Formal framework to support organizational design

Catholijn M. Jonker, Viara Popova, Alexei Sharpaniskyh, Jan Treur, Pinar Yolum

Abstract

Organizational design is an important topic in the literature on organizations. Usually the design principles are addressed informally in this literature. This paper makes a first attempt to formally introduce design operators to formalize the design steps in the process of designing organizations. These operators help an organization designer create an organization design from scratch as well as offer the possibility to revise existing designs of organizations. The operators offer both top-down refinements and bottom-up grouping options. Importantly, the operators can be combined into complex operators that can serve as patterns for larger steps in an organization design process. The usability of the design operators is demonstrated in a running example. The contribution of this paper provides a solid basis for the development of a software environment supporting interactive organization design processes. This is demonstrated by an implemented prototype example tool.

1. Introduction

Organizations play a key role in the modern society. To a large extent, the vitality and productivity of an organization situated in an environment of a certain type depend on the kinds of structure and behavior of the organization that should conform to the environmental conditions. Business modeling is instrumental in addressing a number of problems such as: understanding the structure and dynamics of the organization [24], diagnosing problems and detecting avenues for improvement [15], ensuring common vocabulary and understanding, and formulating requirements needed for interoperability. A number of frameworks and tools for business modeling have been developed such as the OMG Business Modeling and Management Specifications (http://www.omg.org), ARIS [37], CIMOSA [14], MEMO [17], and IBM WebSphere Business Modeler. Less comprehensive with respect to scope, but still relevant, is the research on Enterprise Ontology [13] which aims at defining an abstract high-level model that captures the essence of the organization and thus enabling easier communication and shared understanding between inter- and intra-organizational parties.

The Business architecture defines the structure of the enterprise including its business processes and information and governance structure and is typically organized in a number of views clustering related aspects of the organizational structure and dynamics. For example, one important view is the organization-oriented view which defines the structure of the organization into business units, roles and their relationships, communication and capabilities. Another essential view is the process-oriented view which describes the business processes and tasks, the material and information resources needed for and produced by these processes and so on. This view has been the focus of extensive research and can be modeled by a number of existing methods and tools such as PetriNets [46], BPMN (http://www.bpmn.org), EPC [47], and YAWL (http://www.yawlfoundation.org). Some authors consider communication relations between organizational actors as organizational processes too (see e.g., [16]).

In this paper, for reasons of specificity, we focus on the organization-oriented view in the context of the framework presented in [31–34,41,42,21]. It is beyond the scope of the paper to discuss how the chosen framework compares to other existing ones—for this the reader is referred to [41]. The approach presented here, however, can be extended and adapted to other existing frameworks as well.

The specific problem addressed in the paper is to formalize the process of organizational design and organizational change in the context of the organization-oriented view. Organization design is concerned “with what an organization is ought to be” [30]. More specifically, Galbraith [18] stated that orga-
nization design “is conceived to be a decision process to bring about a coherence between the goals or purposes for which the organization exists, the patterns of division of labor and interunit coordination and the people who will do the work”. Further Galbraith argues that design is an essential process for “creating organizations, which perform better than those, which arise naturally”.

In the literature, a range of theories and guidelines concerning the design of organizations are present [18,12,28,5]. For example, Duncan proposed a contingency model for designing organizations with environmental variables being the principal determinants of organizational models. Mintzberg described a number of guidelines applicable mostly to designing hierarchical organizations that function in a relatively stable environment. However, despite the abundance of organizational design theories no general principles applicable to organizational design at all times and places can be identified [39]. Moreover, almost all theoretical findings in organizational design are informal and often vague. In order to provide an organization designer or a manager with operational automated tools for creating, analyzing, and revising organizations, in the first place a formal representation of an organization model as a design object description should be provided. In addition to this, to address the operations performed on such design object descriptions during a design process, a formal representation of design operators underlying possible design steps is needed. Such design operators describe the possible transitions between design object descriptions. Using the design operators, a design process can be described by choosing, at various points in time, the next operator to be applied to transform the current design object description into the next one. Examples of very simple design operators are adding or deleting an element of a design object description. More sophisticated design operators can involve, for example, the introduction of further refinement of the aggregation levels within a design object description. In this paper we introduce a formal organizational model format, to be used to represent design object descriptions for the organization-oriented view. On top of this, a set of design operators is formally defined. The formalization is based on the sorted predicate logic [26].

Often in the literature organizational design is recognized as an engineering problem [10]. From this perspective, design is considered as a continuous process of a gradual change of an organizational model by applying certain operations [30]. For example, [28] describes the design process as the following sequence of operations: given overall organizational needs, a designer refines the needs into specific tasks, which are further combined into positions. The next step is to build the “superstructure” by performing unit grouping using special guidelines and heuristics (e.g., grouping by knowledge and skill, by work process and function, by time, by place, etc.). Then, the grouping process is repeated recursively, until the organization hierarchy is complete.

For this paper, we aimed at identifying the most commonly and generally used set of operators for designing organizations. For this purpose, the literature from social sciences, and design principles used in other disciplines were investigated. For example, useful principles for organizational design can be found in the area of derivative grammars. Thus, graphical changes in organizational designs may be described by shape [45] and graph grammars [36]. Whereas changes in textual (or symbolic) structural and dynamic descriptions of organizational elements may be specified by string [11] and graph grammars, which allow representation of relationships between the descriptions of different elements. In order to relate graphical organizational designs to designs described in a symbolic form, parallel grammars (or grammars defined in multiple algebras) may be used [45]. For designing organization structures with multiple levels of representation (e.g., hierarchical organizations with departments, groups, sections) abstraction grammars [38] and hierarchical graph grammars [19] can be useful. By means of abstraction grammars, design is performed from the top level of the abstraction hierarchy to the bottom (most concrete) level, with each design generation using the prior level design as a pattern. Furthermore, mechanisms for choosing the most appropriate design generated by different transformations defined by grammars have been developed in different areas (e.g., recursive annealing in mechanical design [38]). Although it is widely recognized in social studies that no “best” design of an organization exists, a number of informal guidelines and best practices developed in the area of organizational design can help in identifying the most suitable organizational designs.

Thus, based on the rich literature on design, this paper makes a first attempt to formalize the operators underlying organization design processes. A set of design operators is formally introduced, which provides the means for creating a design of an organization from scratch as well as revising existing designs for organizations. Furthermore, the formalization of the operators provides a solid basis for a software tool supporting interactive organization design processes. A formal organizational specification provides a clear overview of organizational structures and dynamics, which would facilitate decision making by organizational managers. Furthermore, a formal, consistent specification of an organization may be used for analysis of organizational structures and interactions (e.g., identifying inconsistencies and bottlenecks in an organizational structure) by managers as a part of a dedicated knowledge-based system. Such a system may be also used for organizational design:

- it may advise on the choice of design operators based on the type of the organization under consideration (e.g., hierarchical, flat organic);
- it may be able to trace the design operations of the user and to suggest suitable design operators based on design patterns stored in the system.

The research presented in this paper does not overlap with the area of organizational learning [1,2,40,29], however there are some meeting points between the two areas. More specifically, within the technical view on organizational learning, the two areas can meet when organizational learning results in organizational change that will be reflected in the formal specification structure of the organization and/or the roles of the organization. Our research concerns the process of incorporating new elements and properties in the specification. It does not address the question of what these elements and properties should be in order to achieve a more faithful representation or to improve the functioning of the organization, nor does it consider the process of deriving them.

In Section 2 a formal framework for the specification of design object descriptions for organizations is described. Sections 3 and 4 introduce a set of classes of operators to create and modify design object descriptions for organizations. In Section 5 checking consistency of organizational specifications during the organizational design is considered. Section 6 illustrates the application of a developed prototype by an example. Finally, Section 7 discusses future work and provides general conclusions.

2. Format for an organizational model as a design object description

We consider a generic organization model, abstracted from the specific instances of agents (actors), which consists only of structural and behavioral descriptions of organizational roles and relations between them. A top-down ordering of definitions is used, meaning that concepts are referred to before they are defined.
Definition 1 (A specification of an organization). A specification of an organization with the name $O$ is described by the relation $\text{is\_org\_described\_by}(O, \Gamma, A)$, where $\Gamma$ is a structural description and $A$ is a description of dynamics.

An organizational structure is characterized by the patterns of relationships or activities in an organization, and described by sets of roles, groups, interaction and interlevel links, relations between them and an environment.

Definition 2 (A structural description of an organizational specification). A structural description $\Gamma$ of an organizational specification described by the relation $\text{is\_org\_described\_by}(O, \Gamma, A)$ is determined by a set of relations, among which:

- a relation $\text{has\_basic\_components}$ ($\Gamma, R, \text{G, II, ILL, ONT, M, ENV}$) defined on the subsets $R$, $\text{G, II, ILL, ONT, M, ENV}$ of the corresponding general sets $\text{ROLE}$ (the set of all possible role names), $\text{GROUP}$ (the set of all possible group names), $\text{INTER\_ACT\_LINK}$ (the set of all possible interaction links names), $\text{INTER\_LEVEL\_LINK}$ (the set of all possible interlevel links names), $\text{ONTO\_LINK}$ (the set of all possible ontology names), $\text{ONTO\_MAPPING}$ (the set of all possible ontology mappings names), $\text{ENVIRONMENT}$ (the set of all possible environment names)\footnote{The difference between $R$ and $\text{ROLE}$, for example, is that $R$ (subset of $\text{ROLE}$) is the set of all role names that occur in $\Gamma$.};
- a relation $\text{has\_input\_ontology}$ ($r, o$) that assigns an input ontology $o \in \text{ONT}$ to a role $r \in R$ (similarly the relations for output, internal, and interaction ontologies are introduced: $\text{has\_output\_ontology}$ ($r, o$), $\text{has\_interaction\_ontology}$ ($r, o$), $\text{has\_internal\_ontology}$ ($r, o$));
- a relation $\text{has\_input\_ontology}$ ($\text{env}, o$) that assigns an input ontology $o \in \text{ONT}$ to an environment $\text{env} \in \text{ENV}$ (similarly the relations for output, internal, and interaction ontologies are introduced: $\text{has\_output\_ontology}$ ($\text{env}, o$), $\text{has\_interaction\_ontology}$ ($\text{env}, o$), $\text{has\_internal\_ontology}$ ($\text{env}, o$));
- a relation $\text{is\_ontology\_for}$ ($e, o$) that assigns an ontology $o \in \text{ONT}$ either to a role $e \in R$ or an environment $e \in \text{ENV}$;
- a relation $\text{has\_onto\_mapping}$ ($\text{ll}, m$) that associates an interlevel link $\text{ll} \in \text{ILL}$ with an ontology mapping $m \in M$ (an ontology mapping for an interlevel link is defined similarly);
- a relation $\text{is\_interaction\_link\_of\_type}$ ($e, \text{type}$) that specifies an interaction link $e \in \text{IL}$ of one of the types: role interaction link, $\text{env\_input\_link}$, $\text{env\_output\_link}$;
- a relation $\text{connects\_to}$ ($e, r, r', \Gamma$) that specifies a connection by an interaction link $e \in \text{IL}$ from a source-role $r \in R$ to a destination role $r' \in r' \in \Gamma$;
- a relation $\text{connects\_to}$ ($e, \text{env}, r, \Gamma$) that specifies a connection by an interaction link $e \in \text{IL}$ of type $\text{env\_input\_link}$ from an environment $\text{env} \in \text{ENV}$ to a role $r \in R$ in $\Gamma$ (similarly for connects\_to ($e, \text{env}, r, \Gamma$));
- a relation $\text{subrole\_of\_in}$ ($r', r, \Gamma$) that specifies a subrole $r' \in R$ of a role $r \in \Gamma$;
- a relation $\text{member\_of}$ ($r, g, \Gamma$) that specifies a member role $r \in R$ of a group $g \in G$ in $\Gamma$.

\footnote{Notice that all the following relations are defined using the names of organization elements; the specifications for these elements will be provided in the following definitions.}

- a relation $\text{interlevel\_connection}$ ($\text{ii}, r, r', \Gamma$) that specifies a connection by an interlevel link $\text{ii} \in \text{ILL}$ between roles $r, r' \in R$ of adjacent aggregation levels (i.e., between a role and one of its subroles);

Organizational behavior is described by dynamic properties of the organizational structure elements.

Definition 3 (A description of dynamics of an organizational specification). A description of dynamics $A$ of an organizational specification described by the relation $\text{is\_org\_described\_by}(O, \Gamma, A)$ is determined by a set of relations, among which:

- a relation $\text{has\_basic\_components}$ ($A, \text{DP}$) that specifies a set of dynamic properties names $\text{DP}$ defined in an organizational specification;
- a relation $\text{has\_dynamic\_property}$ ($r, d$) that specifies a dynamic property $d \in \text{DP}$ for a role $r \in R$ (the relations for dynamic properties of an interlevel link, a group and an environment are defined in a similar manner: $\text{has\_dynamic\_property}$ ($e, d$), $\text{has\_dynamic\_property}$ ($\text{env}, d$));
- a relation $\text{has\_expression}$ ($d, \text{expr}$) that identifies a dynamic property name $d \in \text{DP}$ with a dynamic property expression $\text{expr} \in \text{DPEXPRESS}$ (e.g., a formula in sorted first-order predicate logic).

A role is a basic structural element of an organization. It represents a subset of functionalities, performed by an organization, abstracted from specific agents (or actors) who fulfill them. Each role has an input and an output interface, which facilitate the interaction (communication) with other roles. The interfaces are described in terms of interaction (input and output) ontologies: a vocabulary or a signature specified in order-sorted logic. An ontology contains objects that are typed with sorts, relations, and functions. Generally speaking, an input ontology determines what types of information are allowed to be transferred to the input of a role, and an output ontology predetermines what kinds of information can be generated at the output of a role.

Each role can be composed of a number of other roles, until the necessary detailed level of aggregation is achieved. Thus, roles can be specified and analyzed at different aggregation levels, which correspond to different levels of an organizational structure. A role that is composed of (interacting) subroles, is called a composite role.

Definition 4 (Role). A specification of a role $r$ is determined by:

**Objects:**
- $r, o, o', o'' \in \text{ONT}$, $o = o \cup o' \cup o''$, $o = o' \cup o''$, here $\cup$ is a functional symbol that maps names of ontologies to a name of the joint ontology

**Relations:**
- $\text{has\_internal\_ontology}$ ($r, o$), $\text{has\_input\_ontology}$ ($r, o'$), and $\text{has\_output\_ontology}$ ($r, o''$);
- $\text{has\_ontology}$ ($r, o$) and $\text{has\_interaction\_ontology}$ ($r, o$);
- $d \in \text{DP}$, $\text{has\_dynamic\_property}$ ($r, d$).

The ontologies, which describe interfaces of interacting roles, can be different. Therefore, if necessary, the specification of a role interaction process includes ontology mapping. An ontology mapping $m$ between ontologies $o$ and $o'$ is characterized by a set of relations $\text{is\_part\_of\_onto\_map}$ $a, a', m$, where $a$ is an atom expressed in ontology $o$ and $a'$ is an atom expressed using ontology $o'$.  

Definition 5 (Ontology mapping). An ontology mapping $m$ between ontologies $o$ and $o'$ is characterized by:
The conceptualized environment represents a special component of an organization model. According to some sociological theories (e.g., contingency theory), an environment represents a key determinant in organizational design, upon which an organizational model is contingent. Similarly to roles, the environment is represented in this proposal by an element having input and output interfaces, which facilitate in interaction with roles of an organization. The interfaces are conceptualized by the environment interaction (input and output) ontologies. Interaction links between roles and the environment are indicated in the organizational model as links that have a specific type, namely env_input_link or env_output_link by means of the predicate is_interaction_link_of_type.

The internal structure of the environment is not fixed, i.e., the designer has freedom to provide his/her own conceptualization of the environment. For example, the environment can be defined by a set of objects with certain properties and states and by causal relations between objects. On the one hand, roles are capable of observing states and properties of objects in the environment; on the other hand, they can act or react and, thus, affect the environment. We distinguish passive and active observation processes. For example, when some object is observable by a role and the role continuously keeps track of its state, changing its internal representation of the object if necessary, passive observation occurs. For passive observation, no initiative of a role is needed. Active observation is always concerned with the role's initiative and focusing. For particular purposes the internal specification for the environment can be conceptualized using one of the existing world ontologies (e.g., CYC, SUMO, TOVE [4]). However, despite the richness and the extensiveness of these ontological bases, more specific and refined types of concepts and relations are required for modelling particular types of organizations and environments.

**Definition 9 (Environment).** A specification of an environment env is determined by:

**Objects:**
- oe, oi, o', o'' ∈ ONT, oe = o ∪ o' ∪ o'' and oi = o' ∪ o''

**Relations:**
- has_internal_ontology (env, o), has_input_ontology (env, o'), and has_output_ontology (env, o'')
- has_ontology (env, oe) and has_interaction_ontology (env, oi)
- d ∈ DP, has_dynamic_property (env, d)

**Constraints:**
- IL' ⊆ IL, ∀e ∈ IL' is_interaction_link_in (e, G) ⇒ ∃r' ∈ R such that connects_to (e, env, r', G) ∨ ∃r'' ∈ R such that connects_to (e, r', env, G)

The behavior of each element of an organizational structure is described by a set of dynamic properties. With each name of a dynamic property, an expression is associated. Dynamic property expressions represent formulae specified over a certain ontology (ies). In particular, a dynamic property for a role is expressed using a role ontology. A dynamic property for an interaction link is constructed using the output ontology of a role-source of a link and the input ontology of a role-destination. A group dynamic property is expressed using ontologies of roles-members of a group.

**Definition 10 (Dynamic Property).** A specification of a dynamic property d ∈ DP is described by:

- has_expression (d, expr) for some expr ∈ DPEXPR
- uses_ont (d, o) for some o ∈ ONT
The application of the basic components of an organizational model is illustrated by means of a running example. Consider the process of organizing a conference. A partial model for the considered conference organization is shown in Fig. 1.

At the most abstract level 0 the organization is specified by one role CO (Conference Organization) that interacts with the environment Env. Role CO can act in the environment, for example by posting a call for papers in different media. Note, that the organizational model is depicted in a modular way; i.e., components of every aggregation level can be visualized and analyzed both separately and in relation to each other. Consequently, scalability of graphical representation of an organizational model is achieved. At the first aggregation level the internal structure of the composite role CO is revealed. It consists of subrole Ch (Conference Chair), which interacts with two other subroles: OC (Organizing Committee) and PS (the Paper Selection role). At the second aggregation level the internal structure of role PS is represented. It consists of subrole PCh (Program Chair), subrole PCM (Program Committee Member), and subrole R (Reviewer), which interact with each other. The input interface of role PS is connected to the input interface of its subrole PCh by means of an interlevel link. In our example the interlevel link describes the mapping between the input ontology of role PS and the input ontology of its subrole PCh. It means that information, transmitted to the role PS at the first aggregation level, will immediately appear at the input interface of subrole PCh, expressed in terms of its input ontology at the second aggregation level.

For example, if Ch requests some information from PS, the request actually arrives at the input of PCh. As a result of the internal communications among PCh, PCM and R, PCh will generate a reply that will appear as a response of PS for Ch.

For each element of the considered organizational model a set of dynamic properties is identified and formally specified in TTL. In fact, these properties define constraints on the behavior of elements, thus forming their expected behavioral repertoire in the organization.

For example, for the role Reviewer the dynamic property may be specified expressing that a reviewer should send his/her review to the Program Chair before a certain deadline. This property is expressed in TTL as follows:

\[ \forall t \in \text{time} \wedge \text{state}(t, \text{environment}) \Rightarrow \text{deadline_for_conference}(d) \Rightarrow \exists t' < d \text{ state}(t', t, \text{output}(\text{Reviewer})) \Rightarrow \text{communicate_from_to}(t, \text{Reviewer}, \text{Program Chair}, \text{inform}, \text{review_report}) \]

The predicate \( \text{communicate_from_to}(t; \text{ROLE}_1, t; \text{ROLE}_2, \text{s}_\text{act}; \text{SPEECH_ACT}, \text{message}; \text{STRING}) \) is used to specify the speech act \( \text{s}_\text{act} \) (e.g., inform, request, ask) from role-source \( \text{ROLE}_1 \) to role-destination \( \text{ROLE}_2 \) with the content message.

\[ \text{3. Representing design operators for organizational design} \]

In this section, a formal format for representing design operators is presented and, based on this format, formulations are introduced for a number of primitive design operators for designing organizations. Each primitive operator represents a specialized one-step operator to transform a design object description (organizational model) into a next one. Each operator is concerned with a part of the design object description to which it will be applied and the part of the transformed design object description, resulting from the operator application. The parts of the organization \( O \) that are being modified in terms of structure and dynamics (i.e., sets of dynamic properties) are specified using the in-focus relations: \( \text{structure_in_focus}(O, \text{RF}, \text{GF}, \text{ILF}, \text{ILLF}, \text{ONTF}, \text{MF}, \text{ENVF}) \) and \( \text{dynamics_in_focus}(O, \text{DPF}) \). With \( \text{RF} \subseteq \text{R}, \text{GF} \subseteq \text{G}, \text{ILF} \subseteq \text{IL}, \text{ILLF} \subseteq \text{ILL}, \text{ONTF} \subseteq \text{ONT}, \text{MF} \subseteq \text{M}, \text{ENVF} \subseteq \text{ENV}, \text{DPF} \subseteq \text{DP} \). The remaining parts of the organization stay the same.
The following operations all refer to an organization $O \in$ ORGANIZATION described by relations is_org_described_by $(O, I', A)$, has_basic_components $(I', R, G, IL, ILL, ONT, M, ENV)$. This organization is modified by an operator, leading to a second organization $O' \in$ ORGANIZATION described by relations is_org_described_by $(O', I'', A')$, has_basic_components $(I'', R', G', IL', ILL', ONT', M', ENV')$.

Our choice of primitive operators is motivated by different design guidelines and theories from social sciences [18,5,25], other disciplines, and our own research on formal modeling of organizations [9]. However, the application of the proposed set of operators is not restricted only to these theories. Thus, a designer has freedom to choose any sequence of operators for creating models of organizations. The operators are divided into three classes, which are consecutively described in the following subsections. Thus, in Section 3.1 the operators for roles are specified; in Section 3.2 the operators for different types of links are described; and in Section 3.3 the operators for groups are introduced.

### 3.1. Operators for roles

The classes of primitive operators for creating and modifying roles in a design object description for an organization are shown in Table 1.

A *role introduction operator* adds a new role to the organization. Usually, in organizational design after organizational tasks have been identified, these tasks should be further combined into positions (roles), based on the principles of labor division [23]. For example, in the conference organization setting, if the number of reviewers turns out to be insufficient, a Reviewer Recruiter role may contact researchers to ask them to review for the existing role in the organization and a *role retraction operator* removes all links connected to a role with their dynamic properties and mappings; it also deletes dynamic properties associated with the role and the role itself. In the example of the conference organization, when the Reviewer Recruiter has found enough reviewers, then the role can safely be removed from the organization.

#### 3.1.2. Role retraction operator

Let $op (O, O', \delta)$ be an operator that changes $O$ into $O'$ with a focus on $\delta$. Then $op$ is a role retraction operator iff it satisfies:

1. $\delta \in R$ such that is_role_in $(\delta, \Gamma)$
2. $\delta \notin R'$
3. structure_in_focus $(O, \{\delta\}, \emptyset, IFF, ILL, ONT)$ $\text{ILLf} = \{e \in IL | \exists r' \in R \text{ connects_to}_{(e, \delta, r', \Gamma)} \}$
   $\text{IFFf} = \{i \in ILL | \exists r \in R \text{ interlevel_connection}_{(i, \delta, r, \Gamma)} \}$
4. is_ontology_for $(\delta, \emptyset)$ $\text{ONTf} = \{m \in M | \exists i \in ILL \text{ has_onto_mapping}_{(i, m)} \}$
5. dynamics_in_focus $(O', \emptyset)$ $\text{DPf} = \{dp \in DP | \text{has_dynamic_property}_{(\delta, dp)} \}$

**Table 1**

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Introduction</td>
<td>Introduces a new role</td>
</tr>
<tr>
<td>Role Retraction</td>
<td>Deletes all links connected to a role with their dynamic properties and mappings; deletes a role and all dynamic properties associated with this role</td>
</tr>
<tr>
<td>Role Dynamic Property</td>
<td>Adds a new dynamic property to a role</td>
</tr>
<tr>
<td>Property Addition</td>
<td>Deletes an existing role dynamic property</td>
</tr>
<tr>
<td>Revocation</td>
<td></td>
</tr>
</tbody>
</table>

A *role dynamic property addition operator* creates a new property for the existing role in the organization and a *role dynamic property revocation operator* deletes a property from the dynamic description of a role.

#### 3.1.3. Role dynamic property addition operator

Let $op (O, O', \delta)$ be an operator that changes $O$ into $O'$ with a focus on $\delta$. Then $op$ is a role dynamic property addition operator iff it satisfies:

A *role dynamic property revocation operator* creates a new property for the existing role in the organization and a *role dynamic property revocation operator* deletes a property from the dynamic description of a role.

#### 3.1.1. Role introduction operator

Let $op (O, O', \delta)$ be an operator that changes $O$ into $O'$ with a focus on $\delta$. Then $op$ is a role introduction operator iff it satisfies:

1. $\delta \notin R$, $\delta \in R'$ such that is_role_in $(\delta, \Gamma')$
2. structure_in_focus $(O, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
3. structure_in_focus $(O', \{\delta\}, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$ where $\text{ONTf} = \text{is_ontology_for}_{(\delta, \emptyset)}$

A role introduction operator adds a new role to the organization and all dynamic properties associated with this role. In the example of the conference organization, when the Reviewer Recruiter has found enough reviewers, then the role can safely be removed from the organization.

**Fig. 1.** Model of the conference organizing committee.
3.1.4. Role dynamic property revocation operator

Let \( \text{op} (O, O', \delta) \) be an operator that changes \( O \) into \( O' \) with a focus on \( \delta \). Then \( \text{op} \) is a role dynamic property revocation operator iff it satisfies:

1. \( \text{dynamics_in_focus} (O, \varnothing) \)
2. \( \text{dynamics_in_focus} (O', \text{DPf}) \)

\( \text{DPf} = \{ \delta \in \text{DP} | \exists \gamma \in \text{R'} \text{ has dynamic_property} (r, \gamma) \} \)

3.2. Operators for links

In this subsection, we propose a set of classes of primitive operators for creating and modifying links in a design object description for an organization (see Table 2).

An interaction link addition operator allows the creation of an interaction link (information channel) between two existing roles in the organization. In the organizational design, after organizational subtasks are assigned to roles, the problem of coordination of interdependencies among subtasks should be solved.

In the conference management example, the Program Chair (playing in this case a managerial role) may request two reviewers to discuss their reviews. This requirement can be handled by the addition of interaction links between the appropriate reviewer roles in the design object description for an organization (see Fig. 3).

3.2.1. Interaction link addition operator

Let \( \text{op} (O, O', \delta) \) be an operator that changes \( O \) into \( O' \) with a focus on \( \delta \). Then \( \text{op} \) is an interaction link addition operator iff it satisfies:

1. \( \delta \in \text{IL}, \delta \in \text{IL'} \) such that \( \text{is_interaction_link_in} (\delta, \Gamma') \)
2. \( \text{structure_in_focus} (O, \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing, \text{Mf}) \)
3. \( \text{structure_in_focus} (O', \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing, \varnothing) \)
4. \( \text{dynamics_in_focus} (O, \text{DPf}) \)
5. \( \text{dynamics_in_focus} (O', \varnothing) \)

An interaction link deletion operator is used to delete an existing interaction link between two roles as well as to revoke all dynamic properties, associated with this link. For example, the Program Chair has taken care of the acceptance proceedings for the conference. He does not need to be in contact with the reviewers anymore. This case can be handled by the deletion of the interaction between two roles in the design object description for an organization.

3.2.2. Interaction link deletion operator

Let \( \text{op} (O, O', \delta) \) be an operator that changes \( O \) into \( O' \) with a focus on \( \delta \). Then \( \text{op} \) is an interaction link deletion operator iff it satisfies:

1. \( \delta \notin \text{IL}, \delta \in \text{IL} \) such that \( \text{is_interaction_link_in} (\delta, \Gamma') \)
2. \( \text{structure_in_focus} (O, \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing, \text{Mf}) \)
3. \( \text{structure_in_focus} (O', \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing, \varnothing) \)
4. \( \text{dynamics_in_focus} (O, \text{DPf}) \)
5. \( \text{dynamics_in_focus} (O', \varnothing) \)

An interaction dynamic property addition operator creates a new property for an existing interaction link. An interaction dynamic property revocation operator deletes a property from the dynamic description of an interaction link.

3.2.3. Interaction dynamic property addition operator

Let \( \text{op} (O, O', \delta) \) be an operator that changes \( O \) into \( O' \) with a focus on \( \delta \). Then \( \text{op} \) is an interaction dynamic property addition operator iff it satisfies:

1. \( \delta \in \text{DP}, \delta \in \text{DP'} \) with a focus on \( r \)

\( \text{dp} = \{ m \in \text{M} | \text{has_oneto_one_mapping} (r, \delta, \gamma) \} \)

2. \( \text{structure_in_focus} (O, \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing, \varnothing) \)
3. \( \text{dynamics_in_focus} (O, \text{DPf}) \)

\( \text{DPf} = \{ \text{dp} \in \text{DP} | \text{has_dynamic_property} (\delta, \text{dp}) \} \)

4. \( \text{dynamics_in_focus} (O', \varnothing, \varnothing, \delta, \varnothing, \varnothing, \varnothing) \)

5. \( \text{dynamics_in_focus} (O', \varnothing) \)

An interaction dynamic property revocation operator deletes a property from the dynamic description of an interaction link.
3.2.4. Interaction dynamic property revocation operator

Let \( \text{op}(O, O', \delta) \) be an operator that changes \( O \) into \( O' \) with a focus on \( \delta \). Then \( \text{op} \) is an interaction dynamic property revocation operator iff it satisfies:

1. \( \text{dynamics}_{\text{in}}(O, \emptyset) \)
2. \( \text{dynamics}_{\text{in}}(O', \text{DPr}') \)
   \[ \text{DP} = \{ \delta \in \text{DP} \mid \exists \varepsilon \in \text{IL} \text{ has dynamic property } (e, \delta) \} \]

3.3. Operators for groups

The classes of primitive operators for creating and modifying groups in a design object description for an organization are shown in Table 3.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouping</td>
<td>Combines roles into groups</td>
</tr>
<tr>
<td>Degrouping</td>
<td>Moves roles outside of a group and deletes the group</td>
</tr>
<tr>
<td>Group-to-Role</td>
<td>Transforms groups into roles</td>
</tr>
<tr>
<td>Role-to-Group</td>
<td>Transforms roles into groups</td>
</tr>
</tbody>
</table>

3.3.1. Grouping operator

Let \( \text{op}(O, Rg, O', Gn) \) be an operator that changes \( O \) into \( O' \) wrt. \( Gn \in G, Rg \subseteq R \). Then \( \text{op} \) is a grouping operator that creates a new group \( Gn \) from the subset of roles \( Rg \) iff it satisfies:

**Structural aspect:**
1. \( \forall a \in Rg \text{ member of } in (a, Gn, \Gamma) \).
2. \( \text{structure}_{\text{in}}(O, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset) \)
3. \( \text{structure}_{\text{in}}(O', \emptyset, Gn, \emptyset, \emptyset, \emptyset, \emptyset) \)

**Dynamic aspect:**
1. \( \text{dynamics}_{\text{in}}(O, \emptyset) \)
2. \( \text{dynamics}_{\text{in}}(O', \text{DPr}) \)
3. \( \text{EPr}_{\Gamma}' \in \text{IL} \mid \exists r1 \in Rg \exists r2 \in Rg \text{ connects to } (e, r1, r2, \Gamma) \)
   \[ \text{DP} = \{ \delta \in \text{DP} \mid \exists \varepsilon \in \text{Er} \text{ has dynamic property } (r, \varepsilon) \} \]
4. \( \text{DP} \subseteq \text{DCL}(\text{DPr}) \), where \( \text{DCL}(\text{DPr}) \) is the deductive closure of \( \text{DPr} \)

---

Fig. 4. Application of the interlevel link introduction operator for adding an interlevel link between Paper Selection role and Program Committee Member role (PCM).

---

**Table 3**

<table>
<thead>
<tr>
<th>Operator classes for creating and modifying groups.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Class</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Grouping</td>
</tr>
<tr>
<td>Degrouping</td>
</tr>
<tr>
<td>Group-to-Role</td>
</tr>
<tr>
<td>Role-to-Group</td>
</tr>
</tbody>
</table>
A natural dual to the role grouping is role degrouping. This operator takes a group of roles and moves the roles to outside of the group. Role Degrouping transforms a group into a set of roles.

### 3.3.2. Degrouping operator

Let \( \text{op} \ (O, Gd, O', Rdg) \) be an operator that changes \( O \) into \( O' \) wrt. \( Gd \in G \) \( , \) and \( Rdg \subseteq R' \). Then \( \text{op} \) is a degrouping operator iff it satisfies:

**Structural aspect:**
1. \( Rdg = \{ r \in R| \text{member of in} \ (r, Gd, \Gamma) \} \)
2. \( Gd \in G' \)
3. \( \text{structure.in.focus} (O, Gd, [Gd], \varnothing, \varnothing, \varnothing, \varnothing) \)
4. \( \text{structure.in.focus} (O', G, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing) \)

**Dynamic aspect:**
1. \( \text{dynamics.in.focus} (O, \text{DPf}) \ dpf = \{ \text{dp} \in \text{DP}| \text{has.dynamic.property} (\text{Gd}, \text{dp}) \} \)
2. \( \text{dynamics.in.focus} (O', \varnothing) \)

A group can be transformed into a role, a more coherent, integrated and formal organizational unit with proper interfaces (e.g., a department of an organization). For a group to act as a role, it should have well-defined (formalized) input and output interfaces. A Group-to-Role operator takes a group and adds these interfaces. In an organic organization with loosely defined frequently changing structure this would correspond to the formalization of one of the organizational units, i.e., providing a formal (permanent) structural description with the subsequent specification of formal functional procedures. For example, in the conference organization setting, the Program Committee group from the Paper Selection role can be further transformed into the Program Committee role, a formal organizational unit with certain characteristics and functions (e.g., final decision making for the paper acceptance). Such transformation can be achieved by means of the Group-to-Role operator (see Fig. 6). The next logical step would be to limit the interactions of the subroles of the Program Committee role only to those that exist within the Program Committee role, and replace all interactions with the roles outside of the Program Committee role by corresponding interactions between outer roles and the Program Committee role. This can be done by applying interaction and interlevel link addition and retraction operators. In this case reviewers should follow a formal procedure for interactions with the Program Committee role and cannot directly address any arbitrary Program Committee Member.

### 3.3.3. Group-to-Role operator

Let \( \text{op} \ (O, g, O', r) \) be an operator that transforms group \( g \in G \) in \( O \) into role \( r \in R' \) in \( O' \). Then \( \text{op} \) is a Group-to-Role operator iff it satisfies:

**Structural aspect:**
1. \( r \notin R, g \notin G' \)
2. \( \forall a \in R \text{member.of.in} (a, g, \Gamma) \Rightarrow \text{subrole.of.in} (a, r, \Gamma') \)
3. \( \text{structure.in.focus} (O, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing) \)
4. \( \text{structure.in.focus} (O', \{r\}, \varnothing, \varnothing) \)

**Dynamic aspect:**
1. \( \text{dynamics.in.focus} (O, \text{DPf}) \ dpf = \{ \text{dp} \in \text{DP}| \text{has.dynamic.property} (g, \text{dp}) \} \)
2. \( \text{dynamics.in.focus} (O', \text{DPf}) \ dpf = \{ \text{dp} \in \text{DP}| \text{has.dynamic.property} (r, \text{dp}) \} \)
3. \( \text{DPf} \Rightarrow \text{DPf} \)

A role may consist of several other roles that are not exposed to the rest of the world. When a role is converted to a group, it exposes the input and output interfaces of the roles inside it. Transforming a role into a group results in the subroles now residing on the level of the prior composite role. For example, during the reorganization some formal organization units (e.g., sections and departments) have been eliminated, whereas the roles that constituted these units and relations between them were kept, thus, creating a basis for new organizational formations.

### 3.3.4. Role-to-Group operator

Let \( \text{op} \ (O, r, O', Gr) \) be an operator that changes \( O \) into \( O' \), wrt. \( r \in R, \) and \( Gr \in G' \). Then \( \text{op} \) is a Role-to-Group operator that transforms role \( r \) into group \( Gr \) iff it satisfies:

**Structural aspect:**
1. \( Gr \notin G, r \notin R' \)
2. \( \forall a \in R \text{subrole.of.in} (a, r, \Gamma) \Rightarrow \text{member.of.in} (a, Gr, \Gamma) \)
3. \( \text{structure.in.focus} (O, \{r\}, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing) \)
4. \( \text{structure.in.focus} (O', \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing) \)

Fig. 5. Application of the grouping operator to create the Program Committee group that consists of roles Program Chair (PCh) and Program Committee Member (PCM) for making final decisions concerning paper acceptance.

Fig. 6. Application of the Group-to-Role operator to transform Program Committee group into the Program Committee role.
4. Composing operators

The primitive operators described above reflect major principles of organizational design. In practice, in addition to the primitive operators more complex operators are used. Complex operators are represented as a combination of a certain number of primitive operators; some of them are given in Table 4.

Sometimes an effect produced by application of some composite operator to a design object description for an organization can be achieved by different combinations of primitive operators.

Consider the Role refinement operator as an example. This operator divides a role into several roles such that the role properties of the first role are distributed over the newer roles. In organizational design, role refinement corresponds to the fine-tuned specialization and division of labor for increasing efficiency. It is usually recommended to divide the work so that the portions can be differentiated rather than similar, and that each role is responsible for a small portion of the overall task. According to Adam Smith, division of labor is limited by the extent of the market; other general principles of labor division can be found in [23].

Let us illustrate the application of the Role refinement operator in the context of the conference organizing example. In Fig. 7 the design object description for an organization is represented at the first aggregation level. Consider the situation when the decision is made to divide the tasks of Organizing Committee (OC) between the Local Organizing Committee (LOC), which is hence responsible for negotiations with publishers for printing proceedings and arranging the conference venue, and the General Organizing Committee (GOC), which is designated for solving financial and other questions. Thus, the role OC is refined into two new roles LOC and GOC. These roles are able to interact with each other and with the role Conference Chair.

Alternatively, every composite operator can be considered as an aggregated one-step operator. Such descriptions define formal conditions for a design object description for an organization before and after the application of a complex operator; therefore, they can serve the purpose of checking integrity and consistency of a design object description.

An example of such a representation for the Role refinement operator is given below.

4.1. Refinement operator (integrity definition)

Let \( op (O, r, O', R_{ref}) \) be an operator that refines role \( r \in R \) in \( O \) into a set of roles \( R_{ref} \subseteq R' \) in \( O' \). Then \( op \) is a refinement operator iff it satisfies:

**Structural aspect:**

1. \( r \in R, r \notin R, R_{ref} \cap R' = \emptyset \)
2. \( \text{structure in focus } (O, \{r\}, \emptyset, \text{ILf, ILLf, ONTf, Mfa, } \emptyset) \)
   \[ \text{ILf=} \{e \in \text{IL}\mid \exists r' \in R \text{ connects_to } (e, r', r, \Gamma) \text{ OR } \exists r' \in R \text{ connects_to } (e, r, r', \Gamma) \} \]
   \[ \text{ILLf=} \{\text{ill} \in \text{ILL} \mid \exists r' \in R \text{ interlevel_connection } (\text{ill}, r', r, \Gamma)\} \]
   \[ \text{ONTf=} \{o \in \text{ONT} \mid \text{has_ontology } (r, o)\} \]

\[ R_{ref} \text{ connects } (\text{Mf}, \text{Mfi}, \text{Mfo}) \]
\[ \text{Mf=} \{m \in \text{M} \mid \exists e \in \text{IL} \text{ has_onto_mapping } (e, m)\} \]
\[ \text{Mfi=} \{m \in \text{M} \mid \exists e \in \text{IL} \text{ has_onto_mapping } (e, m)\} \]
\[ \text{Mfo=} \{m \in \text{M} \mid \exists e \in \text{IL} \text{ has_onto_mapping } (e, m)\} \]

**Dynamic aspect:**

1. \( \text{dynamics in focus } (O, \text{DPf}) \text{ DPF=} \{dp \in \text{DP} \mid \text{has_dynamic_property } (r, dp)\} \)
2. \( \text{dynamics in focus } (O', \text{DPf'}) \text{ DPF'}=} \{dp \in \text{DP} \mid \text{has_dynamic_property } (g, dp)\} \)

![Fig. 7. Example of the application of the Role refinement operator, in which the Organizing Committee role (OC) is refined into the Local Organizing Committee (LOC) and General Organizing Committee roles (GOC).](image-url)
3. structure_in_focus (O’, Ref, ) II f, ONT f, Mf a, ) II f = \{e ∈ II | ∃1 ∈ Ref ∃2 ∈ Ref \text{connects}_\text{to} (e, r_1, r_2, Γ)\} OR ∃1 ∈ Ref ∃2 ∈ R, r_2 \# Ref \text{connects}_\text{to} (e, r_1, r_2, Γ) OR ∃1 ∈ Ref ∃2 ∈ R, r_2 \# Ref \text{connects}_\text{to} (e, r_2, r_1, Γ)).

ILL f = ILL f ∪ ILL f

ILL f = \{ \text{\text{interlevel\_connection}}(\text{ill}_r, r_1, Γ)\}

ILL f = \{ \text{\text{interlevel\_connection}}(\text{ill}_r, r_1, Γ)\}

Mf a = Mf a ∪ Mf a ∪ Mfo

Mf = \{m ∈ M | ∃e ∈ II f \text{has\_onto\_mapping} (e, m)\}

Mf = \{m ∈ M | ∃e ∈ II f \text{has\_onto\_mapping} (e, m)\}

Mf = \{m ∈ M | ∃e ∈ ILL f \text{has\_onto\_mapping} (e, m)\}

ONT f = \{o ∈ ONT | ∃1 ∈ Ref \text{has\_ontology} (r_1, o)\}.

4. \forall e ∈ I L, \forall b ∈ R, \forall r_1, r_2, e \# Ref \text{connects}_\text{to} (e, r, b, Γ) ⇒ \exists e’ ∈ I L’, \exists r_‘ ∈ Ref \text{connects}_\text{to} (e’, r’, b, Γ)\}

5. \forall e’ ∈ I L’, \forall r’ ∈ Ref \forall b ∈ R’ and b \# Ref \text{connects}_\text{to} (e, r, b, Γ) ⇒ \exists e’ ∈ I L, \exists r’ ∈ Ref \text{connects}_\text{to} (e, r, b, Γ)\}

6. \forall a, a’, m ∈ Mf i is_part_of_onto_map (a, a’, m) ⇒ ∃a”, m’ ∈ Mf i is_part_of_onto_map (a”, m’)

Dynamic aspect:

Fig. 8. Example of the application of the Adding Aggregation Levels operator, in which the roles Program Chair (PCh) and Program Committee Member (PCM) are grouped together and transformed into the Paper Selection (PC) role.

5. Consistency of organizational specifications

To ensure internal consistency and validity of organizational specifications, specification constraints are identified, which can be checked automatically during the design process. The role of the constraints may differ in organization modeling which influences their format, purpose and way of use. Here we present a classification framework for constraints covering a range of
perspectives on organizations from very detailed to global, and from internal to external point of view, connecting the organization with its environment.

Specification constraints can be checked at every step of the design process in order to ensure the consistency and validity of the current specification. They can be classified based on their origin into: generic constraints that need to be satisfied by any organizational specification; domain-specific constraints dictated by the application domain of the specification.

Two types of generic constraints are considered: structural consistency constraints used to ensure consistency of the specification; constraints imposed by the physical world – the laws of the physical world render certain events, relationships between concepts, etc. impossible (e.g. a role cannot be at two locations at the same time).

The consistency of a specification is checked w.r.t. the set of structural consistency constraints. These constraints are axioms of the specification language and their logical consequences formulated based on the definitions of the language and reflecting the rules of correct and consistent use of the elements of the language in modeling. These constraints ensure internal integrity of the structures defined using the language. A specification S is consistent w.r.t. a set of structural consistency constraints SCC iff in each of its models each formula from SCC is true: \((l, v) = SCC\), where \(l\) is an interpretation of the sorts, functions and predicates of the language of S and \(v\) is a valuation of variables in \(S\).

Consider two examples of structural consistency constraints for the organization-oriented view:

**CS1**: A role should receive only information types specified by its input ontology.

**CS2**: An information type outputted by some role should be related by the corresponding mapping to an information type from the ontology of the role-recipient.

Domain-specific constraints are imposed by the application domain in which the particular specification will be used and can be classified according to their sources:

*Constraints imposed by the organization* have been chosen (e.g. by the management of the company) as necessary and need to be satisfied by any specification for the particular organization. Such constraints can often be found in company policy documents, internal procedures descriptions, etc. *Constraints coming from external parties* are enforced by external parties (e.g. the society or government) and contain rules about working hours, safety procedures, emissions, etc. Sources for such constraints are regulations, agreements, etc. *Constraints of the physical world* come from the physical world w.r.t. the specific application domain and should be satisfied by any specification in this domain (in contrast to the generic physical constraints which should be satisfied by any specification irrespective of the application domain).

The *validity* of a specification is checked w.r.t. a set of physical-world and domain-specific constraints. An organizational specification \(S\) is valid w.r.t. a set of physical world and domain-specific constraints \(C\) iff in each of its models each formula from \(C\) is true: \((l, v) = C\).

To reduce the complexity of modeling and analysis, organizational specifications can be considered at different aggregation levels (e.g., to investigate certain organizational aspects, while abstracting from irrelevant details). To ensure consistency of specifications and sets of constraints of different aggregation levels, and integrity of a complete organizational specification, a set of inter-level consistency constraints is defined. A part of these constraints belong to the class of generic structural consistency constraints. For example, CS3: A role can be a subrole of one role at most.

In the structure \(G\), \(r_1, r_2, r_3, r_4\): role \(r_1\) subrole-of-in \((r_1, r_2, r_3, r_4)\) & sub-role-of-in \((r_3, r_2, r_3)\)

CS4: Each subrole of a composite role \(r\) should interact with at least one other subrole of \(r\).

In the structure \(G\): role \(r_1, r_2, r_3, r_4\) & sub-role-of-in \((r_1, r_2, r_3, r_4)\) & connects-to \((e, r_2, r_1, r_3)\)

CS5: Information provided to the input of a composite role should be further transmitted to one or more of its subroles.

CS6: No role can be a subrole of itself at any aggregation level.

CS7: Information generated at the output of a composite role is transferred from the output of one of its subroles.

CS8: Any subrole of a composite role is not allowed to interact directly with any other role outside of this composite role.

The rest are domain-specific and should be identified and checked for a particular organization. For example:

CS9: Information of a type inf produced by a role \(r_1\) for a role \(r_2\) should be able to reach \(r_2\).

For checking if a path exists for communicating \(inf\) from \(r_1\) to \(r_2\) the following algorithm is proposed.

**Algorithm 1. CHECK-EXISTENCE-OF-INTERACTION-PATH**

1. \(i := \text{max}(\text{AGR}_{LEVEL}(r_1), \text{AGR}_{LEVEL}(r_2))\), \(r_1 \leftarrow r_1, r_2 \leftarrow r_2\)
2. if \(\exists r_1, r_2, \exists \Gamma \text{subrole-of-in}(r_1, r_1, \Gamma)\) and \(\exists \text{subrole-of-in}(r_2, r_2, \Gamma)\) and \(\Gamma = r_2\), then return \(true\), else return \(false\).
3. if \(\text{IS}_PATH\_FROM\_TO\_FOR(r_1, r_2, \inf) = true\), then return \(true\), else return \(false\).
4. if \(\text{AGR}_{LEVEL}(r_1) \geq i\) and \(\exists \text{subrole-of-in}(r_1, r_1, \Gamma)\) then return \(false\).
5. if \(\text{IS}_PATH\_FROM\_TO\_FOR(r_1, r_1, \inf) = false\), then return \(false\).
6. \(r_1 \leftarrow r_1, r_2 \leftarrow r_1\), then return \(true\).
7. if \(\text{AGR}_{LEVEL}(r_2) \geq i\) and \(\exists \text{subrole-of-in}(r_2, r_2, \Gamma)\) then return \(false\).
8. if \(\text{IS}_PATH\_FROM\_TO\_FOR(r_2, r_2, \inf) = false\), then return \(false\).
9. \(r_1 \leftarrow r_1\), then return \(false\).
10. \(r_1 \leftarrow r_2\), then return \(false\).
11. if \(\text{AGR}_{LEVEL}(r_2) \geq i\) and \(\exists \text{subrole-of-in}(r_2, r_2, \Gamma)\), then return \(false\).
12. \(r_1 \leftarrow r_1\), then return \(false\).
13. \(r_1 \leftarrow r_1\), then return \(false\).
14. \(r_1 \leftarrow r_2\), then return \(false\).
15. \(i \leftarrow i - 1\)
16. \(\text{until } i > 0\), perform steps 2–15.

**Function AGR_{LEVEL}(r)**

**Output**: returns the aggregation level number for role \(r\)

1. \(i \leftarrow 1, r \leftarrow r\)
2. until \(\exists r h r \neq \text{ORG} \land \exists \Gamma \text{subrole-of-in}(r, r, \Gamma)\), perform step 3
3. \(r \leftarrow r, l \leftarrow l + 1\)
4. return \(l\)

**Function IS\_PATH\_FROM\_TO\_FOR(src, dest, inf)**

**Output**: returns true if a communication path exists from role src to role dest for information type inf, or returns false otherwise.

1. \(R \leftarrow \{\text{src}\}, R \leftarrow \emptyset\)
2. \(R \leftarrow R \cup RT\)
3. \(R \leftarrow \{r \not\in \text{ORT} \land \exists e \exists r_1 \in R \exists \Gamma \text{connects-to}(e, r_1, r_2, \Gamma)\), and has_onto_mapping(e, inf, inf)\}
4. if \(\text{dest} \in RT\), then return \(true\).
5. until \(RT \not\subseteq R\), perform 2–5.
6. return \(false\).

The general idea of the algorithm is to check if a communication path exists at every aggregation level, through which information is transferred on its way from the role-source to the role-destination. The algorithm begins from the maximum aggregation level.
of both roles (Algorithm 1:1). Then the information flow is con-
structed gradually by proceeding from both ends (the source
and the destination) simultaneously (from the source- Algorithm 1:
7–10, from the destination- Algorithm 1: 11–14) until the point
is reached, at which the source-part flows directly into the destina-
tion part (Algorithm 1: 2–5 the parts connect at one level; 6: the
parts connect across two levels). The worst case time complexity
of the algorithm is estimated as O (|LEVEL| · |LINK|2), where
|LEVEL| is the number of aggregation levels, and |LINK| is the num-
ber of links between roles in the specification.

An instantiated version of this constraint with the template
CS8(role1, role2, inf) for the conference organization case study con-
sidered in the paper is the following:

CS9(Conference Chair, Program Committee Member, inquiry):
An inquiry about a paper sent by Conference Chair to Program Committee
Member should be able to reach Program Committee Member.

6. A prototype tool to support the design of organizations

The formal representations of the organization’s entities and the
design operators described in this paper provide a solid basis for
the development of a software environment supporting interactive
organization design processes.

For the purpose of illustration and evaluation, a prototype tool
was implemented, using the LEADSTO environment [7] and TTL
Checking environment [8]. This tool supports organizational design
and allows organization designers to investigate its dynamics. The
LEADSTO tool was used to implement design operators and the de-
sign process. The TTL Checking tool was used to check the consist-
cy of the organizational specification during the design process.

The application of the design prototype is demonstrated on the
example of role refinement as described in the previous Section.
The dynamics of the design process is described in Table 5. Parts
of the corresponding design specification implemented in the tool
are provided in Fig. 9. Based on this specification, a trace was
generated automatically, a part of which is provided in Fig. 10 (the
complete trace is provided in Appendix A).

A design specification consists of three essential types of ele-
ments: sorts, intervals and rules. Sorts (Fig. 9a) are used to define
types of entities used in the design process (e.g., roles, design oper-
ators). Intervals are used to specify the input from the designer,
external events and the initial design object description. For exam-
ple, the interval in Fig. 9b specifies that the designer supports the
choice of the Role refinement operator for role OC in ORG in the
time interval [4, 5]. The design process and the design operators
introduced in this paper are described using generic leadsto-rules.
These rules are expressed in the form of direct temporal dependen-
cies between two state properties in successive states. The format
is defined as follows. Let α and β be state properties of the form
‘conjunction of atoms or negations of atoms’, and e, f, g, h non-
negative real numbers. In the LEADSTO language the notation
$\alpha \land (e, f, g, h)$ means: if state property $\alpha$ (antecedent indicated by
A in Fig. 9c) holds for a certain time interval with duration $g$, then
after some delay (between e and f) state property $\beta$ (consequent
indicated by C in Fig. 9c) will hold for a certain time interval of
length h. The default time parameters are $e = f = 0$ and $g = h = 1$.

For example, the rule in Fig. 9c generates possible operator alterna-
tives, which could be applied to a role specified by the variable r.

In the design process, first, a designer chooses a part of the de-
sign object description, on which she intends to put her attention
(in the considered example it is the role Organizing Committee).
Next, the software proposes to the designer a number of operators, which
are potentially applicable to the chosen part of the design
object description. The designer chooses one of them, for the
example, the Role refinement operator. Role refinement is a com-
posite operator that consists of an ordered sequence of primitive
operators. Usually, most of the primitive operators constituting
composite ones are imperative (e.g., Role Introduction for Refine-
ment); yet application of some of them may be postponed to the
future (e.g., Role dynamic property addition for Refinement) or
skipped (e.g., Interlevel link deletion for Refinement). Further,
the tool demands specifying roles, into which role OC has to be refined.
The designer specifies role names (for this example, Local Organiz-
ing Committee (LOC) and General Organizing Committee (GOC))
and their ontologies. At this step the software will check if the
input ontology of role OC constitutes a subset of the union of
the input ontologies elements of roles LOC and GOC. This is done by
automated checking of the corresponding TTL property using the
TTL Checking tool (see Fig. 11). Note that TTL properties can be
defined in advance and be checked on the design specification at any
step of the design process.

After that the software tool requests the designer to specify dy-
namic properties for the created roles. The designer may postpone
this operation to a future point in time. Thereafter, the tool pro-
poses to add interaction links between roles LOC, GOC and role
Conference Chair (Ch), with which the original role OC was con-
ected. At this step it is checked based on the integrity definition
for refinement, whether the links, corresponding to the interaction
links between Ch and OC in the original design object description,
are present in the obtained design object description. Furthermore,
if the original role had interlevel links with other roles, these links
need to be deleted, and new interlevel links will be added. After
that the integrity constraints for the ontology mappings for these links need to be checked according to the integrity definition of the refinement operator. As the last step, role OC and interaction links connecting it with role Ch, as well as dynamic properties of role OC and its interaction links are automatically removed from the design object description.

7. Discussion

This paper introduces a representation format and a variety of operators for the design of organization specified in this representation format. The described operators have several important characteristics. First, they can be combined into composite oper-
In this paper we focused on designing specifications of the organization-oriented view from the generic organization modeling and analysis framework [41]. In the future, the design of other views of the framework such as the process-oriented view [31] will be addressed as well.

To a certain extent organizations can be considered as compositional systems [48]. However, models and design methods for such systems do not allow representing many organization domain-specific concepts and operators (e.g., a group, a Group-to-Role operator) and, therefore, cannot capture many important organization phenomena.

In the area of component-based software engineering a number of design patterns for building software components (e.g., refinement, chaining, disjoint composition) have been introduced [20]. These patterns specify general-purpose manipulations with programming constructs (e.g., interface and private methods of components); while in organizational design literature organization transformations are described using domain-specific concepts. The formal representation format proposed in this paper bridges this gap and facilitates the abstraction of organization domain into general-purpose programming design patterns.

Formal specification of design processes enables verification of structural and dynamic consistency of a design object description for an organization. The verification of structural consistency is based on the consistency definitions for operators, such as one given in Section 4 for the Role refinement operator, and the consistency constraints described in Section 5. To check the consistency constraints, algorithms were developed and implemented, some of which were considered in the paper. For verifying dynamic consistency (e.g., checking relations between dynamic properties defined at different aggregation levels of a model representation) model checking techniques [27,43] may be used. Furthermore, verification mechanisms based on certain requirements on organizational functioning and performance (e.g., using organization performance indicators) were developed [33].

Another way to evaluate an organizational model is by performing simulations. For this purpose, agents with different types of attitudes and internal architectures may be allocated to roles within an organization model on certain conditions. After that, by considering different types and sequences of environmental influences provided within certain simulation scenarios, traces (i.e., temporal sequences of events in the environment and within the organization) corresponding to the execution of scenarios can be generated. These traces may be further used for analysis of the organizational model, more specifically, for evaluating different global properties of the organizational model (e.g., robustness, stability, efficiency, and effectiveness).

In conclusion, this paper introduced a representation format and a set of formally represented design operators dedicated to the design of organizations of most types. Although the choice of operators is motivated by different theories and guidelines from the area of organizational design, the application of the proposed operators is not restricted to any theories from social studies. The formalization of the operators provides a solid basis for the development of a software tool supporting interactive organization design processes. A prototype implementation for such a tool is demonstrated by an example in this paper.

Acknowledgments

This research is partly supported by the Dutch Technology Foundation STW, which is the applied science division of NWO, and the Technology Programme of the Ministry of Economic Affairs.
Appendix A. Screen print of a trace illustrating the dynamics of the design process for role refinement