Cross-Organizational and Context-Sensitive Modeling of Organizational Dependencies in $\mathcal{C} - \mathcal{ORG}$

Alexander Lawall, Thomas Schaller, and Dominik Reichelt

Institute for Information Systems at Hof University, Alfons-Goppel-Platz 1, 95028 Hof, Germany
{alexander.lawall,thomas.schaller,dominik.reichelt}@iisys.de

Abstract. Almost every application used to run a business relies on a model of the organization structure, the roles and the actors in order to define access rights or assign tasks. This article proposes a novel approach to organizational modeling. It also describes a way to connect internal and external organizational models in order to implement cross-organizational processes. It demonstrates the approach on two examples. One of them is a cross-organizational business process and the other a joint research project. The paper includes a description of the metamodel that constitutes the approach for context-sensitive modeling. It shows concrete language expressions that describe sets of actors and how these expressions are interpreted on the organizational model. The article concludes with a short overview of a prototypical implementation of the system $\mathcal{C} - \mathcal{ORG}$.

Keywords: cross-organizational, inter-organizational, context-specific, metamodel, organizational model, formal language.

1 Introduction

For years, companies have been facing an increasing complexity of their environment. In order to be viable (cf. [1]), their organization structure has to be flexible enough to react to those changes (cf. [2]). Otherwise they will be eliminated from the market. So it is no big surprise that a lot of companies changed their structure from traditional stereotypes like hierarchies or tensor to more flexible concepts like process or virtual organization forms (cf. [3, p. 277]). Nowadays the organizational structure is driven by the work in project teams, global teams, networks and global teams in networks (cf. [4]). Especially cooperative structures of independent partners like virtual organizations[1] are arising (cf. [4]). They bundle the core competencies of all involved companies over spacial distances [3, p. 278].

[1] A virtual organization has the potential of a traditional organization without having a comparable institutional frame. They go beyond intra- and inter-organizational limitations, cf. [5].

A. Nanopoulos and W. Schmidt (Eds.): S-BPM ONE 2014, LNBIP 170, pp. 89–109, 2014.

Traditional structures for organizing a company like hierarchies, matrices, etc. are partly disused, but still exist in companies (cf. [6]), though new organization paradigms are arising. In addition to these intra-organizational aspects, cooperation paradigms like supply chain management or joined product development make it necessary to pay special attention to inter-organizational (same as cross-organizational) processes and structures ([7, pp. 4]).

Almost every application used to run a business relies on a model of the organization structure, the roles and the actors in order to define access rights or assign tasks. This article proposes a novel approach to organizational modeling. It focuses on representing cross-organizational interactions and context-specific organizational relations.

In order to consistently demonstrate key issues, section 2 introduces a practical cross-organizational business process. We consider this process from the perspectives of the different involved parties, but with a strong emphasis on cross-organizational communication.

Chapter 3 then introduces extensions to an existing approach described in [8]. It proposes new entity types and sets of relationship types for describing cooperations. After that the foundations of the metamodel and a corresponding query language is presented.

The following chapter illustrates the proposed concepts by applying them to the example described in section 2. It also shows how temporary cooperations, e.g. projects, can be represented within the frame of the metamodel.

In order to answer questions to concrete organizational models (e.g. the definition of actors in workflow management systems), different algorithms are used to traverse the model. Section 5 describes these algorithms, especially with regard to the novel concepts introduced in section 3.

We conclude with the presentation of a prototypical implementation (section 6) and by giving an outlook on topics for future research (section 7).

The appendix contains a more detailed and limited view on aspects discussed in section 2.

2 Motivating Example

This section describes a cross-organizational business process – a purchase. First, we consider the process stakeholders from an abstract perspective. In the following, we focus on the different internal *subjects*, i.e. behaviors as shown by the process stakeholders (cf. [9]). For clarity purposes, we omit parts of the individual subjects' behavior and concentrate on cross-organizational interactions.

Figure 1 depicts the interaction of the involved subjects on a high level of abstraction.

We now consider the subjects described in fig. 1 in more detail. Figures 8 (cf. appendix A) and 2a are representations of internal subjects within the *customer* subject.

The start subject of the whole process is depicted in figure 8. An employee of the university (the process *initiator*) decides that they need to purchase an

Fig. 1. Information flow between subjects

article. They fill out a request for purchase, which is then reviewed by their *supervisor*. If the supervisor approves the request, the initiator forwards it to the *purchase department*. The employees of this department handle the actual purchase order process, and after a while, the process initiator receives the package and a delivery receipt from the *distributor*. They then validate the receipt and the delivery. They also forward the receipt to the purchase department as signal to finalize the order process (omitted in the figure).

The following subjects are of relevancy:

- Initiator
- Supervisor
- Purchase Department
- Distributor

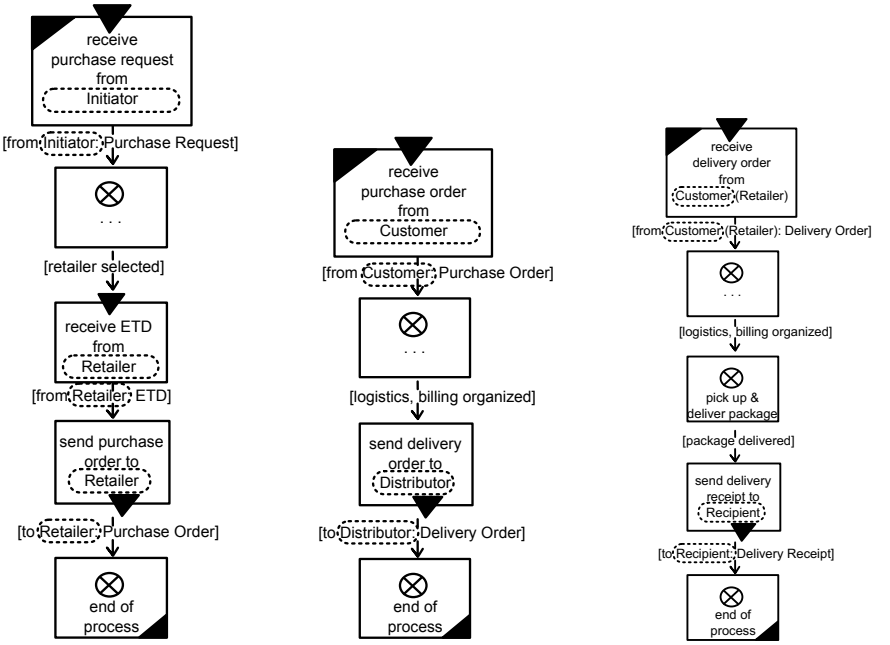
Fig. 2a shows the actions that need to be taken by the subject *purchase department*, once they receive a purchase request from the process *initiator*. In the depiction, we focus on interactions with other subjects that are described in our context. Missing steps, indicated by "...", include the actual comparison of retailers and internal accounting affairs. A purchase might be time-critical, so one criterion for the selection of a retailer can be the request for an estimated time of delivery (ETD). This ETD is received by the *purchase department*'s clerk.

After selecting a *retailer*, the clerk sends the concrete purchase order to them. This concludes this limited view of the purchase department's role in the overall process. From this perspective, only the Retailer is a relevant new subject.

At this point, the subjects contained in the *customer* subject in figure 1 are described adequately. We continue by describing the purchase processing enacted by the *retailer* (cf. fig. 2b).

The reception of a purchase order from the *customer* triggers the actual handling of the purchase. As previously, we focus on cross-organizational interactions. Omitted process steps include organizing the logistics, billing and actual handling of the ordered products. In the final step of this process, the delivery order is sent to the *distributor*, indicating that the package is ready to be delivered to the customer.

Figure 2c picks up at this point from the distributor's perspective. Please keep in mind that here, the subject *customer* does not denote the same subject as in fig. 1. This is a result of the shifted perspective. The *customer* of the distributor is actually the *retailer* that sends them the delivery order. After receiving the delivery order from their *customer*, the distributor organizes their



(a) Customer: purchase department

(b) Retailer: purchase processing

(c) Distributor: delivery order processing

Fig. 2. Internal subjects of the involved organizations

billing and logistics for the delivery. Again, these internal steps are omitted in the depiction. At some point, an employee actually picks up the package from the sender and delivers it. Additionally, they hand the delivery receipt to the *recipient*. As we omit internal steps like handling the recipient countersigning the delivery, the process is at an end here.

The following additional subjects are of relevancy:

- Customer: Actually the *retailer* in the cross-organizational view (fig. 1)
- Recipient: The recipient of the delivery, actually the *customer* in fig. 1

3 Metamodel

This section describes the metamodel for cross-organizational models. It includes entity and relationship types to model organizational requirements. Additionally, the relationship types can be restricted by different kinds of constraints.

The metamodel for modeling organizational requirements consists of sets of entity type sets $\mathcal{V} = \{\mathcal{V}_{internal}, \mathcal{V}_{external}, \mathcal{V}_{cooperation}\}$ and sets of relationships types $\mathcal{R} = \{\mathcal{R}_s, \mathcal{R}_o, \mathcal{R}_u\}$. The formal specification of the basic metamodel is described in [8].

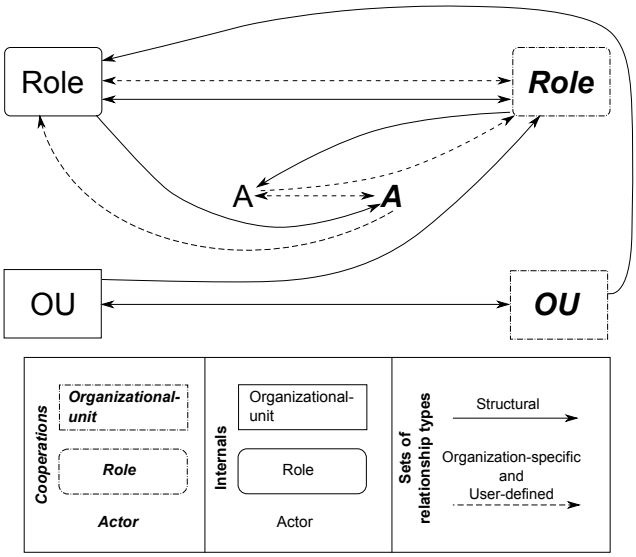


Fig. 3. Excerpt from the metamodel focussing on cooperations and internals

The focus of this paper is the cross-organizational (equals to inter-organizational) context. Cross-organizational means that more than one organization form a cooperation. There is a difference between time-persistent and time-limited (temporary) cooperations. In the example shown in figure 1, the cooperation is a time-persistent cooperation. The *customer*, *retailer* and *distributor* are involved in the process for each concrete purchase.

These requirements have to be met by the metamodel. It has to support modeling both permanent and temporary federations in order to allow interconnecting organizations.

Entity- and Relationship-Types

All entity type sets of $\mathcal{V}$ include the entity types organizational-units $\mathcal{OU}$, roles $\mathcal{ROLE}$ and actors $\mathcal{ACTOR}$. The entity types of $\mathcal{V}_{internal}$ represent the elements of the internal organizational model (cf. [8] and [10]). Entity types that denote externals $\mathcal{V}_{external}$ and cooperations $\mathcal{V}_{cooperation}$ are an extension to the set of entity sets $\mathcal{V}$. The set $\mathcal{V}_{cooperation}$ includes all entity types used to specify cooperations (time-persistent and time-limited). Externals $\mathcal{V}_{external}$ are used to distinguish between internal $\mathcal{V}_{internal}$ (based on the "own" organizational model) and external $\mathcal{V}_{external}$ (all involved organizations, except the "own" one) organizational model entity types. External entity types are needed to interconnect (over cooperation entities) external with internal entities.

The relations between internal and external entities have to be constrained context-specific. This means that they are only valid in a given situation. A more detailed explanation of this concept is given in section 4.

The metamodel consists, besides of entities, of the set $\mathcal{R}$ of relationship type sets. This set can be broken down into *structural* $\mathcal{R}_s$, *organization-specific* $\mathcal{R}_o$ and *user-defined* $\mathcal{R}_u$ relationship type sets.

The *structural* set of a relationship type

$$\mathcal{R}_s = \{\mathcal{HAS}\}$$

acts as an "IS PART OF" relation and is used to relate recursive organizational-units, organizational-units to roles and roles to actors.

The *organization-specific* set of relationship types

$$\mathcal{R}_o = \{\mathcal{HAS\_DEPUTY}, \mathcal{HAS\_SUPERVISOR}, \mathcal{REPORTS\_TO}\}$$

are partitioned in:

- $\mathcal{HAS\_DEPUTY}$ to model deputyship between entities,
- $\mathcal{HAS\_SUPERVISOR}$ to represent the supervisor relationship, and
- $\mathcal{REPORTS\_TO}$ relations that are used to specify the duty of reporting.

All these *organization-specific* relationship types connect role to role, actor to actor or actor to role. The same rules apply between external and between cooperation entities.

The *user-defined* set of relationship types $\mathcal{R}_u$ is freely definable. Relationship types that are needed can be defined and are included in the traversal process. For example, we can define a relationship type $\mathcal{HAS\_DRIVER}$. When a query looking for drivers of an actor is interpreted on the organizational model, the corresponding set of drivers results from evaluating the $\mathcal{HAS\_DRIVER}$ relations. The *user-defined* set of relationship types $\mathcal{R}_u$ allows extending the relationship types as needed.

The aforementioned sets of relationship types $(\mathcal{R}_s, \mathcal{R}_o, \mathcal{R}_u)$ can be used to interconnect the different sets of entity types, such as entity types of internals $\mathcal{V}_{internal}$, externals $\mathcal{V}_{external}$ and cooperations $\mathcal{V}_{cooperation}$. The rules defined above are the same for this interconnection. Figure 3 shows these rules in a graph-based manner. For clarity, the figure only shows rules applicable between different sets of entity types, i.e. cooperations $\mathcal{V}_{cooperation}$ and internals $\mathcal{V}_{internal}$. It omits rules that apply within the same entity type (i.e. between internals). All permutations between the three sets (internals, externals and cooperations) are subject to these rules.

Constraints on Relationship Types

The relations of the two sets of relationship types $\mathcal{R}_o$ and $\mathcal{R}_u$ can be restricted with a set of constraints $\mathcal{C}$[2].

$$\forall \Gamma_{r_1,r_2}(\Gamma_{r_1,r_2} \in \mathcal{R}_o \vee \Gamma_{r_1,r_2} \in \mathcal{R}_u) : \Gamma_{r_1,r_2} \subseteq (\mathcal{ROLE} \times \mathcal{ROLE}) \times \mathcal{C} \quad (1)$$

$$\forall \Gamma_{a_1,a_2}(\Gamma_{a_1,a_2} \in \mathcal{R}_o \vee \Gamma_{a_1,a_2} \in \mathcal{R}_u) : \Gamma_{a_1,a_2} \subseteq (\mathcal{ACTOR} \times \mathcal{ACTOR}) \times \mathcal{C} \quad (2)$$

$$\forall \Gamma_{a,r}(\Gamma_{a,r} \in \mathcal{R}_o \vee \Gamma_{a,r} \in \mathcal{R}_u) : \Gamma_{a,r} \subseteq (\mathcal{ACTOR} \times \mathcal{ROLE}) \times \mathcal{C} \quad (3)$$

$$with\ a, a_1, a_2 \in \mathcal{ACTOR}, r, r_1, r_2 \in \mathcal{ROLE}$$

[2] The set of relationship types $\mathcal{R}_s$ can not be restricted in this fashion.

The set $\mathcal{C}$ includes the empty symbol ε to make constraints optional[3]. The constraints are possible on relations between roles (1), between actors (2) and between actor and role (3). Examples of concrete constraints are shown in figures 4 and 5. Constraints are assigned to relations and reduce the solution space when traversing relations. Traversal takes place when evaluating a query (language expression). The constraints $\mathcal{C}$ can be distinguished as follows:

- *Context*-based: If the context of the query is equal to the context on the relation, the traversal follows the relation.
- *Attribute*-based: If the attribute of a concrete entity fulfills the constraint (predicate) on the relation, the entity is retained in the result set. Otherwise it is removed. For the detailed algorithm see section 5.
- *Parameter*-based: If the parameter of the query fulfills the constraint (predicate) on the relation, the traversal follows the relation.

Language Expression for Constraints

The constraint $c \in \mathcal{C}$ on a relation can be formulated as

$$[< context > [.]][ATT. < attribute >< operator >< value >] \quad (4)$$

$$[< context > [.]][< parameter >< operator >< value >] \quad (5)$$

Context is an optional term that can be combined with attribute and parameter based constraints. This means that context-specific attribute / parameter constraints can only evaluate positively if the context is correct. "ATT" is a special terminal symbol to distinguish between attribute and parameter based evaluation (cf. language expression (4)). "$ATT.$" is mandatory for defining a predicate based on attributes. An example for attribute-based constraints is the language expression "$ATT.HiringYear > 2$". It is also possible to assign only the context to the relation, depicted in figure 5. This context-specific constraint is independent of attributes and parameters.

The main difference between constraints based on attributes and parameters is that the values of the entities' attributes are stored in the organizational model. The attribute based constraints are evaluated purely based on model-internal information. Parameters, in contrast, are passed from outside the organizational model and evaluated on the predicates on the relations. The syntax of the language expression for constraints based on parameters is shown in (5). The external parameter is formulated as part of the WITH-clause, described in [11, Figure 4].

Role-Dependent Relationship Type

The role-dependent relationship type is used to "constrain" a relationship type of the sets $\mathcal{R}_o$ and $\mathcal{R}_u$.

$$\forall \Psi_{a_1,a_2}(\Psi_{a_1,a_2} \in \mathcal{R}_o \vee \Psi_{a_1,a_2} \in \mathcal{R}_u) : \Psi_{a_1,a_2} \subseteq (\Gamma_{a_1,a_2}) \times \mathcal{ROLE} \quad (6)$$

$$\forall \Psi_{a,r}(\Psi_{a,r} \in \mathcal{R}_o \vee \Psi_{a,r} \in \mathcal{R}_u) : \Psi_{a,r} \subseteq (\Gamma_{a,r}) \times \mathcal{ROLE} \quad (7)$$

[3] Concrete relations in the model without constraints are generally valid.

A "basic" relation $\gamma \in (\Gamma_{a_1,a_2} \cup \Gamma_{a,r})$ is only active if the source is an actor that acts in the role r assigned to γ. Functions (8) and (9) specify the assignment from γ to $\psi \in (\Psi_{a_1,a_2} \cup \Psi_{a,r})$. Formulas (10) and (11) specify the assignment from ψ to r.

$$f_{\Gamma_{a_1,a_2}} : \Gamma_{a_1,a_2} \to \Psi_{a_1,a_2} \tag{8}$$

$$f_{\Gamma_{a,r}} : \Gamma_{a,r} \to \Psi_{a,r} \tag{9}$$

$$f_{\Psi_{a_1,a_2}} : \Psi_{a_1,a_2} \to \mathcal{ROLE} \tag{10}$$

$$f_{\Psi_{a,r}} : \Psi_{a,r} \to \mathcal{ROLE} \tag{11}$$

This is shown in figure 4 as "basic" relation between ARB and P and the role-dependent relation between this relation and the role *Lecturer*. Section 4 demonstrates role-dependent traversal by example.

Language Expression for Role-Dependent Traversal

In the previous section, we describe relations that are only valid if an actor assumes a given role r. There are two ways to specify which role an actor assumes in a given query:

1. Explicit definition within the query: Using the "*AS*" terminal symbol, such as "*ARB*" *AS Lecturer*, actors can be assigned to roles. In the example, ARB acts as *Lecturer*. This makes the relation between ARB and P active.
2. Implicit definition: If actors are declared using an expression of the form *Researcher*(*Research Group A*), the role they assume is implicitly contained in the query. In the example, the resulting actors ARA and ARB act as *Researcher*s. The relation between ARB and P is inactive.

4 Cooperation and Context

This section shows the concepts of section 3 by examples. These examples refer to figures illustrating realistic scenarios.

4.1 Purchase Example

The following considerations discuss the cross-organizational purchase process described in 2. In order to illustrate concepts from section 3, we revisit the subjects that receive messages in the process. We show by example how they can be declared using language expressions. We then proceed to retrace the concrete lookup of actors belonging to these subjects. Base for these considerations is the organizational model depicted in figure 4.

The figure shows the organizational model of the three organizations involved in the purchase as seen by the *customer* located within the "University". The

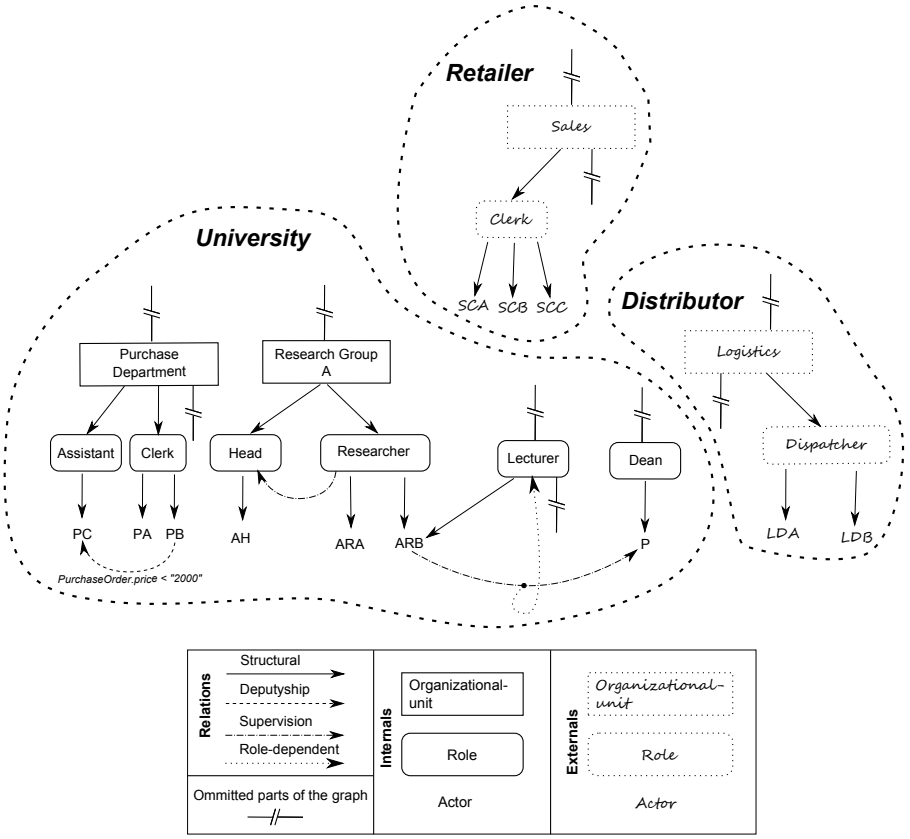


Fig. 4. Organizational model from the customer's perspective

entities of the external organizations "Retailer" and "Distributor" are denoted as *externals*.

The models of the external organizations are limited to actors that interact with the "University".

- For the "Retailer", three concrete actors are modeled: *SCA*, *SCB* and *SCC*. All of them fulfill the same role, *Clerk*, within the same organizational-unit, *Sales*.
- On the "Distributor" side, two actors are defined: *LDA* and *LDB*. They both are *Dispatcher*s within the same organizational-unit, *Logistics*.

We represent the internal organization of the "University" as a subgraph of the complete model. The subgraph is limited to entities relevant to the example. It consists of the organizational-units *Purchase Department* and *Research Group A*. Within the *Purchase Department*, *PA*, *PB* are *Clerk*s and *PC* is *Assistant*. Within *Research Group A*, *AH* is *Head* of the group, *ARA* and *ARB* are *Researcher*s. Additionally, the "University" contains the roles *Lecturer* and *Dean*.

P is *Dean*, *ARB* is *Lecturer* in addition to his role as *Researcher*. This concludes the structural composition of the "University". The entities are connected by structural relations.

The organization-specific relations used in the model of the "University" are:

1. the supervision from the role *Researcher* to *Head*
2. the deputyship from the actor *PB* to *PC*
3. the supervision from *ARB* to *P*

(1.) is a generally valid relation. The deputyship relation (2.) is constrained. It is only valid, if the predicate *price is lower than 2000* — in the context *PurchaseOrder* — is true. The supervision relation (3.) is role-dependent. It is only valid, if *ARB* acts as a *Lecturer*.

Definition of Actors

The problem of resolving actors to subjects arises only in the *send* state (cf. [11]). A subject enters *send* state. The set of possibly receiving actors (the subject) is determined by evaluating the language expression. All the actors contained in the set can decide to *receive* the message. The subjects addressed in the *send* state in figures 2a, 2b, 2c and 8 are assigned to actors by language expressions (queries). These define actors that are responsible for a task. In section 2, we found that the following subjects exchange messages:

– *Customer* perspective
  1. **Initiator** denotes the person that started the process. This is a concrete actor, for the example we assume that *ARB* initiated the process. Consequently, the language expression for this subject is the literal "*ARB*" that resolves to the concrete actor *ARB*.
  2. **Supervisor** refers to the supervisor or the initiator. This is the first occurrence of a language expression where a lookup has to occur. An expression to formalize this actor description is $SUPERVISOR(\text{"}ARB\text{"})$. The literal "*ARB*" remains from step 1, and we want to find their $SUPERVISOR$. Running this query on the organizational model returns *AH*, who is the general supervisor of the researchers in research group A.
  3. **Purchase Department** is the subject that handles purchase requests. In the example model, this means all *Clerk*s of the *Purchase Department*. For failover purposes, the business process management system can also supply additional information to the model[4]. An expression for this subject is $Clerk(Purchase\ Department)$ $WITH\ price = \text{"1500"}\ AND\ context = \text{"PurchaseOrder"}$. The *price* parameter can be used from the concrete process instance, while the *context* "PurchaseOrder" can be a static property of the process template. Running this more complex query on the organizational model returns *PA*

[4] In this case a parameter and a context, cf. section 3.

and *PB* if they are available[5]. If both these actors are unavailable, however, the organizational model could use the additional information to determine *PC* as replacement actor[6]. For further details on the traversal procedure, see section 5.

4. **Retailer** denotes an external subject — an employee of the retailer that can process purchase requests. An expression for this group of actors from the organizational perspective is $Clerk(Sales)$, which denotes any *Clerk* within a *Sales* organizational-unit. Evaluating this query returns the three external actors *SCA*, *SCB* and *SCC*.
5. **Distributor**, as seen from the customer perspective, is not a subject to send to. Consequently there is no need to specify a language expression for this subject.

– *Retailer* perspective
  1. **Customer**, in the simplified version of the process described in section 2, is also not a subject to send to. In a more detailed process, however, the customer would at least be sent a confirmation of the order. The subject would also be represented as a literal with a value extracted from the received order. This is similar to the **Initiator** example in the customer's perspective.
  2. **Distributor**, from the retailer's perspective, is an external subject that can execute delivery orders. An expression that describes the relevant actors in the example organizational model is $Dispatcher(*)$. It addresses any actors fulfilling the *Dispatcher* role, independent of the organizational-unit they are part of. The result set for this query consists of the actors *LDA* and *LDB*.

– *Distributor* perspective
  1. **Customer** is a subject that acts similar to the **Customer** subject in the retailer's perspective. In the simplified version of the process, it is not sent any messages.
  2. **Recipient** is the subject that receives the package and the delivery receipt. For the example, it is represented by the literal "*ARB*". This means that the initiator of the process is carried as a process variable all the way through the external organizations.

So what happens if there exist internal and external entities with the same name? A conflict arises, when a role within an organizational-unit exists in both the internal and external organizations. Imagine an additional internal *Sales* organizational-unit, staffed with *Clerk*s, within the "University" depicted in fig. 4. Then the language expression $Clerk(Sales)$ resolves to all — external, internal and cooperation — actors that fulfill the role *Clerk* within a *Sales* organizational-unit.

This may not always be the desired behavior. We do not want to order supplies from our own sales team. So we need a way to address only external sales clerks.

[5] The simpler expression $Clerk(Purchase\ Department)$ would yield the same result.
[6] This would not be possible with the simple expression.

This can be done by using complex queries as described in [11]. We can subtract the set of internal actors from the result set returned by $Clerk(Sales)$. If we assume an organizational unit *University* (omitted in the figures), we can modify the language expression as follows: $Clerk(Sales)\ NOT\ Clerk(University)$.

The same problem not only applies to internal and external organizational structures, but to cooperation structures as well.

4.2 Temporary Cooperation – A Sample Project

In the previous example, we addressed external actors that were not connected to the internal organization. In order to demonstrate a tighter form of cooperation, we introduce a new example. We consider a joint research project that is run by the "University" and a "Company".

Due to the nature of a project as "temporary endeavor" [12, p.5], the cooperation on the organizational level is restricted in time as well. Figure 5 illustrates the organizational structure of the project. It also shows relevant sections of the internal and external organizational model. The internal structures contain the *Research Group A* from the purchase example. Please note that some internal relations that are not relevant to this context have been omitted from the figure for clarity purposes.

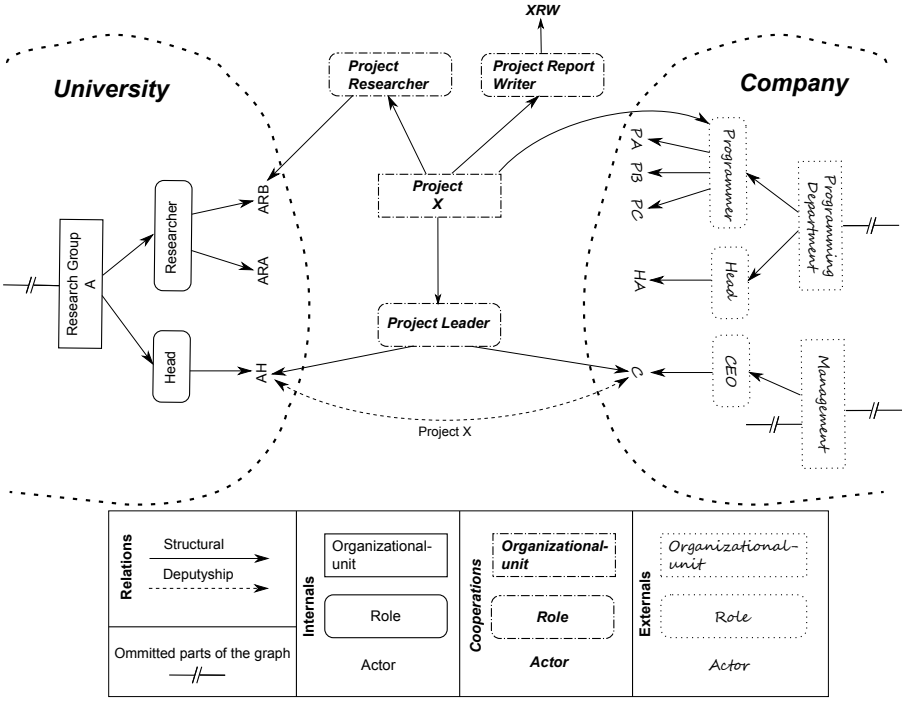


Fig. 5. Cross-organizational project

The *Project X* is represented as an organizational-unit in between the two cooperating organizations "University" and "Company". As such, it is neither internal nor external but a cooperation entity[7]. Within the project, there are the three roles *Project Researcher*, *Project Report Writer* and *Project Leader*. *ARB* is the only *Project Researcher*, *XRW* is the only *Project Report Writer*. The role *Project Leader* is shared between *AH* and *C*. As we can see, most of the actors that fulfill the roles of the joint project are members of organizations, except for *XRW*. *XRW* is a special employee that is employed specifically for the project and for the duration of the project.

As mentioned before, the structure on the "University" side basically consists of *Research Group A* with its *Head*, *AH*, and its *Researchers ARA*, *ARB* and *ARC*. Only *ARB* and *AH* participate in the project and fulfill their roles therein.

The "Company" contributes manpower to the project by providing the *Programmer*s from the *Programming Department* — *PA*, *PB* and *PC*. The *Head* of the organizational-unit, *HA*, is not involved. The *CEO C*, however, shares the *Project Leader* role. The role *CEO* is part of the *Management* organizational-unit of the "Company".

We just determined that *AH* and *C* share the role *Project Leader*. For practical purposes, they decided to be each other's deputy. If one of them is unavailable, the project team remains capable of acting. That is why they established a deputyship relation between each other. As the metamodel discussed in section 3 demands a context constraint, this relation is limited to matters regarding *Project X*. This prevents *C* from having authority on any issues that arise in the "University" outside the scope of the project. If we consider the purchase example from section 2, *C* can not act as supervisor of *ARA* and counter-sign their purchase request.

5 Algorithms to Identify Actors

The subject of this section are the algorithms for traversing the model concerning structural, organization-specific and user-defined relations. We illustrate the algorithms by describing the procedure of interpreting the sample language expressions of section 4 on the organizational model.

A part of the algorithms are excerpts that are described in a less formal manner in [11].

The knowledge hierarchy is used to define different levels of organizational knowledge. The hierarchy is specified from general organizational "rules" (top level) to most specific ones (bottom level). More details on this topic can be found in [8]. In this paper, the consideration of the knowledge hierarchy is reduced to a minimum. Thus, just the resulting actors out of the knowledge levels are listed in the traversal algorithms.

The result set of the traversal algorithms consists purely of actors (if it is not empty). In the following examples, the "Result" is only formed by uniting sets of actors. Intermediate results of traversals are excluded.

[7] As such, it belongs to the entity type $\mathcal{V}_{cooperation}$ discussed in section 3.

5.1 Definitions

Definition 1. $\mathbb{M}_{entity} \rightarrow_{relation-type} \mathbb{M}_R$*: Resulting set* $\mathbb{M}_R$ *of entities listed after traversal of entities connected by* $\rightarrow_{relation-type}$ *and starting from* $\mathbb{M}_{entity}$ *(the cardinal number of set* $\mathbb{M}_{entity}$ *is one, meaning exactly one entity is in that set)*

Definition 2. $\mathbb{M}_{entity} \rightarrow^{\mathcal{C}}_{relation-type} \mathbb{M}_R$*: This definition is analogous to the one before.* $\mathbb{M}_R$ *is the resulting set which lists entities after traversal.* $\mathbb{M}_{entity}$ *is the same as before. The traversal for entities which are possible candidates for the result set will* ***only*** *be done if the constraint* $c \in \mathcal{C}$ *is fulfilled.*

Definition 3. $\mathfrak{expression} \Rightarrow^* \mathbb{M}_{actors}$*: Set of actors* $\mathbb{M}_{actors}$ *resulting after traversal based on* $\mathfrak{expression}$

Definition 4. $\mathbb{M}_1 \cup \mathbb{M}_2 \cup \mathbb{M}_3 \Rightarrow \mathbb{M}$*: Set* $\mathbb{M}$ *results after* $\bigcup_{i=1}^{3} \mathbb{M}_i$*, with* i*: knowledge level. Knowledge level 3 (template level) is in this paper excluded to reduce complexity (for detail see [8], [10]).*

Definition 5. X^k*: Actor* X *results out of considering knowledge level* k *(*$\emptyset^i$ *means that no actor found regarding knowledge level* i*)*

5.2 Structural Traversal

The section describes the traversal of structural relations, instances of the relation type $\mathcal{HAS} \in \mathcal{R}_s$. The interpretation of the language expression starts with a lookup within the index (e.g. organizational-units $I_{\mathcal{OU}}$, roles $I_{\mathcal{ROLE}}$ or actors $I_{\mathcal{ACTOR}}$). If the interpretation is not terminated, the further processing of the traversal is done by following relations.

Example 1: $\boldsymbol{\mathcal{ACTOR}}$

Language Expression: "ARB"

1. Index $I_{\mathcal{ACTOR}}$ lookup for ARB
2. Result: $\{ARB\}$

Examples 2 and 3: $\boldsymbol{\mathcal{ROLE}(\mathcal{OU})}$

Language Expression: $Clerk(Sales)$

1. Index $I_{\mathcal{OU}}$ lookup for $Sales$
2. $\{Sales\} \rightarrow_{\mathcal{HAS}} \{Clerk\}$
3. $\{Clerk\} \rightarrow_{\mathcal{HAS}} \{SCA, SCB, SCC\}$
4. Result: $\{SCA, SCB, SCC\}$

Language Expression: $Dispatcher(*)$

1. Index $I_{\mathcal{OU}}$ lookup for $*$ [8]
2. $\{Sales, Logistics, Purchase\ Department, Research\ Group\ A\} \rightarrow_{\mathcal{HAS}} \{Dispatcher\}$
3. $\{Dispatcher\} \rightarrow_{\mathcal{HAS}} \{LDA, LDB\}$
4. Result: $\{LDA, LDB\}$

5.3 Explicit Search for a Specific Relation

The explicit search for organization-specific relations is valid for instances of $\mathcal{HAS\_SUPERVISOR} \in \mathcal{R}_o$ and $\mathcal{REPORTS\_TO} \in \mathcal{R}_o$ relation types. The lookup within indexes for the start of the interpretation is omitted (cf. section 5.2).

Example 4: $\mathcal{SUPERVISOR}(\mathcal{ACTOR})$

Language Expression: $SUPERVISOR(\text{“}ARB\text{”})$

1. Embedded Expression: “ARB” $\Rightarrow^* \{ARB\}$
2. Next Resolution: $SUPERVISOR(\{ARB\})$
   (a) $\{ARB\} \rightarrow^{\mathcal{C}}_{\mathcal{HAS\_SUPERVISOR}} \emptyset^1$
   (b) $\{ARB\} \rightarrow_{\mathcal{HAS}} \{Researcher, Lecturer\}$
   (c) $\{Researcher\} \rightarrow_{\mathcal{HAS\_SUPERVISOR}} \{Head\} \rightarrow_{\mathcal{HAS}} \{AH^2\}$
   (d) $\{Lecturer\}$ results in $\emptyset^2$
3. Result: $\emptyset \cup \{AH\} \Rightarrow \{AH\}$

The traversal step $\{ARB\} \rightarrow^{\mathcal{C}}_{\mathcal{HAS\_SUPERVISOR}} \{P\}$ is **not** included. The supervisor relation is role-dependent and if no role is specified in the language expression the constraint is by default violated. The example in section 5.4 shows one possibility to fulfill the constraint.

The continuing search proceeds analogously, for both the relations $\mathcal{HAS\_SUPERVISOR}$ and $\mathcal{REPORTS\_TO}$ (e.g. supervisor of supervisor, ...). The resulting set of the embedded expression(s) is the origin for the following resolution, and so on.

The organization-specific relations, except the $\mathcal{HAS\_DEPUTY}$ relation, are traversed across all levels of the knowledge hierarchy. The result set is formed by following the relations on actor, role and template level to a depth of 1.

5.4 Implicit Search

The traversal concerning instances of the $\mathcal{HAS\_DEPUTY} \in \mathcal{R}_o$ relation type is different to the other organization-specific relations. The deputy relation traversal terminates as soon as the concerned knowledge level is processed and the result set is not empty.
The following sample expression describes the traversal procedure for deputies.

[8] The semantics of $*$ is that all entries of the index are assigned to the resulting set.

Example 5: $\mathcal{ROLE}(\mathcal{OU})$
$WITH$ <parameter>=<value> $AND\ CONTEXT$ =<value>

Language Expression (*PA*, *PB* are unavailable): $Clerk(Purchase\ Department)\ WITH\ price =$"1500" $AND\ CONTEXT =$"PurchaseOrder"

1. Expression: $Clerk(Purchase\ Department) \Rightarrow^* \{PA, PB\}$
   (a) $\{PA\}$ is unavailable thus[9] $\emptyset^1 \cup \emptyset^2 \Rightarrow \emptyset \cup$
   (b) $\{PB\} \rightarrow^{\mathcal{C}}_{\mathcal{HAS\_DEPUTY}} \{PC^1\}$ [10]
2. Result: $\emptyset \cup \{PC\} \Rightarrow \{PC\}$

Example 6: $\mathcal{SUPERVISOR}(\mathcal{ROLE}(\mathcal{OU}))$

Language Expression: $SUPERVISOR(Lecturer(*))$

1. Embedded Expression: $Lecturer(*) \Rightarrow^* \{ARB\}$
2. Next Resolution: $SUPERVISOR(\{ARB\})$
   (a) $\{ARB\} \rightarrow^{\mathcal{C}}_{\mathcal{HAS\_SUPERVISOR}} \{P^1\}$ [11]
   (b) $\{ARB\} \rightarrow_{\mathcal{HAS}} \{Researcher, Lecturer\}$
   (c) $\{Researcher\} \rightarrow_{\mathcal{HAS\_SUPERVISOR}} \{Head\} \rightarrow_{\mathcal{HAS}} \{AH^2\}$
   (d) $\{Lecturer\}$ results in $\emptyset^2$
3. Result: $\{P\} \cup \{AH\} \Rightarrow \{P, AH\}$

The language expressions discussed previously are just a small subset of possible statements. They serve to illustrate key aspects of the traversal algorithms. For further information on the grammar of the language, refer to [11].

6 Prototype

This section explains the structure of the editor that manages, among others, the organizational model. For clarity, the example screenshot depicts only the organizational model of the organization "University". The "*IS*" relations in the organizational graph are a result of implementation considerations that improve the runtime behavior. They are analogous to the "*HAS*" described on the conceptual level described in section 3.

The editor, depicted in figure 6, can be broken down into different partitions. The left part, the "navigation" area, displays the different entities of the organizational graph shown in the center. The green icons labeled "O" indicate organizational-units, blue icons labeled "F" are roles (corresponding to functional-units described in [8]) and red icons labeled "A" are the actors. The tree structure on the left shows

[9] There is no deputy relation, concerning the actor (PA) or their role ($Clerk$), on any different knowledge level.

[10] The constraint "$PurchaseOrder.price$ < "2000"" is fulfilled by the value of the parameter and the context is valid.

[11] The role *Lecturer* is explicitly stated in the language expression. Consequently, the constraint (role-dependent relation) is fulfilled.

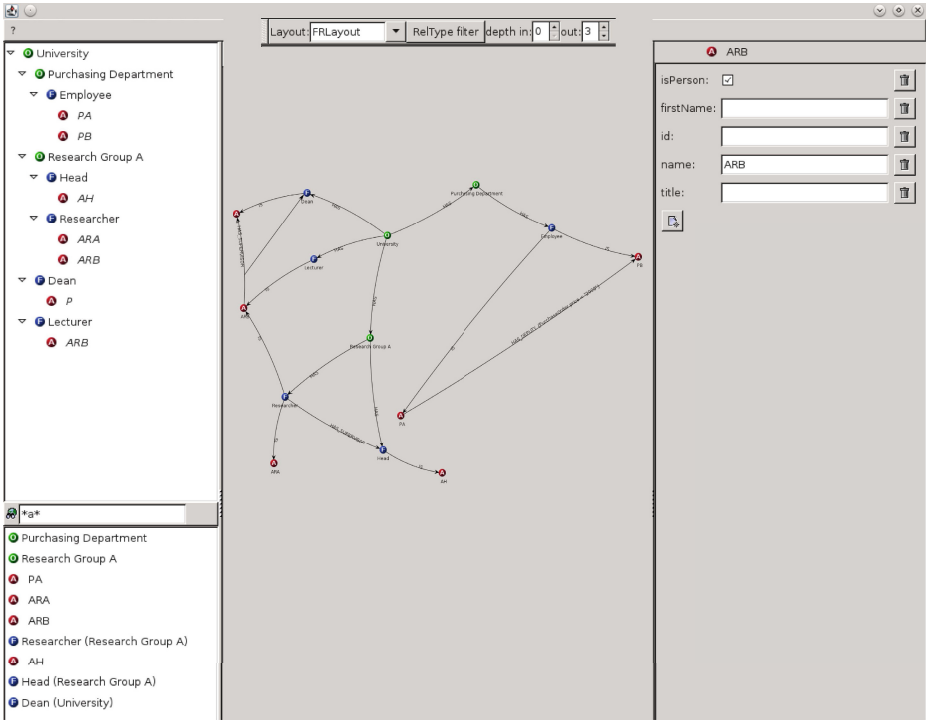


Fig. 6. Screenshot: The graph-based editor of $\mathcal{C} - \mathcal{ORG}$

the organizational structure as quasi-hierarchical composition. Roles are represented as aggregation of actors, organizational-units are aggregations of roles and organizational-units. The drawback of this form of representation is that entities may be shown multiple times. The actor *ARB* is listed twice as he is a *Researcher* as well as a *Lecturer*. He is just displayed twice but stored only once in the organizational model (cf. figure 7).

The tree can be used to quickly navigate to specific entities of the organizational graph (cf. center of figure 6 and figure 7).

By clicking on any icon, the organizational graph focuses on the (sub-)graph of the model. The GUI elements on top configure the depth of this graph (*in*coming and *out*going edges). The depth setting affects only structural relations. It has no influence on organization-specific, user-defined and role-dependent relations. In this screenshot the outgoing depth is 3. The topmost organizational-unit "University" was selected so that the figure shows the resulting (sub-)graph. Furthermore, all organization-specific, user-defined and role-dependent relations, which have relations within this subgraph, are included. An example is the constrained deputy relation between the actors *PA* and *PB*.

If the models of organizations are bigger and more complex than this easy example, the search area on left bottom helps to find entities. The search supports looking for the exact entity (e.g. *ARB* and *Research Group A*). It can also be

used to search based on text segments of the entity name (e.g. *Univ** and **a*). A combination of both search strategies is depicted in figure 6. Clicking on a single search result leads to the same functionality as clicking on an item in the "navigation" part.

The right partition is used to edit entities and relations which are selected in the organizational graph (center partition). This "edit" partition can also be used to add or remove attributes of entities. The administration of constraints on relations is also done in this area. In this example, the actor *ARB* is selected within the organizational graph. Consequently, the "edit" area shows *ARB*'s attributes and their values as depicted in 6.

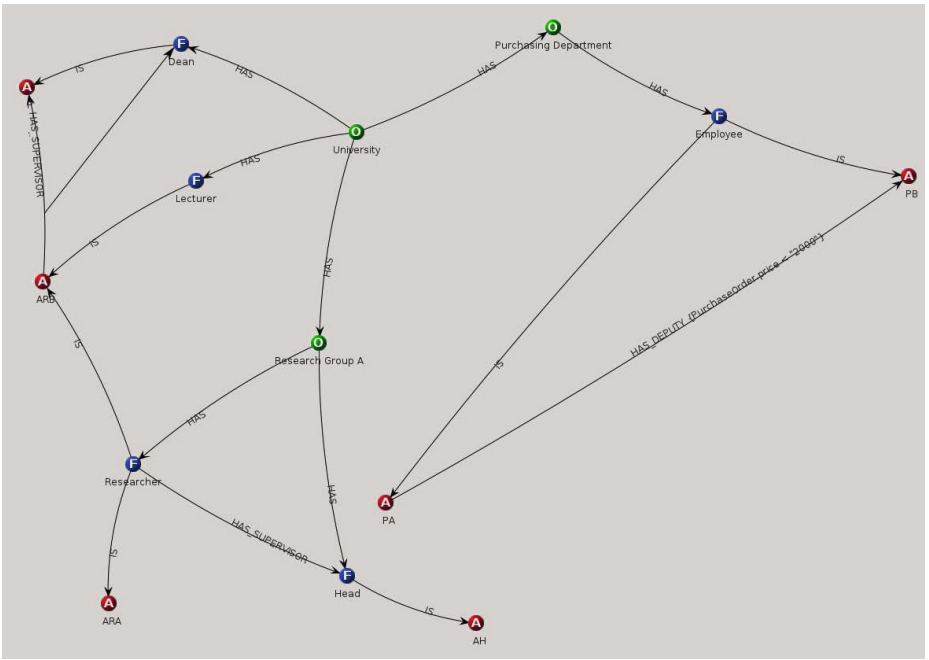


Fig. 7. Screenshot: The organizational structure

The center area of the editor is the main area for displaying the organizational model as arbitrary directed graph (cf. figure 7). This makes it more powerful than the tree structure of the "navigation", as it is not limited to hierarchical structure relations. It can directly show the organization-specific, user-defined and role-dependent relations.

The "graph" partition can be used to connect entities within the organizational model. There are different ways to do so. The first is that a user clicks on an entity and drags a line to an entity. Then, a pop-up dialogue is shown and the user has to decide on a relation type. The relation types are the ones described in section 3 (structural, organization-specific and user-defined).

After the selection of the relation type, the metamodel is used to verify the consistency of the model. If the selected relation type between the selected entities violates the metamodel, the user is informed by an error message and can change it immediately. If the model is consistent with the metamodel, it can be stored in the organization server[12].

Another option to connect entities with relations is to right-click on an entity. After clicking the pop-up menu item "Connect...", the user can select the entity to connect, as well as type and direction of the relation (incoming / outgoing). The selection of the entity can also be done with the aid of a search function. The consistency check is the same as described previously.

In general, the editor is a GUI that enables the user to perform the following operations on the model:

- *Create* entities and relations.
- *Read* subgraphs of the organizational model and display them in an intuitive manner.
- *Update* entity attributes and constraints on relations. The start and end entities of relations can be altered as well.
- *Delete* relations and entities while maintaining consistency[13].

7 Conclusion and Outlook

The approach we described in this contribution makes it possible to describe participants in cross-organizational business processes. This can be done based on their organizational context, as opposed to the total enumeration of members of roles required in traditional approaches. External subjects can be resolved to concrete actors in the same consistent way. They need not be treated as complete black boxes.

In addition to representing cooperations based on business processes, our approach is able to represent cooperations on an organizational level. This is especially true for joint projects that represent a temporary structural relation between two or more organizations but are not defined by reoccurring choreographies. As both internal and external organizational entities and relations are kept within the same model, it is possible to transfer external structures into the internal structures. This may be relevant when considering the acquisition of one organization by another or the foundation of a holding company.

The metamodel enables us to declare role-dependent relations. This means that the relations that are valid for a concrete actor is determined by the role the actor assumes at the time. If we consider the organizational model from the example, it makes a difference whether *ARB* requests a purchase as a *Researcher* or as *Lecturer*. Different people will be their supervisor and responsible for approving the purchase.

[12] The organization server stores the concrete organizational model of the companies (cf. [10, fig. 2]).

[13] Entities can only be deleted if they are no longer interconnected.

There are two major directions of research that will be pursued in the future. First, the language can be extended to allow for more fine-grained control over the traversal. For auditing purposes, it makes sense to extend the language expressions so that it is possible to ask for all *possible* deputies, supervisors, etc. of actors.

The second area of research is more directly related to the cross-organizational aspects of the approach. In such scenarios there should be a way to reference entities from other *models* in a concrete model. This would allow organizations to "publish" sections of their organization in a formal way based on the metamodel described above. This could serve as a formal description of organizational interfaces.

References

1. Vahs, D.: Organisation: Einführung in die Organisationstheorie und -praxis. Schäffer-Poeschel (2007)
2. Fulda, H.: Neue Organisationsformen und ihre informationstechnische Realisierung. PhD thesis, Universität St. Gallen (2001)
3. Meier, A., Stormer, H.: eBusiness & eCommerce. Springer, Heidelberg (2012)
4. Krcmar, H.: Informations management. Springer, Heidelberg (2010)
5. Wüthrich, H., Philipp, A., Frentz, M.: Vorsprung durch Virtualisierung. Gabler, Wiesbaden (1997)
6. Schulte-Zurhausen, M.: Organisation. Franz Vahlen (2005)
7. Schwarzer, B., Krcmar, H.: Neue Organisationsformen - Ein Führer durch das Begriffspotpurri. In: IM Information Management, vol. 9, pp. 20–27 (1994)
8. Lawall, A., Reichelt, D., Schaller, T.: Intelligente Verzeichnisdienste. In: Barton, T., Erdlenbruch, B., Herrmann, F., Müller, C. (eds.) Herausforderungen an die Wirtschaftsinformatik: Betriebliche Anwendungssysteme, AKWI 2011, pp. 87–100. News & Media, Berlin (2011)
9. Fleischmann, A., Schmidt, W., Stary, C., Obermeier, S., Brger, E.: Subjektorientiertes Prozessmanagement - Mitarbeiter einbinden, Motivation und Prozessakzeptanz steigern. Hanser (2011)
10. Lawall, A., Schaller, T., Reichelt, D.: An approach towards subject-oriented access control. In: Stary, C. (ed.) S-BPM ONE 2012. LNBIP, vol. 104, pp. 33–42. Springer, Heidelberg (2012)
11. Lawall, A., Schaller, T., Reichelt, D.: Integration of dynamic role resolution within the s-bpm approach. In: Fischer, H., Schneeberger, J. (eds.) S-BPM ONE 2013. CCIS, vol. 360, pp. 21–33. Springer, Heidelberg (2013)
12. Larson, E.: Project Management: The Managerial Process. McGraw Hill (2010)

A Detailed Purchase Request

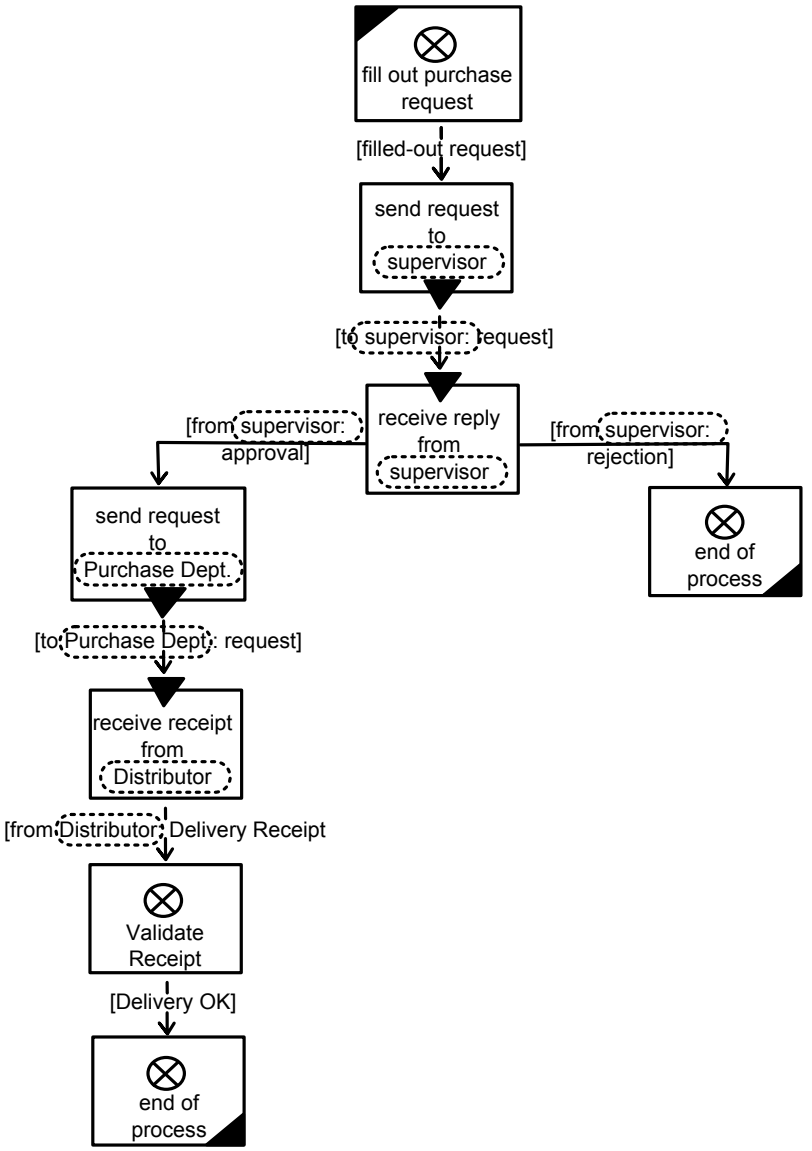


Fig. 8. Purchase request process within the *Customer* subject