure 10:

- In the bottom, there is the *Model Integration* (M2M) layer—the purpose of this layer is to provide an integrated view and transformations between the organizational models represented as metamodels;
- in the middle, there is the *Organizational Interoperability* (ORI) layer—this layer is formed by components that use the M2M layer to translate and adapt the specification and state of agent organizations from one source organizational infrastructure to one or more target organizational infrastructures;
- in the top, there is the *Organizational Infrastructure* (MAS) layer—this layer corresponds to the available infrastructures for implementing organization-centered MAS.

6.2. Model Integration Layer

M2M layer is composed of *metamodels* and *transformations*. For each organizational model OM_i, there is a corresponding metamodel MM_i. The metamodel MM^{int} is the conceptual integration of all MM_i, as described in Section 5.

The transformations are functions that implement the morphisms between the integrated metamodel MM^{int} and the particular metamodels MM_i (Section 3.4). There are two types of transformations. One type is $\mathbf{transf}(from : MM_i) : MM^{int}$, which converts from MM_i to MM^{int}. The other type is $\mathbf{transf}(from : MM^{int}) : MM_i$, which converts from MM^{int} to MM_i. In this way, M2M main functionality is to provide transformations that can be combined to translate specifications and states between organizational models/infrastructures.

6.3. Organizational Interoperability Layer

ORI layer works as an extension of organizational infrastructures. In order to enable heterogeneous agents in the same organization, ORI adds two basic components to the organizational infrastructures: *providers* and *adapters*.

Providers are responsible for exporting the organizational specification/state of agent organizations. In this case, to export means to use $\mathbf{transf}(from : MM_i) : MM^{int}$ to convert the specification/state from from a source MM_i to the integrated metamodel MM^{int}. Adapters are responsible for importing the organizational specification/state that was exported by a provider. The import is done by using $\mathbf{transf}(from : MM^{int}) : MM_i$.

Imagine a scenario where an agent functions on a given organizational infrastructure and consider an agent organization implemented on a different organizational infrastructure. If the agent wants to participate in the organization, an adapter has to be instantiated in the organizational infrastructure of the entering agent. Initially, the responsibility of the adapter is to locate the appropriate provider, establish a connection with it, ask for the organizational specification/state and finally translate this specification/state to the target organizational infrastructure of the entering agent. In this way, for each heterogeneous agent organization there will be an organizational provider. Connected to this provider, there will be several organizational adapters, one for each organizational infrastructure in which there could be external heterogeneous agents.

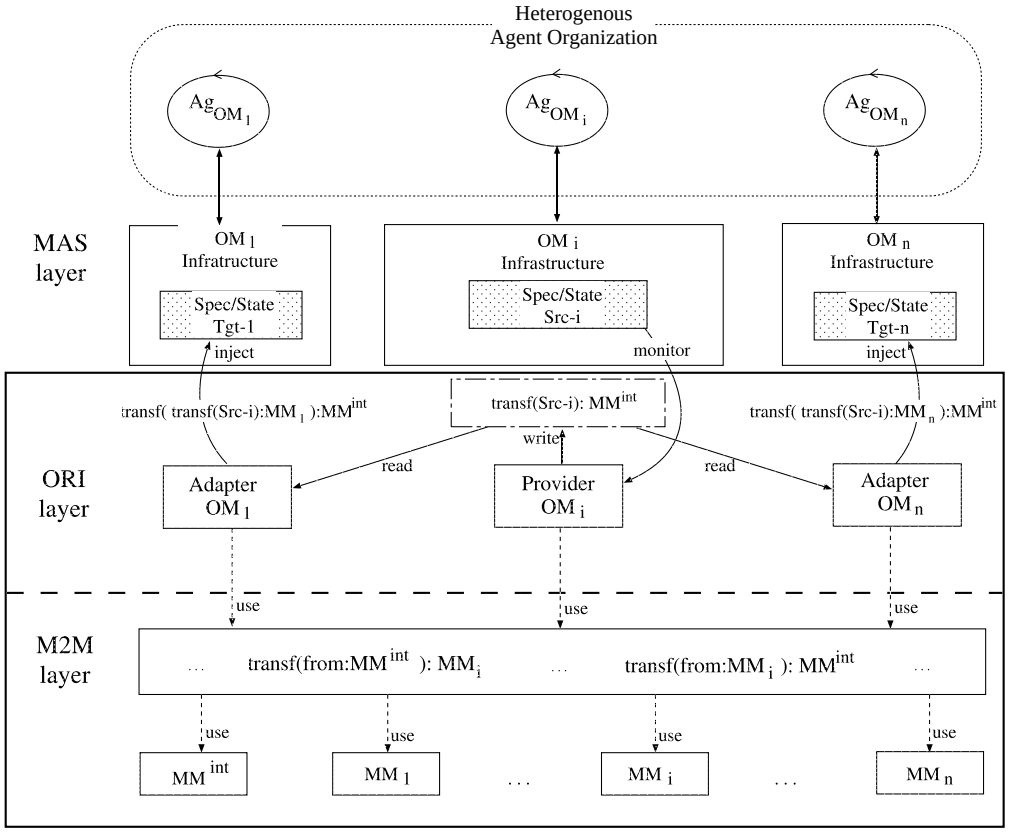


Figure 10. MAORI framework (redesigned from [19]).

6.4. MAORI Implementation

MAORI was implemented in the Java programming language. The metamodels in M2M layer were coded using the Eclipse Modeling Framework (EMF) [55]. Regarding the transformations, there were first prototyped in the Atlas Transformation Language (ATL) [56] and then ported to Java for performance reasons. An implementation was developed as a proof of concept of the ORI layer considering the MADKIT and S-MOISE+ organizational infrastructures.

To evaluate MAORI, some agent organizations were developed. One is the example of a group of agents that wants to write a paper and use for this purpose an explicit organization to help them to collaborate. The organization consists in a group composed of: One agent in the role of *coordinator* (who controls the process and writes the introduction and conclusion of the paper), one to five agents in the role of *collaborator* (who writes the paper sections) and one agent in the role of *librarian* (who compiles the bibliography). Taking this simple example, some experiments were performed. One of them considered an organization composed of five agents—one coordinator (Eric), three collaborators (Greg, Joel and Mark) and one librarian (Carol). Initially the organization was started in the MADKIT platform. In addition, in MADKIT, the agents Eric and Carol were started and,

after that, in the S-MOISE+ platform an organizational adapter was started to import the organization state. The three remaining agents (Greg, Joel and Carol) are started in S-MOISE+. They perceived and enter the organization by requesting the role of collaborator. At this point, the interaction begins: The agents in S-MOISE+ are now members of an organization running in MADKIT. More details about MAORI may be found in [19].

7. Related Work

The proposition of abstract structures, such as patterns, and integrated metamodel to enable interoperability among organization-centered multiagent systems is somehow new. Therefore, related work must be considered in several areas from business to services, including multiagent systems. In the following we contextualize our work within this broad scenario.

In the multiagent systems area, Pechoucek and Marik [57] adopted a model-driven approach to propose a general metamodel for developing multiagent systems. The metamodel is defined as a Platform Independent Model (PIM) considering the MDA abstraction levels definition [18]. In order to identify a unified metamodel to support the development of agent-based systems, they considered seven views: Multiagent view; Agent view; Behavioural view; Organization view; Role view; Interaction view and Environment view. They adopted the top down approach to develop the metamodels that represent each of the views aforementioned after analysing some existing agent-oriented modeling languages, methodologies and programming languages. Moreover, it was conceived to be used independently of the agent-oriented methodology, modeling and programming languages. Nevertheless, their main purpose was to support the development of MAS using a model-driven approach than providing means for interoperability among MAS. Our work presents some similarities with theirs since the proposed integrated metamodel could be used independently of the Organization Model adopted to design and implement an organization-centered MAS. In addition, the abstract structures where defined based on organization-centered multiagent systems dimensions, in a similar approach of theirs when stating their metamodel concerning some views. Nevertheless, although using a model driven approach, we adopted it to define the way integration occurs using the bottom-up approach to define the integrated metamodel, based on existing Agent Organization Models and their underlying metamodels. By doing that we foster interoperability to organization-centered MAS during design time, by providing means of transforming the design of a MAS with an underlying organization model into another one, or during execution time, as presented in Section 6.

Muramatsu and colleagues [58] provided organizational interoperability by using organizational artifacts within the environment where the MAS is situated. They adopted a normative language to describe the organizational structure in artifacts. In this sense, their work is similar to ours while adopting a common language (in our case a common metamodel) to describe several organizational models.

Isern and colleagues [59] classified organizational structures according to organizational paradigms, such as (i) hierarchy, (ii) holarchy, (iii) coalition, (iv) team, (v) congregation, (vi) society, (vii) federation and (viii) market, to support the design of MAS using existing agent-oriented methodologies and organizational models. The main purpose of their work is to provide information for MAS developers that would like to adopt an organization approach to develop MAS and did not know what Organization Model or agent-oriented methodology to choose. Their work is related to ours in the sense they adopted metamodels' characteristics of existing organizational models and existing agent-oriented methodologies to classify such organizational structures as patterns.

Karaenke et al. [60] proposed an inter-organizational interoperability architecture based on multiagent systems, web services and semantic web technologies. In their work, the MAS did not present an underlying organizational model and agents adopt the "head body" paradigm to include web services technologies to provide interoperability among enterprise information systems. Therefore, interoperability is focused on system-to-system communication using web services technologies, which limits the kind of systems that may participate in such communication. Our proposition is

broader in the sense that it provides a solution for interoperability among open organization-centered MAS independently of their underlying organizational Model.

A template description for agent-oriented patterns was given by Oluyomi and colleagues [61]. Based on a classification scheme, they organized agent technology concepts into categories and then identified agent-oriented pattern description templates for each category. Eight agent-oriented pattern templates were described to support the modeling of multiagent systems. Examples of the conformity between the proposed templates and their adoption during the design phase of existing agent-oriented methodologies were provided. Comparing with our proposition, their work adopts a similar approach when considering categories (in our case we adopted dimensions) to guide the patterns definition. In addition, their work is situated in the model level instead of the metamodel level as ours, since their objective is to improve communication among AOSE developers. Nevertheless, the adoption of an integrated metamodel obtained via model driven transformations combined with organizational dimensions give to our patterns high level of formality whenever compared with theirs.

Chella et al. [62] defined agent-oriented patterns to develop multiagent system to support robot programming. The proposed patterns were created based on an existing layered architecture for programming robots [63]. Patterns are described considering three aspects: The problem description; the definition of the solution in terms of MAS models and the description of the solution in terms of implementation. They just define some patterns for an specific domain based on the templates proposed by [61]. The only relation between their work and our is that pattern definition is based on some criteria that can be classified as dimension or category.

Organizational interoperability and integration issues are not new concerns for the administrative practice and research, specially after the wide acceptance and use of Information and Commmunication Technologies in their business models [64]. Several frameworks to provide organization interoperability were defined and even in this domain some dimensions were considered to define such frameworks.

8. Conclusions and Future Work

The research reported in this work has consisted in the use Model Driven Engineering techniques to address the *organizational interoperability problem*: How can we provide means for a set of agents, immersed in a common environment, to evolve, reason, decide and interact with each other based on organizational concepts, since their organizational models may differ? In order to achieve this goal, we have proposed an abstract and integrated view of the main concepts that have been used to specify agent organizations, based on the analysis of several organizational models present in the literature. In this model, we captured the recurring modeling concepts, that were coherently combined into an abstract conceptual structure. We have then presented an adaptation-based solution for the organizational interoperability problem, when we have defined the mappings between different organizational models, by using this abstract conceptual structure. We have built our abstract conceptual structure based on six organizational models (STEAM, MOISE+, AGR, OPERA, TEAMS, ISLANDER), presented in Section 2. For brevity, in Section 5 we illustrated the application of our integration method using three of these models (MOISE+, AGR, STEAM), and concerning exclusively the structural dimension.

A first extension of this work would be to build an integrated metamodel that could cope with OPERA and TEAMS. Moreover, we could evaluate how the other organizational models mentioned in Section 2.2.7 would affect our abstract conceptual structure. Concerning the MAORI framework, described in Section 6, we have tested its use by interoperating two organizational infrastructures, S-MOISE+ and MADKIT. A second extension of this work would be to test the franework with other organizational infrastructures, like AMELI [13] and ORA4MAS [52]. Finally, we would like to test our model driven approach to solve other MAS interoperability problems, like the ones mentioned in Section 1.

Author Contributions: Conceptualization, L.R.C, J.S.S. and O.B.; Formalization, implementation and experiments, L.R.C and A.A.F.B.; Supervision, J.S.S and O.B.; Writing and proofreading the paper, all authors.

Funding: "This research was partially funded by FAPEMA grant number 127/04 and CAPES, Brazil, Grant 1511/06-8". Jaime Sichman was partially supported by CNPq and FAPESP, Brazil. Anarosa Alves Franco Brandão was partially supported by grant #010/20620-5 and #014/03297-7, São Paulo Research Foundation (FAPESP), Brazil.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Castelfranchi, C. Engineering social order. In Proceedings of the Engineering Societies in the Agents World: First International Workshop, ESAW 2000, Berlin, Germany, 21 August 2000; Revised Papers; Omicini, A., Tolksdorf, R., Zambonelli, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1972, pp. 1–18.
2. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* **2007**, *95*, 215–233. [CrossRef]
3. Shang, Y. Hybrid consensus for averager–copier–voter networks with non-rational agents. *Chaos Solitons Fract.* **2018**, *110*, 244–251. [CrossRef]
4. Boissier, O.; Hübner, J.F.; Sichman, J.S. Organisational oriented programming from closed to open organizations. In Proceedings of the 7th International Workshop, Engineering Societies in the Agents World VII, ESAW 2006, Dublin, Ireland, 6–8 September 2006; pp. 86–105.
5. Ferber, J.; Gutknecht, O.; Michel, F. From agents to organizations: An organizational view of multi-agent systems. In Proceedings of the Agent-Oriented Software Engineering IV: 4th International Workshop, AOSE 2003, Melbourne, Australia, 15 July 2003.
6. Gasser, L. Perspectives on organizations in multi-agent systems. In *Multi-Agent Systems and Applications: 9th ECCAI Advanced Course, ACAI 2001, and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, 2–13 July 2001; Selected Tutorial Papers*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2086, pp. 1–16.
7. Zambonelli, F.; Jennings, N.R.; Wooldridge, M. Organisational abstractions for the analysis and design of multi-agent systems. In Proceedings of the Agent-Oriented Software Engineering: First International Workshop, AOSE 2000, Limerick, Ireland, 10 June 2000; Revised Papers; Ciancarini, P., Wooldridge, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 1957, pp. 235–251.
8. Scott, W.R. *Organizations: Rational, Natural and Open Systems*, 4th ed.; Prentice Hall: Upper Saddle River, NJ, USA, 1998.
9. Kleppe, A. *Software Language Engineering: Creating Domain-Specific Languages Using Metamodels*; Addison Wesley: Reading, UK, 2008.
10. Poslad, S.; Charlton, P. *Standardizing Agent Interoperability: The FIPA Approach*; Springer: Berlin, Germany, 2001; Number 2086 in LNAI, pp. 98–117.
11. Magnin, L.; Pham, V.T.; Dury, A.; Besson, N.; Thiefaine, A. Our guest agents are welcome to your agent platforms. In Proceedings of the ACM Symposium on Applied Computing 2002, Madrid, Spain, 11–14 March 2002; pp. 107–114.
12. Tambe, M.; Pynadath, D.V. Towards heterogeneous agent teams. In *Multi-Agent Systems and Applications, Proceedings of the 9th ECCAI Advanced Course, ACAI 2001, and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001, Prague, Czech Republic, 2–13 July 2001; Selected Tutorial Papers*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2086, pp. 187–210.
13. Esteva, M.; Rosell, B.; Rodríguez-Aguilar, J.A.; Arcos, J.L. AMELI: An agent-based middleware for electronic institutions. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), New York, NY, USA, 19–23 July 2004; Volume I, pp. 236–243.
14. Nardin, L.G.; Brandão, A.A.F.; Sichman, J.S. Experiments on semantic interoperability of agent reputation models using the SOARI architecture. *Eng. Appl. Artif. Intell.* **2011**, *24*, 1461–1471. [CrossRef]
15. Giampapa, J.A.; Paolucci, M.; Sycara, K. Agent Interoperation Across Multiagent System Boundaries. In Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000), Barcelona, Spain, 3–7 June 2000; pp. 179–186.
16. Wegner, P. Interoperability. *ACM Comput. Surv.* **1996**, *28*, 285–287. [CrossRef]

17. Coutinho, L.; Sichman, J.; Boissier, O. Modelling Dimensions for Agent Organizations. In *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*; Dignum, V., Ed.; IGI Global: Hershey, PA, USA, 2009; Chapter II, pp. 18–50.
18. Schmidt, D.C. Model-driven engineering. *IEEE Comput.* **2006**, *36*, 25–31. [CrossRef]
19. Coutinho, L.R.; Brandão, A.A.F.; Sichman, J.S.; Hübner, J.F.; Boissier, O. A Model-Based Architecture for Organizational Interoperability in Open Multiagent Systems. In *Coordination, Organizations, Institutions and Norms in Agent Systems V*; Padget, J., Artikis, A., Vasconcelos, W., Stathis, K., Silva, V., Matson, E., Polleres, A., Eds.; Springer: Heidelberg, Germany, 2010; Volume 6069, pp. 102–113, doi:10.1007/978-3-642-14962-7_7.
20. Yourdon, E. *Análise Estruturada Moderna*; Campus: Rio de Janeiro, Brazil, 1992.
21. Booch, G.; Rumbaugh, J.; Jacobson, I. *The Unified Modeling Language*; Addison Wesley: Reading, UK, 1999.
22. Ackoff, R.L. *Ackoff's Best, His Classic Writings on Management*; John Wiley & Sons: New York, NY, USA, 1999.
23. Decker, K.; Lesser, V. Task environment centered design of organizations. In *AAAI Spring Symposium on Computational Organization Design*; AAAI: Menlo Park, CA, USA, 1994; pp. 32–38.
24. Tambe, M.; Adibi, J.; Al-Onaizan, Y.; Erdem, A.; Kaminka, G.A.; Marsella, S.C.; Muslea, I. Building agent teams using an explicit teamwork model and learning. *Artif. Intell.* **1999**, *110*, 215–239. [CrossRef]
25. Hübner, J.F.; Sichman, J.S.; Boissier, O. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In Proceedings of the Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002 Porto de Galinhas/Recife, Brazil, 11–14 November 2002; Bittencourt, G., Ramalho, G.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2507, pp. 118–128.
26. Esteva, M.; Padget, J.; Sierra, C. Formalizing a language for institutions and norms. In Proceedings of the Intelligent Agents VIII: 8th International Workshop, ATAL 2001, Seattle, WA, USA, 1–3 August 2002; pp. 348–366.
27. Dignum, V. A Model for Organizational Interaction: Based on Agents, Founded in Logic. Ph.D. Thesis, Utrecht University, Utrecht, The Netherlands, 2004.
28. Ferber, J.; Gutknecht, O. A meta-model for the analysis and design of organizations in multi-agent systems. In Proceedings of the Third International Conference on Multi Agent Systems, Paris, France, 3–7 July 1998; pp. 128–135.
29. Horling, B.; Lesser, V. Quantitative organizational models for large-scale agent systems. In Proceedings of the International Workshop on Massively Multi-Agent Systems, Kyoto, Japan, 10–11 December 2004; pp. 297–312.
30. Ferber, J.; Michel, F.; Baez, J. AGRE: Integrating environments with organizations. In Proceedings of the Environments for Multi-Agent Systems: First International Workshop, E4MAS 2004, New York, NY, USA, 19 July 2004; Revised Selected Papers; Weyns, D., Parunak, H.V.D., Michel, F., Eds.; Springer: Berlin, Germany, 2005; Volume 3374, pp. 48–56. [CrossRef]
31. Gâteau, B.; Boissier, O.; Khadraoui, D.; Dubois, E. *MoiseInst: An Organizational Model for Specifying Rights and Duties of Autonomous Agents*; EUMAS, Gleizes, M.P., Kaminka, G.A., Nowé, A., Ossowski, S., Tuyls, K., Verbeeck, K., Eds.; Koninklijke Vlaamse Academie van Belie voor Wetenschappen en Kunsten: Brussels, Belgium, 2005; pp. 484–485.
32. Dignum, V.; Vázquez-Salceda, J.; Dignum, F. OMNI: Introducing social structure, norms and ontologies into agent organizations. In Proceedings of the Programming Multi-Agent Systems: Second International Workshop ProMAS 2004, New York, NY, USA, 20 July 2004; Selected Revised and Invited Papers; Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3346, pp. 181–198.
33. Vázquez-Salceda, J.; Dignum, F. Modelling electronic organizations. In Proceedings of the Multi-Agent Systems and Applications III: 3rd International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague, Czech Republic, 16–18 June 2003; Marík, V., Müller, J., Pechoucek, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2691, pp. 584–593.
34. Silva, V.; Choren, R.; Lucena, C. A UML based approach for modeling and implementing multi-agent systems. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), New York, NY, USA, 19–23 July 2004; pp. 914–921.

35. Silva, V.; Garcia, A.; Brandão, A.; Chavez, C.; Lucena, C.; Alencar, P. Taming agents and objects in software engineering. In *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications*; Garcia, A., Lucena, C., Zambonelli, F., Omicini, A., Castro, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2603, pp. 1–26.
36. Weyns, D.; Haesevoets, R.; Helleboogh, A. The MACODO Organization Model for Context-driven Dynamic Agent Organizations. *ACM Trans. Auton. Adapt. Syst.* **2010**, *5*, 16:1–16:29. [CrossRef]
37. Zambonelli, F.; Jennings, N.R.; Wooldridge, M. Developing multiagent systems: The Gaia methodology. *ACM Trans. Softw. Eng. Methodol.* **2003**, *12*, 317–370. [CrossRef]
38. Bresciani, P.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J.; Perini, A. Tropos: An agent-oriented software development methodology. *J. AAMAS* **2004**, *8*, 203–236. [CrossRef]
39. Jouault, F.; Bézivin, J. KM3: A DSL for Metamodel Specification. In *Formal Methods for Open Object-Based Distributed Systems*; Gorrieri, R., Wehrheim, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4037, pp. 171–185, doi:10.1007/11768869_14.
40. OMG. *Meta-Object Facility (2.0) Core Specification*; OMG Document Formal/2006-01-01. Available online: https://www.omg.org/spec/MOF/2.0/About-MOF/ (accessed on 10 June 2019).
41. OMG. *Object Constraint Language*, Version 2.0, 2006. Available online: https://www.omg.org/spec/OCL/2.0/About-OCL/ (accessed on 10 June 2019).
42. Clark, T.; Sammut, P.; Willans, J. *Applied Metamodelling—A Foundation for Language Driven Development*; CETEVA: London, UK, 2008.
43. Steinberg, D.; Budinsky, F.; Paternostro, M.; Merks, E. *EMF: Eclipse Modeling Framework*; Addison-Wesley: Reading, UK, 2008.
44. Pottinger, R.; Bernstein, P.A. Merging models based on given correspondences. In Proceedings of the 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, 9–12 September 2003; Freytag, J.C., Lockemann, P.C., Abiteboul, S., Carey, M.J., Selinger, P.G., Heuer, A., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 2003; pp. 826–873.
45. Sabetzadeh, M.; Easterbrook, S. View Merging in the Presence of Incompleteness and Inconsistency. *Requir. Eng. J.* **2006**, *11*, 174–193. [CrossRef]
46. Diskin, Z.; Dingel, J. A metamodel-independent framework for model transformation: Towards generic model management patterns in reverse engineering. In Proceedings of the 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies, Genoa, Italy, 1–6 October 2006; Favre, J.M., Gasevic, D., Laemmel, R., Winter, A., Eds.; Springer: Berlin, Germany, 2007; Volume 4364, pp. 52–55.
47. Lesser, V.; Decker, K.; Wagner, T.; Carver, N.; Garvey, A.; Horling, B.; Neiman, D.; Podorozhny, P.; Prasad, N.N.; Raja, A.; et al. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Auton. Agents Multi-Agent Syst.* **2004**, *9*, 87–143. [CrossRef]
48. Esteva, M. Electronic Institutions: From Specification to Development. Ph.D. Thesis, Institut d'Investigació en Intel.ligència Artificial, Bellaterra, Catalonia, Spain, 2003.
49. Hübner, J.F. Um modelo de reorganização de sistemas multiagentes. Tese (doutorado), Escola Politécnica da Universidade de São Paulo, 2003. Available online: http://www.teses.usp.br/teses/disponiveis/3/3141/tde-17052004-151854/ (accessed on 10 June 2019).
50. Fowler, M. *Refactoring: Improving the Design of Existing Code*; Addison Wesley: Reading, UK, 1999.
51. Boissier, O.; Bordini, R.H.; Hübner, J.F.; Ricci, A.; Santi, A. Multi-agent oriented programming with JaCaMo. *Sci. Comput. Program.* **2013**, *78*, 747–761. [CrossRef]
52. Kitio, R.; Boissier, O.; Hubner, J.F.; Ricci, A. Organisational artifacts and agents for open multi-agent organisations: Giving the power back to the agents. In Proceedings of the International Workshops COIN@AAMAS 2007, Durham, UK, 3–4 September 2007; Springer: Berlin, Germany, 2008; Volume 4870, pp. 171–186.
53. Hübner, J.F.; Sichman, J.S.; Boissier, O. S-Moise+: A middleware for developing organised multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems*; Boissier, O., Padget, J., Dignum, V., Lindemann, G., Matson, E., Ossowski, S., Sichman, J.S., Vázquez-Salceda, J., Eds.; Springer: Heidelberg, 2006; LNCS (LNAI), Volume 3913, pp. 64–77.
54. Gutknecht, O.; Ferber, J. The MADKIT agent platform architecture. In *International Workshop on Infrastructure for Multi-Agent Systems: Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*; Springer: London, UK, 2000; Volume 1887, pp. 48–55.

55. Eclipse Team. Eclipse Modeling Framework (EMF). Available online: http://www.eclipse.org/emf/ (accessed on 10 June 2019).
56. Jouault, F.; Kurtev, I. Transforming models with ATL. In Proceedings of the Satellite Events at the MoDELS 2005 Conference, Montego Bay, Jamaica, 2–7 October 2005; Bruel, J.M., Ed.; Springer: Berlin, Germany, 2006; Volume 3844, pp. 128–138.
57. Hahn, C.; Madrigal-Mora, C.; Fischer, K. A platform-independent metamodel for multiagent systems. *Auton. Agents Multi-Agent Syst.* **2009**, *18*, 239–266. [CrossRef]
58. Muramatsu, F.T.; Vitorello, T.M.; Brandão, A.A.F. Towards Organizational Interoperability through the Environment. In *Agent Environments for Multi-Agent Systems IV*; Weyns, D., Michel, F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 209–231.
59. Isern, D.; Sanchez, D.; Moreno, A. Organizational structures supported by agent-oriented methodologies. *J. Syst. Softw.* **2011**, *84*, 169–184. [CrossRef]
60. Karaenke, P.; Schuele, M.; Micsik, A.; Kipp, A. Inter-organizational Interoperability through Integration of Multiagent, Web Service, and Semantic Web Technologies. In *Agent-Based Technologies and Applications for Enterprise Interoperability, (ATOP 2009, ATOP 2010)*; Fischer K., Müller J.P., Levy R., Eds; Lecture Notes in Business Information Processing, Volume 98; Springer: Berlin/Heidelberg, Germany, 2012, pp. 55–75.
61. Oluyomi, A.; Karunasekera, S.; Sterling, L. Description templates for agent-oriented patterns. *J. Syst. Softw.* **2008**, *81*, 20–36. [CrossRef]
62. Chella, A.; Cossentino, M.; Gaglio, S.; Sabatucci, L.; Seidita, V. Agent-oriented software patterns for rapid and affordable robot programming. *J. Syst. Softw.* **2010**, *83*, 557–573. [CrossRef]
63. Chella, A.; Cossentino, M.; Gaglio, S.; Sabatucci, L.; Seidita, V. An architecture for autonomous agents exploiting conceptual representations. *Robot. Auton. Syst.* **1998**, *25*, 231–240. [CrossRef]
64. Kubicek, H.; Cimander, R. Three dimensions of organizational interoperability. *Eur. J. Epractice* **2009**, *6*, 3–14.