DESIGN OPERATORS TO SUPPORT ORGANISATIONAL DESIGN

CATHOLIJN M JONKER
University of Nijmegen, The Netherlands

ALEXEI SHARPANSKYKH, JAN TREUR
Vrije Universiteit Amsterdam, The Netherlands

and

PINAR YOLUM
Bogazici University, Turkey

Abstract. Organisational design is an important topic in the literature on organisations. 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 organisations. These operators help an organisation designer create an organisation design from scratch as well as offer the possibility to revise existing designs of organisations. 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 organisation design process. The usability of the design operators is demonstrated in a running example. This is demonstrated by an implemented prototype example tool.

1. Introduction

Organisations play a key role in the modern society. The welfare of the society as a whole depends upon the effectiveness, efficiency and viability of organisations. Organisational structures and processes are studied in social sciences, where organisational design is a special topic. Organisation design is concerned "with what an organisation is ought to be" (Pfeffer 1978). More specifically, Galbaith (1978) stated that organisation design "is conceived to be a decision process to bring about a coherence between the goals or purposes for which the organisation exists, the patterns of division of labor and interunit coordination and the people who will do the work". Further

© 2006 *Springer. Printed in the Netherlands.*

Galbaith argues that design is an essential process for "creating organisations, which perform better than those, which arise naturally".

In literature, a range of theories and guidelines concerning the design of organisations are present (Galbraith 1978; Duncan 1979; Minzberg 1993; Blau and Schoenherr 1971). However, despite the abundance of organisational design theories no general principles applicable to organisational design in all times and places can be identified (Scott 1998). Moreover, almost all theoretical findings in organisational design are informal and often vague. In order to provide an organisation designer or a manager with operational automated tools for creating, analyzing, and revising organisations, in the first place a formal representation of an organisation 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, at the various points in time, choosing a 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.

In this paper we introduce a formal organisational model format, to be used to represent design object descriptions. On top of this, a set of design operators is formally defined. The formalisation is based on an extension of predicate logic (Huth and Ryan 2004).

Often in the literature organisational design is recognized as an engineering problem (Child 1973). From this perspective design is considered as a continuous process of a gradual change of an organisational model by applying certain operations (Pfeffer 1978). For example, Minzberg (1993) describes design process as the following sequence of operations: given overall organisational 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 organisation hierarchy is complete.

For this paper we aimed at identifying the most commonly and generally used set of operators for designing organisations. For this purpose the literature from social sciences, and design principles used in other disciplines were investigated. Useful principles for organisational design can be found in the area of derivative grammars. Thus, graphical changes in organisational designs may be described by shape (Stiny 1991) and graph grammars (Rozenberg 1997). Whereas changes in textual (or symbolic) structural and dynamic descriptions of organisational elements may be specified by string (Chomsky 1965) and graph grammars, which allow representation of relationships between descriptions of different elements. In order to relate graphical organisational designs to designs described in a symbolic form, parallel grammars (or grammars defined in multiple algebras) may be used (Stiny 1991). For designing organisation structures with multiple levels of representation (e.g., hierarchical organisations with departments, groups, sections) abstraction grammars (Schmidt and Cagan 1995) and hierarchical graph grammars (Habel and Hoffmann 2004) 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 (Schmidt and Cagan 1995)).

Thus, based on the rich literature on design, this paper makes a first attempt to formalize the operators underlying organisation design processes. A set of design operators is formally introduced, which provides the means for creating a design of an organisation from scratch as well as revising existing designs for organisations.

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

2. Format for an Organisational Model as a Design Object Description

We consider a generic organisation model, abstracted from the specific instances of agents (actors), which consists only of descriptions of organisational roles and relations between them.

Definition 1 (Organisation)
A specification of an organisation with the name O is described by the relation is_org_described_by(O, Γ, Δ), where Γ is a structural description and Δ is a description of dynamics.

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

Definition 2 (Organisation Structure)

A structural description Γ of an organisational specification described by the relation is_org_described_by(O, Γ , Δ) is determined by a set of relations, among which[1]:

- a relation has_basic_components(Γ, R, G, IL, ILL, ONT, M, ENV) defined on the subsets R, G, IL, ILL, ONT, M, ENV of the corresponding general sets ROLE (the set of all possible role names), GROUP (the set of all possible group names), INTERACTION_LINK (the set of all possible interaction links names), INTERLEVEL_LINK (the set of all possible interlevel links names), ONTOLOGY (the set of all possible ontology names), ONTO_MAPPING (the set of all possible ontology mappings names), ENVIRONMENT (the set of all possible environment names)2
- a relation for specifying a role r∈R in Γ is_role_in(r, Γ)
- a relation for specifying an interaction link e∈IL in Γ is_interaction_link_in (e, Γ)
- a relation for specifying an interlevel link il∈ILL in Γ is_interlevel_link_in(il, Γ)
- a relation for specifying an environment env∈ENV is_environment_in(env, ENV)
- a relation has_input_ontology(r, o) that assigns an input ontology o∈ONT to a role r∈R (similarly the relations for output, internal, and interaction ontologies are introduced: has_output_ontology(r, o), has_interaction_ontology(r, o), has_internal_ontology(r, o))
- a relation has_input_ontology(env, o) that assigns an input ontology o∈ONT to an environment env∈ENV (similarly the relations for output, internal, and interaction ontologies are introduced: has_output_ontology(env, o), has_interaction_ontology(env, o), has_internal_ontology(env, o))
- a relation is_ontology_for(el, o) that assigns an ontology o∈ONT either to a role el∈ R or an environment el∈ ENV
- a relation has_onto_mapping(il, m) that associates an interlevel link il∈IL with an ontology mapping m∈M (an ontology mapping for an interaction link is defined similarly)
- a relation is_interaction_link_of_type(e, type) that specifies an interaction link e∈IL of one of the types: role_interaction_link, env_input_link, env_output_link

[1] 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.

[2] The difference between R and ROLE, for example, is that R (subset of ROLE) is the set of all role names that occur in Γ.

- a relation connects_to(e, r, r', Γ) that specifies a connection by an interaction link $e \in IL$ from a source-role $r \in R$ to a destination role $r' \in R$ in Γ
- a relation connects_to(e, env, r, Γ) that specifies a connection by an interaction link $e \in IL$ of type env_output_link from an environment $env \in ENV$ to a role $r \in R$ in Γ (similarly for connects_to(e, r, env, Γ))
- a relation subrole_of_in(r', r, Γ) that specifies a subrole $r' \in R$ of a role $r \in R$ in Γ
- a relation member_of_in(r ,g ,Γ) that specifies a member role $r \in R$ of a group $g \in G$ in Γ
- a relation interlevel_connection(il, r, r', Γ) that specifies a connection by an interlevel link $il \in ILL$ between roles r, $r' \in R$ of adjacent aggregation levels

Organisational behavior is described by dynamic properties of the organisational structure elements.

Definition 3 (Organisation Dynamics)
A description of dynamics Δ of an organisational specification described by the relation is_org_described_by(O, Γ , Δ) is determined by a set of relations, among which:

- a relation has_basic_components(Δ, DP) that specifies a set of dynamic properties names DP defined in an organisation model
- a relation has_dynamic_property(r, d) that specifies a dynamic property $d \in DP$ 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: has_dynamic_property(e, d), has_dynamic_property(g, d), has_dynamic_property(env, d))
- a relation has_expression(d, expr) that identifies a dynamic property name $d \in DP$ with a dynamic property expression $expr \in DPEXPR$ (e.g., a formula in sorted first-order predicate logic)

A role is a basic structural element of an organisation. It represents a subset of functionalities, performed by an organisation, 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.

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 organisational 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:

- or, oi, o, o', o''∈ONT, or= o ∪ o' ∪ o'', oi= o' ∪ o'', here ∪ is a functional symbol that maps names of ontologies to a name of the joint ontology
- Relations:
- has_internal_ontology(r, o), has_input_ontology(r, o'), and has_output_ontology(r, o'')
- has_ontology(r, or) and has_interaction_ontology(r, oi)
- d∈DP, 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 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'.

Roles of the same aggregation level interact with each other by means of interaction links. The interaction between roles is restricted to communication acts.

Definition 5 (Interaction link)
An interaction link e is determined by:
Relations:

- is_interaction_link_in(e, Γ)
- has_onto_mapping(e, m) for some m∈M
- has_dynamic_property(e, d) for a number of d∈DP
- Constraints:
- An interaction link e should connect two roles at the same aggregation level: is_interaction_link_in(e, Γ) ⇒ ∃r, r'∈R connects_to(e, r, r', Γ) ∧ ¬has_subrole(r, r') ∧ ¬has_subrole(r', r)

An interlevel link connects a composite role with one of its subroles. It represents an information transition between two adjacent aggregation levels. It may describe an ontology mapping for representing mechanisms of information abstraction. For example, consider a situation, in which only a (abstracted) part of information communicated within a certain composite role should be made available as output from this role.

Definition 6 (Interlevel link)
A specification for an interlevel link il is determined by:
Relations:

- is_interlevel_link_in(il, Γ)

- has_onto_mapping(il, m) for some $m \in M$
- Constraints:
- An interlevel link il should connect two roles at two adjacent aggregation levels: is_interlevel_link_in(il, Γ) $\Rightarrow \exists$r, r'$\in$R subrole_of_in(r', r, Γ) $\wedge$ (interlevel_connection(il, r, r', Γ) $\vee$ interlevel_connection(il, r', r, Γ))

A group is a composite structural element of an organisation that consists of a number of roles. In contrast to roles a group does not have well-defined input and output interfaces. Groups can be used for modeling units of organic organisations, which are characterized by loosely defined or sometimes informal frequently changing structures that operate in a dynamic environment. Furthermore, groups can be used at the intermediate design steps for identifying a collection of roles, which may be further transformed into a composite role.

Definition 7 (Group)
A group g is defined by the relations to other concepts:

- membership relation member_of_in: r$\in$R member_of_in(r, g, Γ)
- has_dynamic_property(g, d) for a number of d$\in$DP

The conceptualized environment represents a special component of an organisation model. According to some sociological theories (e.g., contingency theory), an environment represents a key determinant in organisational design, upon which an organisational 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 organisation. The interfaces are conceptualized by the environment interaction (input and output) ontologies. Interaction links between roles and the environment are indicated in the organisational model as ones that have a specific type, namely env_input_link or env_output_link by means of the predicate is_interaction_link_of_type. Roles interact with the environment by initiating observations and obtaining observation results, and performing actions that can change a state of the environment.

The behavior of each element of an organisational 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. An example of the dynamic property expression will be given in Section 3.1.

The application of the basic components of an organisational model is illustrated by means of a running example. Consider the process of organizing a conference. A partial model for the considered conference organisation is shown in Figure 1.

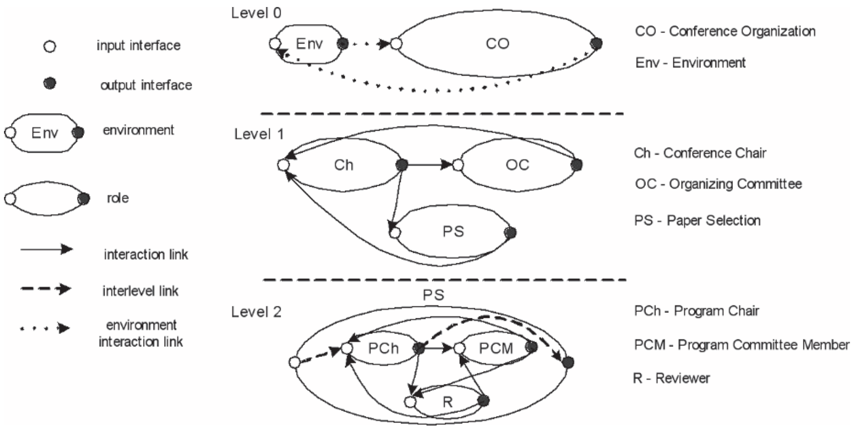

Figure 1. Model of the conference organizing committee.

At the most abstract level 0 the organisation is specified by one role CO (Conference Organisation) 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 organisational 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 organisational 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 (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.

3. Representing Design Operators for Organisational Design

In this section a formal format to represent design operators and based on this format representations are introduces for a number of primitive design operators for designing organisations. Each primitive operator represents a specialized one-step operator to transform a design object description (organisational model) into a next one. The parts of the organisation O that are being modified in terms of structure and dynamics (i.e., sets of dynamic properties) are specified using the in-focus relations: structure_in_focus(O, Rf, Gf, ILf, ILLf, ONTf, Mf, ENVf) and dynamics_in_focus(O, DPf), with $Rf \subseteq R$, $Gf \subseteq G$, $ILf \subseteq IL$, $ILLf \subseteq ILL$, $ONTf \subseteq ONT$, $Mf \subseteq M$, $ENVf \subseteq ENV$, $DPf \subseteq DP$. The remaining parts of the organisation stay the same.

The following operations all refer to an organisation $O \in ORGANISATION$ described by relations is_org_described_by(O, Γ, Δ), has_basic_components(Γ, R, G, IL, ILL, ONT, M, ENV). This organisation is modified by an operator, leading to a second organisation $O' \in ORGANISATION$ described by relations is_org_described_by(O', Γ', Δ'), has_basic_components(Γ', R', G', IL', ILL', ONT', M', ENV').

Our choice of primitive operators is motivated by different design guidelines and theories from social sciences (Galbraith 1978; Blau and Schoenherr 1971; Lorsch and Lawrence 1970), other disciplines, and our own research on formal modeling of organisations (Broek et al. 2005). 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 organisations. The operators are divided into three classes, which are consecutively described in the following subsections. Thus, in Section 3.1 the operators for creating and modifying roles are specified; in Section 3.2 the operators for introducing and modifying different types of links are described; and in Section 3.3 the operators for composing and modifying 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 organisation are shown in Table 1.

TABLE 1. Operator classes for creating and modifying roles.

CLASS	DESCRIPTION
Role Introduction	Introduces a new role
Role Retraction	Deletes all links, connected to a role with their dynamic properties and mappings; deletes a role and all dynamic properties, associated with this role
Role Dynamic Property Addition	Adds a new dynamic property to a role
Role Dynamic Property Revocation	Deletes an existing role dynamic property

A *role introduction operator* adds a new role to the organisation. Usually, in organisational design after organisational tasks have been identified, these tasks should be further combined into positions (roles), based on the labor division principles (Kilbridge and Wester 1966).

Role introduction operator

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

1. $\delta \notin R, \delta \in R'$ such that is_role_in(δ, Γ')
2. structure_in_focus(O,$\emptyset$,$\emptyset$,$\emptyset$,$\emptyset$,$\emptyset$,$\emptyset$,$\emptyset$)
3. structure_in_focus(O', {δ}, $\emptyset$,$\emptyset$,$\emptyset$, ONTf', $\emptyset$, $\emptyset$), where ONTf'= is_ontology_for(o, δ) and $o \in$ ONT'

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 organisation, when the Reviewer Recruiter has found enough reviewers, then the role can safely be removed from the organisation.

A formal representation for the role retraction operator has been left out due to the limited space and can be found in Jonker et al. (2005).

A *role dynamic property addition operator* creates a new property for the existing role in the organisation. For example, a role property that may be added to role Reviewer (R) expresses that a reviewer should send her review to the Program Chair before a certain deadline. This property can be formalized using the Temporal Trace Language (TTL) (Jonker and Treur 2003a), which is a variant of an order-sorted predicate logic with facilities for reasoning about the dynamics properties of a system. Thus, the dynamic part of the organisational model is changed by adding the following dynamic property for role R:

$\forall$t state(γ, t) |= deadline_for_conference(d)$\Longrightarrow \exists$t' < d state(γ, t', output(Reviewer)) |= communicated(send_from_to(Reviewer, Program_Chair, review_report))

A *role dynamic property revocation operator* deletes a property from the dynamic description of a role.

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 organisation, Table 2.

TABLE 2. Operator classes for creating and modifying links.

CLASS	DESCRIPTION
Interaction Link Addition	Adds a new interaction link between any two roles
Interaction Link Deletion	Deletes an interaction link and all dynamic properties, associated with this link
Interlevel Link Introduction	Introduces a new interlevel link
Interlevel Link Retraction	Retracts an existing interlevel link
Interaction Dynamic Property Addition	Adds a new dynamic property to an interaction link
Interaction Dynamic Property Revocation	Deletes an existing dynamic property, associated with an interaction link

An *interaction link addition operator* allows the creation of an interaction link (information channel) between two existing roles in the organisation. In the organisational design after organisational 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 an interaction link between the appropriate reviewer roles in the design object description for an organisation.

Interaction link addition operator
Let op(O, O', δ) be an operator that changes O into O' with a focus on δ. Then op is an interaction link addition operator iff it satisfies:

1. $\delta \notin IL, \delta \in IL'$ such that is_interaction_link_in(δ, Γ')
2. structure_in_focus(O, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$, $\emptyset$)
3. structure_in_focus(O', $\emptyset$, $\emptyset$, {δ}, $\emptyset$, $\emptyset$, Mf', $\emptyset$)
   Mf'= {m$\in$M'| has_onto_mapping(δ, m)}

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 any more. This case can be handled by the deletion of the interaction between two roles in the design object description for an organisation.

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

An interlevel link creates a relation between a composite role and its subroles. It allows information that is generated outside the role, to be passed into the role through its input interface or it allows information, generated within a role to be transmitted outside through the role output interface. Normally, in hierarchical (mechanical) organisations decisions made at a managerial level are transferred to an operational level, e.g, to a certain department. Within the department this information is obtained by a certain role(s). For identifying, which roles obtain this information interlevel links are used. In the conference management example, the Conference Chair may have the possibility to send inquiries to Program Committee Members. This can be achieved by introduction of an interlevel link between composite role Paper Selection (with which role Conference Chair has a direct connection by an interaction link) and its subrole Program Committee Member. An *interlevel link introduction operator* allows addition of such a link into a role.

Interlevel link introduction operator
Let op(O, O', δ) be an operator that changes O into O' with a focus on δ. Then op is an interlevel link introduction operator iff it satisfies:

1. $\delta \notin$ IL, $\delta \in$ IL' such that is_interaction_link_in(δ, Γ')
2. structure_in_focus(O,Ø,Ø,Ø,Ø,Ø,Ø,Ø)
3. structure_in_focus(O',Ø,Ø,{δ},Ø,Ø,Mf',Ø)
   Mf'= {m$\in$M'| has_onto_mapping(δ, m)}

An *interlevel link retraction operator* is used for breaking off interaction between some composite role and one of its subroles. This operation removes an interlevel link from the design object description for an organisation. If the Conference Chair does not need to communicate with Program Committee Members any more, the interlevel link between these two roles can be retracted.

3.3. OPERATORS FOR GROUPS

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

Often an organisation designer can easily list a number of roles needed in an organisation. However, it is not always clear, which roles are related to each other; which roles would most often interact with each other, and so on.

TABLE 3. Operator classes for creating and modifying groups.

CLASS	DESCRIPTION
Grouping	Combines roles into groups
Degrouping	Moves roles outside of a group and deletes the group
Group-to-Role Transformation	Transforms groups into roles
Role-to-Group Transformation	Transforms roles into groups

In the literature on organisational design (Minzberg 1993) different principles of grouping are described. For example, role grouping can be performed based on (1) similarities in role functional descriptions; (2) role participation in the same technological process; (3) identity or similarity of role technical specialties; (4) role orientation on the same market or customer groups. Often roles belonging to the same group interact with each other intensively. However, in the proposed organisational model in contrast to roles, groups do not have interfaces. It means that every role within a group is allowed to interact with roles outside the group by means of direct interaction links. A group can be transformed into a role, a more coherent, integrated and formal organisational unit with proper interfaces (e.g., a department of an organisation). For example, in the conference organisation the Program Chair and the Program Committee Members can be joined in one Program Committee group that will be responsible for making final decisions concerning paper acceptance. This can be accomplished by applying the grouping operator.

Grouping operator

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

Structural aspect:

1. ∀a∈Rg: member_of_in(a, Gn, Γ').
2. structure_in_focus(O,∅ ,∅ ,∅ ,∅ ,∅ ,∅ ,∅)
3. structure_in_focus(O', ∅, {Gn}, ∅ ,∅ ,∅ ,∅ ,∅)

Dynamic aspect:

1. dynamics_in_focus(O, ∅)
2. dynamics_in_focus(O', DPf')
   DPf'={dp∈DP'| has_dynamic_property(Gn, dp) }.
3. Er={e∈IL| ∃r1∈Rg ∃r2∈Rg connects_to(e, r1, r2, Γ)}
   DPr={dp∈DP|∃r∈Rg has_dynamic_property(r, dp) ∨ ∃e∈Er has_dynamic_property(e, dp)}
   DPg={dp∈DP'| has_dynamic_property(Gn, dp)}
4. DPg⊆DCL(DPr), where DCL(DPr) is the deductive closure of DPr

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.

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 organisation with loosely defined frequently changing structure this would correspond to the formalisation of one of the organisational units, i.e., providing a formal (permanent) structural description with the subsequent specifying formal functional procedures. For example, in the conference organisation setting Program Committee group from the Paper Selection role can be further transformed into Program Committee role, a formal organisational unit with certain characteristics and functions (e.g., final decision making for the paper acceptance). In this case reviewers should follow a formal procedure for interactions with Program Committee role and cannot directly address any arbitrary Program Committee member. Such transformation can be achieved by means of Group-to-Role operator.

Group-to-Role operator

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

Structural aspect:

1. r∉R, g∉G'
2. ∀a∈R: member_of_in(a,g, Γ)⟹subrole_of_in(a, r, Γ').
3. structure_in_focus(O,∅ , {g}, ∅,∅ ,∅ ,∅ , ∅)
4. structure_in_focus(O', {r}, ∅ ,∅ ,∅ , ONTf',∅ , ∅)
   ONTf'={o∈ONT'| has_internal_ontology(r, o) ∨ has_input_ontology(r, o) ∨ has_output_ontology(r, o)}

Dynamic aspect:

1. dynamics_in_focus(O, DPf)
   DPf={dp$\in$DP| has_dynamic_property(g, dp)}.
2. dynamics_in_focus(O', DPf')
   DPf'={dp$\in$DP'| has_dynamic_property(r, dp)}.
3. DP(g)$\Longrightarrow$DP(r)

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 reorganisation some formal organisation units (e.g., groups, 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 organisational formations.

4. Composing Operators

The described above primitive operators reflect major principles of organisational design. In practice next to the primitive operators more complex operators are used. Complex operators are represented as a combination 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 organisation 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 organisational 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 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 (Kilbridge and Wester 1966).

Let us illustrate the application of Role Refinement operator in the context of the conference organizing example. In Figure 2 the design object description for an organisation is represented at the first aggregation level. The symbol * denotes that an operator can be applied zero, one or multiple times.

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, role OC is refined into two newer roles LOC and GOC. These roles are able to interact with each other and with role Chair.

TABLE 4. Sample complex operators for creating and manipulating organisations.

NAME	PATTERN FOR	DESCRIPTION
Interaction Level Ascent	Interaction link deletion*. Role interaction dynamic property addition*. Interlevel link addition*. Interaction link addition*.	Represents interaction between roles at a higher aggregation level
Role refinement	Role Retraction. Interlevel link deletion*. Interaction link deletion*. Interaction dynamic property addition*. Interlevel link addition*. Interaction link introduction*. Role dynamic property addition*. Role introduction*	Divides a role into several roles such that the role properties of the first role are distributed over the newer roles
Role join	Role Retraction*. Interlevel link deletion*. Interaction link deletion*. Interaction dynamic property addition*. Interlevel link addition*. Interaction link introduction*. Role dynamic property addition*. Role introduction	Joins several roles into a single role
Adding aggregation levels	Interaction Level Ascent. G-t-R. Role grouping. Role refinement*	Aggregates existing roles of the organisation in more complex roles

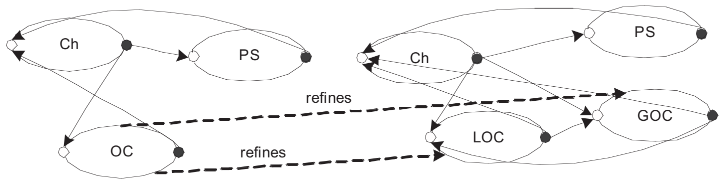

Figure 2. Example of Role refinement operator application, in which the Organizing Committee role (OC) is refined into the Local Organizing Committee (LOC) and General Organizing Committee roles (GOC).

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 organisation before and after the application of a complex operator; therefore, they can serve for the purposes of checking integrity and consistency of a design object description.

A natural dual to the role refinement is role joining. This operator takes several roles and joins them into a single role. Consider again the organisation arranging a conference. If over time the differences between the tasks of the Program Committee Member and Reviewer roles disappear, then the roles Program Committee Member and Reviewer can be joined in one role.

Let us consider one more often used complex operator *Adding Aggregation Levels*. When certain roles have been joined in one group, this operator allows representing this group as an integral structural unit of an organisation at the more abstract aggregation level. This operator has a counterpart in organisational design studies called *departmentalization*. Based on the departmentalization principles (cf. Galbraith 1978) an organisation is partitioned into structural units (called departments) with certain areas of responsibilities, a functional orientation, and a local authority power.

In the conference organisation Adding Aggregation Levels operator can be applied for representing the Program Committee as an integral role that consists of the Program Chair and the Program Committee Member roles within Paper Selection role. Such choice, for example, can be motivated by introducing a general formal procedure for paper acceptance. Hence, the Program Committee role is empowered (has a corresponding dynamic property) to make final decisions concerning paper selection. Adding Aggregation Levels operator for this example can be considered as three-step process (see Figure 3 for the representation of the organisation model (role Paper Selection) at the second aggregation level).

First, roles Program Chair (PCh) and Program Committee Member (PCM) are joined into one group by application of Grouping operator. After that, at step 2 by means of the Group-to-Role operator the created group is transformed into role Program Committee by adding interaction interfaces. Finally, as the last step using Interaction Level Ascent operator interaction links between roles PC and Reviewer (R) are created, as well as interlevel links within role PC.

5. A Prototype Tool to Support the Design of Organisations

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

The proposed formalism accurately distinguishes different types of organisation entities with their objects, relations and constraints, which can be naturally represented as classes with members and methods in object-oriented programming (OOP) languages. Furthermore, the identified relationships among organisation entities may be fully captured by the fundamental OOP mechanisms (e.g., inheritance, interfaces and inner classes). The design operators can be programmed as transformation functions with explicitly defined arguments, conditions and effects of their application. Moreover, most of the introduced formal concepts are based on the notions from organisation theories, which will facilitate use of a tool by organisation modelers.

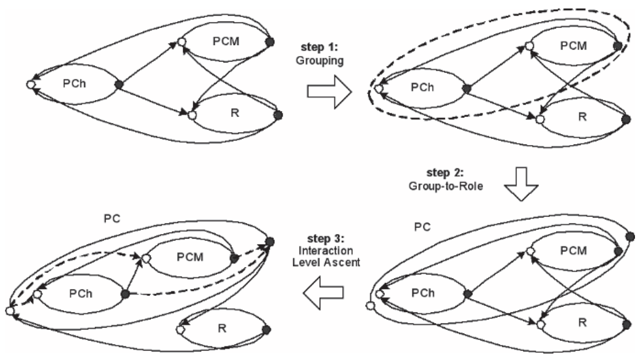

Figure 3. Example of Adding Aggregation Levels operator application, in which the roles Program Chair (PCh) and Program Committee Member (PCM) are grouped together and transformed into the Paper Selection (PC) role.

For the purpose of illustration and evaluation a prototype tool was implemented. This tool supports organisational design and allows investigating its dynamics. 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, which is graphically illustrated by a partial trace taken from the tool in Figure 4.

In the design process, first, a designer chooses a part of the design 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. Refinement is a composite 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 Refinement); 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 Organizing Committee (LOC) and General Organizing Committee (GOC)) and their ontologies.

After that the software tool requests the designer to specify dynamic properties for the created roles. The designer may postpone this operation to a future time point. Thereafter, the tool proposes to add interaction links between roles LOC, GOC and role Chair (Ch), with which the original role OC was connected. After that dynamic properties for the introduced interaction links may be added. As the last step role OC and interaction links connecting it with role Ch, as well as OC role and interaction links dynamic properties are automatically removed from the design object description.

TABLE 5. Dynamics of the design process for the role refinement.

ACTIONS OF THE DESIGNER	STATES OF THE TOOL
Chooses to address the role Organizing Committee (OC)	Proposes potentially applicable operators for role OC
Chooses the role refinement operator	According to the specification of the role refinement operator, initiates execution of role introduction operator and requests the designer to specify role names
Specifies GOC (General Organizing Committee) and LOC (Local Organizing Committee) names of the roles, into which role OC is refined	Requests to specify the elements of the ontologies for the newly created roles
Specifies the elements of the ontologies for roles LOC and GOC	Initiates execution of the role dynamic property addition operator. Requests to specify dynamic properties for LOC and GOC roles
(optional) Specifies dynamic properties for the roles	Initiates execution of the interaction link introduction operator. Requests to specify interaction links between roles Chair (Ch), LOC and GOC
Specifies, which interaction links are needed between the roles	Initiates execution of the interaction dynamic property addition operator. Requests to specify dynamic properties for the introduced interaction links
(optional) Specifies dynamic properties for the interaction links	Initiates execution of the interaction link deletion operator, which removes all interaction links connected with role OC. Then, initiates execution of the role retraction operator, which removes role OC from the design object description

6. Discussion

This paper introduces a representation format and a variety of operators for the design of organisations specified in this representation format. The described operators have several important characteristics. First, they can be combined into composite operators that can serve as patterns for larger design steps in certain design cases. Second, the identified set of operators is independent of any organisation theory or sociological methodology: they can be used for formalizing design principles from different theories. Third, a designer has freedom to choose any sequence of operators for creating designs of organisations of most types (e.g., functional and organic). The operators offer both top-down refinements, as well as bottom-up grouping options. Finally, as has been shown the developed tool provides interactive support in designing organisations. In the future a graphical interface for representing design objects in the developed tool will be developed.

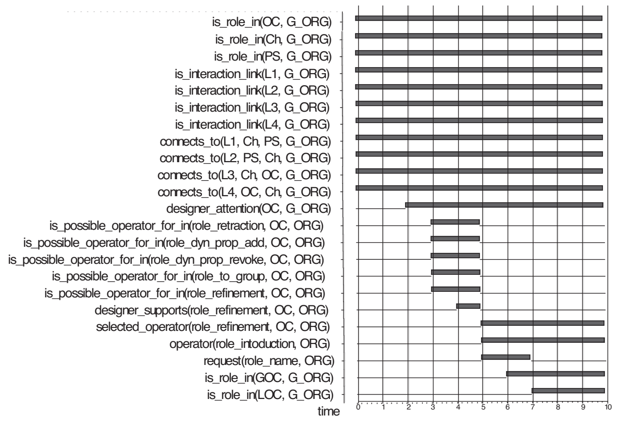

Figure 4. Screen print of a trace illustrating dynamics of the design process for the role refinement.

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 (He et al. 2005). These patterns specify general-purpose manipulations with programming constructs (e.g., interface and private methods of components); while in organisational design literature organisation transformations are described using domain-specific concepts. The formal representation format proposed in this paper bridges this gap and facilitates the abstraction of organisation 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 organisation. The verification of structural consistency is based on the consistency definitions for operators (Jonker et al. 2005). For verifying dynamic consistency model checking techniques (McMillan 1993) may be used, which will be further investigated in the future. Furthermore, verification mechanisms based on certain requirements on organisational functioning and performance (e.g., using organisation performance indicators) represent a subject of our future research.

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

Acknowledgements

The authors wish to thank the anonymous reviewers for their useful comments. Their comments also found their way into the technical report upon which this paper is based.

References

Blau, PM and Schoenherr, RA: 1971, *The Structure of Organisations*, Basic Books Inc., New York London.

Broek, E, Jonker, C, Sharpanskykh, A, Treur, J, and Yolum, P: 2005, Formal modeling and analysis of organisations, *in* O Boissier, V Dignum, E Matson, J Sichman (eds), *Proceedings of the Workshop on Organisations in Multi-Agent Systems*, pp. 17-33.

Child, J: 1973, *Organisation: A Choice for Man*, *in* J Child (ed), *Man and Organisation*, Halsted Press, London, pp. 234-570.

Chomsky, N: 1965, *Aspects of the Theory of Syntax*, The MIT Press.

Duncan, RB: 1979, What is the right organisation Structure? *Organisational Dynamics*, Winter, pp. 59-79.

Galbraith, JR: 1978: *Organisation Design*, Addison-Wesley Publishing Company, London Amsterdam Sydney.

Habel, A and Hoffmann, B: 2004, Parallel independence in hierarchical graph transformation, *International Conference on Graph Transformation*, LNCS, Springer-Verlag, Heidelberg **3256**: 178-193.

He, J, Li, X, and Liu, Z: 2005, Component-based software engineering, *in* DV Hung, M Wirsing (eds), *Theoretical Aspects of Computing*, LNCS, Springer **3722**: 70-95.

Huth, M and Ryan, MD: 2004, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press.

Jonker, CM, Treur J: 2003, A temporal-interactivist perspective on the dynamics of mental states, *Cognitive Systems Research Journal* **4**(3): 137-155.

Jonker, CM, Sharpanskykh, A, Treur, J, and Yolum, P: 2005, *Operators for Formal Modeling of Organisations*, Technical report 06-01AI, Vrije Universiteit, Amsterdam.

Kilbridge, M and Wester, L: 1966, An economic model for the division of labor, *Management Science* **12**(6): 255-269.

Lorsch, JW and Lawrence, PR: 1970, *Organisation Design*, D Richard (ed), Irwin Inc, USA.

McMillan, K: 1993, *Symbolic Model Checking*, Kluwer Academic Publishers.

Mintzberg, H: 1993, *Structure in Fives: Designing Effective Organisations*, Prentice-Hall, NJ.

Pfeffer, J: 1978, *Organisational Design*, AHM Publishing Corp., Illinois, USA.

Rozenberg, G (ed): 1997, *Handbook of Graph Grammars and Computing by Graph Transformation*, **1**: Foundations, World Scientific.

Schmidt, LC and Cagan, J: 1995, Recursive annealing: A computational model for machine design, *Research in Engineering Design* **7**: 102-125.

Scott, WR: 1998, *Organisations: Rational, Natural and Open Systems*, Prentice Hall, USA.

Stiny, G: 1991, The algebras of design, *Research in Engineering Design* **2**: 171-181.

Wijngaards, N: 1999, *Re-design of Compositional Systems*, PhD Thesis, SIKS dissertation Series, 99-6, Vrije Universiteit Amsterdam.