Concepts and Implementation of Computer-Supported Structural Organizations

Hartmut Wedekind, Michael Schlundt, Stefan Jablonksi

University Erlangen-Nürnberg
Computer Science VI (Database Systems)
Martensstrasse 3, 91058 Erlangen (Germany)
wedekind@informatik.uni-erlangen.de

Abstract

Computer-supported structural organizations are providing services for many applications, such as Human Ressource Management, Project Management, Workflow Management, and last but the least eCommerce in the near future. A generic operator getAgent() is specified and implemented to select an agent out of a structural organization with its application-specific rule systems.

Keywords: Human resources, project management, workflow management, electronic commerce, use and mention language level, generic operations, agent selection.

1 Introduction: Organizations of Processes and Structures

In the theory of organization a distinction is made between process and structural organizations. A structural organization is achieved by decomposing repetitively a comprehensive total task into subtasks. A composition of generated subtasks in such a way that a person can handle them is called a position. Homogeneous subtasks of a position are called role. A person as a position holder may take over different roles. Various positions may be composed to a group. Supergroups may be established. Large groups are called departments. Companies, public authorities and governamental departments may be considered as groups as well.

Process organizations are opposed to structural organisations. Processes are distinguished from tasks by concrete changes taking place in time. Processes are classified in starting, intermediate and terminating processes. Processes may be decomposed into subprocesses. At the end of a process a state is reached. Tasks of structural organizations don’t have an end, they lasten forever, except new task versions are introduced at discrete times. Tasks of a structural organisation represent potentials to support the execution of processes. Actions are processes performed by men. In computer-supported organisations computing machines are carrying out processes as well. Man and computers together as bearers of processes are called agents. An agent is a person, a program (software) or a machine (hardware) capable to take over a process. Non-agents, the complement, are structuring organizational objects like groups or departments. Designing and implementing the operation getAgent() with respect to a structural organization is subject to this paper.
2 Applications of Computer-Supported Structural Organizations

For computer-supported structural organizations three large application areas are distinguished:

• Personnel planning and administration (Human Resource Management Systems, HRMS)
• Project Management (ProjMgmt)
• Workflow Management Systems (WfMS)

In the near future electronic commerce (eCommerce) with respect to Business-to-Business (B2B) will be a large forthcoming application not discussed in this paper.

2.1 Personnel Planning and Administration (HRMS)

HRMS are tackling problems of personnel requirements, staffing and task assignment (deployment). Personnel requirement is the number and type of employees to perform a comprehensive total task at a certain time (gross requirement). Staffing is the type and number of employees actually available, and deployment means the assignment of tasks to positions. All three parts of HRMS are using intensively maintenance operations \( \text{modifyStructure()} \) with respect to an organizational structure. \( \text{modifyStructure()} \) is considered to be a generic operation. In order to execute maintenance operations consistently, rules of different types are to be considered. Integrity constraints, assignment rules and synchronisation rule are distinguished.

- **Integrity constraints** define the conditions of an organizational structure. E.g. a tree structure may be required or the span of control may be confined to some specific number. Span of control is the maximum number of employees reporting to one manager.

- **Assignment rules** are particularly important in the deployment phase, e.g. a secretary paid according to tariff BAT IIa gets tasks assigned consistent with this tariff. Or: A non-manager is not supposed to work on managerial tasks.

- **Synchronization rules** are needed if more than one person (in general: agent) is eligible. A synchronization rule is applied to determine finally the agent selected, e.g. the senior in service is picked (1-out-of-many) or the n youngest by age are chosen (n-out-of-many).

The operation \( \text{modifyStructure()} \) in general follows the reading operation \( \text{getAgent()} \). \( \text{getAgent()} \) informs about the actual personnel situation needed to find an assessment of the personnel requirement, staffing and deployment phase. Before a structure is modified, type and number of employees as a potential are supposed to be provided form all three aspects. \( \text{getAgent()} \) is an operator applied in all phases mentioned above.

2.2 Project Management (ProjMgmt)

A project is a timely limited, unusual activity. Tasks in a project are complex in comparison to tasks in „staff-line“-structural organization. They are rather novel and they frequently occur only once. Often specific knowledges are needed overlapping various departments. In contrast to project organizations classical „staff-line“-structures are called „routine-organizations“.

2
A project organization is not supposed to be an „isolated state“. Links to the routine-organization are permanently needed. Three types of links between projet- and routine-organization are possible.

(1) **Subordination:** The project organization is handled like an ordinary department of the routine-organization

(2) **Peer-to-Peer:** A new dimension is generated for a project organization. The term „matrix organization“ was introduced. An employee as an element of such a matrix has two managers: One personnel manager in the routine organisation and one technical project manager.

(3) **Superordination:** The organizational core are project organisations, the routine organization is assigned to special staff departments.

The predicates „novel“, „only once“, „timely restricted“ do not change the basic problems of HRMS. However, project management affects considerably number and type of rules. Special integrity constraints must be imposed depending on whether subordination, peer-to-peer, or superordination is applied. Particular assignment and synchronization rules are needed, because projects require a entirely different deployment. The operator `getAgent()` is of paramount importance to find appropriate employees for projects.

### 2.3 Workflow-Management-Systems (WFMS)

Workflow-Management-Systems belong to the process organization, which is initiated after an order is placed. From this point of view assignment and synchronization rules are entirely different, because concrete order-oriented processes instead of abstract tasks are at stake. Given a process, who is the agent to perform or to control it? The operation `getAgent()` is vital when a WFMS is referring to a structural organization. WFMS must take integrity constraints into consideration, rather than to modify structures and their conditions.

### 2.4 The generic property of the operation `getAgent()`

Fig. 1 shows the generic property of `getAgent()`.

![Fig. 1: getAgent() as a generic operation (RS = Rule System)](image-url)
A conceptual Schema of Structural Organizations

Because of various heterogenous applications a complete schema of a structural organization and its rule systems must be provided. In particular all language levels (object level and higher meta-levels) are subject to design, because we must assume that in particular tools from outside refer to meta-informations.

We start the discussion of a conceptual schema on the first language level (object level) represented in fig. 2. An entity-relationship diagram shows the organizational objects and their relationships.

Below the schema is described in relational form:

First Language level

A) Organizational Objects

• POSITION (PosHOLDER, Pos#, ...)
  • Müller A1
  • Mayer A2
  • Schulze A3
  ...
  • Drummer C21

• GROUP (GName, ........)
  • Design
  • Specification
  • Implementation
  • Administration
  • Department
  • Company XYZ

• ROLE (RName, ...)
  • Head
  • Designer
  • Engineer
  • Technician
  • Secretary

B) Organisational Relationship

• PLAYS (PosHolder, RName, ...)
  • Müller Head
  • Müller Designer
  • Mayer Designer
  ...
  • Drummer Secretary

The generic operation \textit{getAgent()} provides a potential of available agents. Each application area is using its own rule system (RS). The rule systems RS\textsubscript{1}, RS\textsubscript{2}, RS\textsubscript{3} are not disjoint.
Fig. 2: Structural Organization of a Company XYZ
A conceptual Schema of Structural Organizations

- **GROUP_OF** (SubGName, SupGName, ...)
  - Implementation
  - Specification
  - : DepartmentG
  - : Company XYZ

- **LEADER_OF** (PosHolder, GName)
  - Müller Specification
  - Mayer Implementation

- **BELONGS_TO** (PosHolder, GName)
  - Müller Specification
  - Mayer Implementation

- **RESPONSIBLE_FOR** (RNAME, GNAME, ...)
  - Head Design

- **DEPUTY_OF** (SubPosHolder, SupPosHolder, ...)
  - Mayer Müller

---

**Second Language level (Meta-level)**

A meta-level with meta-data is only defined precisely, if a schema is given as proper name and meta-predicates are assigned to the name. Thus a new meta-proposition and thereby a new language level comes into play. POSITION is a schema, in our case a relational schema. In order to introduce the word as a proper name and not as a predicate, POSITION is put in general in quotation marks indicating that a schema and not an object is meant. We may now assign to „POSITION“ a meta-predicate, e.g. Agent. The proposition without quotation marks, i.e. „POSITION is an Agent“, is wrong from a logical point of view, because POSITION is a relation, a predicate on the first language level, introduced for classification purposes. In order to talk about POSITION as a schema, quotation marks are needed. Since this important point is always omitted in the literature, it is emphasized here.

There are two relations on the second language level: ORGANIZATIONAL-OBJECT (Type A) and ORGANIZATIONAL-RELATIONSHIP (Type B)

- **ORGANIZATIONAL_OBJECTS** (GName, Activity)
  - "POSITION“Agent
  - "GROUP“ Non-Agent
  - "ROLE“ Non-Agent

Agent and Non-Agent are meta-predicates assigned to the schema „POSITION“, „GROUP“, „ROLE“, resp. Again: Without quotation marks it is asserted that POSITION is an Agent, which is obviously nonsense. Quotation marks „“ in logic are name-building functors, i.e. particular operations.

---

1. ‘Peter is short’ refers to a man named Peter ‘„Peter“ is short’ refers to the schema Peter as a word. „Short“ is short’ makes sense, called self-describing (autologic). ‘Short is short’ is nonsense.
Let us now consider the organizational relationship (Type B)

\[
\text{ORGANIZATIONAL_}
\text{RELATIONSHIPS (RelName, From, To)}
\]

"PLAYS" POSITION ROLE
"GROUP_OF" GROUP GROUP
"LEADER_OF" POSITION GROUP
"BELONGS_TO" POSITION GROUP
"RESPONSIBLE_FOR" ROLE GROUP
"DEPUTY_OF" POSITION POSITION

„What Agents are responsible for the GROUP Implementation?“ is a typical particular operation \text{getAgent}() using both language levels.

**Third Language Level (meta-meta Level)**

It is well-known that data on the first level are not any more visible on the third level. We are talking completely general. In our case there are only the terms ORGANIZATIONAL_OBJECT and ORGANIZATIONAL_RELATIONSHIP taken from the second level. This corresponds to the notion of a graph, by which organizational structures are visualized quite often.

\[
\text{ORGANIZATIONAL_}
\text{STRUCTURE (ComponentName, ............)}
\]

"ORGANIZATIONAL_OBJECT"
"ORGANIZATIONAL_RELATIONSHIP"

The third language level is considered as a keystone of an archway, keeping the other stones in place. The representation shown above may be extended to a complete repository (1). In a repository for organizational structures all attributes have to be described on a higher level, in our case for example From and To.

**4 Implementation of Structural Organizations**

**4.1 Introduction: Treatment of Meta-Data**

To store meta-data in an object-datastream or to provide a separate storage area is a general problem of Computer Science solved merely in an ad-hoc manner. Take for example classical programming and consider the field length \(fl\) of a field \(f\) in a data definition. Instances of \(fl\) are meta-predicates of instances of \(f\). E.g. if the instance of a field „Country“ is England than the meta-predicate \(fl\) is seven. England is supposed to be put in quotation marks from a logical point of view. If \(fl\) is of variable length then the data are stored in the object-stream. If \(fl\) is of fixed length it is factored out and stored as a meta-data elsewhere. This is a quite simple approach applicable only to simple data. As soon as data get more and more complex and performance as well as logical structures are an issue, no simple solution can be offered. Due to [5] there was quite a debate in the standardization committee of
CORBA on this issue. Finally, the committee decided in favor of a repository, i.e. a separate storage system for meta-data. There is no doubt that this is a more complicated approach than an inline implementation.

4.2 Implementing a Structure

The aim of our implementation is to extend our WfMS called MOBILE by an organizational component [4]. Since MOBILE was designed according to a clean-cut modular decomposition with strictly separated aspects (dataflow, controlflow, application, history, etc.) it is allowed to migrate the organizational component to other applications areas without fundamental difficulties.

The approach we followed may be called a „middle-of-the-road“ solution. In particular the first and second language level were combined for performance reasons, though logically rather doubtful. Let us first have a look at the ORGANIZATIONAL_OBJECTS. Without information losses we transform the objects into the following relational forms by copying down the names „POSITION“", „GROUP“", and „ROLE“":

```
ORGANIZATIONAL_OBJECTS (ONAME, Activity)
    „POSITION“ Agent
    „GROUP“ Non-Agent
    „ROLE“ Non-Agent

O_OBJECT_INSTANCES (InstName, OName, ...)
    Müller POSITION ...
    Mayer POSITION ...
    Schulze POSITION ...
    Drummer POSITION ...
    Design GROUP ...
    COMPANY XYZ GROUP
    HEAD ROLE ...
    Secretary ROLE ...
```

There are some technical advantages:

- Imagine a new organizational object has to be introduced. Take for example POSITION as new. Instead of declaring a new relation on the object level in some expensive DDL-Operation and reflecting it on the meta-level (DDL = Data Definition Language), only tuple insertions are now required.

- If an organization expands the framework of relations remains stable. This is particulary useful if a connection with existing conventional personnel databases is required, where an integration of meta-levels is out of scope. No meta-levels are explicitly defined in conventional systems.

For relationships a „copying down the name“ is introduced likewise:
4.3 Implementing Assignment Rules

Organizational structures are only evaluable by a `getAgent()` operation, if the conceptual schema of a structure is extended by assignment rules.

„If an order value is less than $100.000 give me all agents in the group „specification“ playing the role „designer“, else an agent playing the role „head“ is needed.

This is an assignment rule formulated as a query with respect to a workflow system to find an agent for a subprocess called „Signing a contract“. Notice that not only the organizational structures is involved. The term „order value“ is part of the data aspect of a workflow. The logical structure of the above query exhibits that a conventional query language like SQL is not suitable to specify more general assignments rules. Therefore in our implementation the following approach was chosen:

Assignment rules are implemented in Java within the organizational component. Required data and parameters are provided by other components, e.g. the data aspect to render the particular order value in our case.

Java allows loading the code at runtime by a particular classloader. The existing classloader was modified in order to load java code from databases, networks etc. at runtime.

Summarizing the implementation the following steps should be emphasized:

• Assignment rules are specified as java-classes.

• Java-classes are stored within the organizational component in compiled form.

• If an evaluation of an assignment rule is needed, parameters from outside are loaded at runtime. The rule is executed in an interpretative mode.

As a design goal data structures are kept transparent to programmers of assignment rules, i.e. only the application structures (there are persons, persons are working in groups etc.) are known to programmers. To achieve such an abstraction, an application-independent layer between the „lower“ database schema and the „upper“ applications-dependent programmer’s view is introduced (fig. 3). In the intermediate layer the data base tables are hidden by proxy-objects. Proxy-objects offer defined interfaces for programmers of assignment rules. In addition a translation of application-oriented specifications of assignment rules in SQL-statements is provided. These statements are very complicated, user-unfriendly and showing clearly that SQL was not designed for these types of applications.
The architecture in fig. 3 has the following advantages:

- Assignment rule programmers need not to „see“ the internal DB-table structures.
- Assignment rules are readable (for persons having some knowledge of Java) and thereby maintainable.
- Assignment rules are robust with respect to a change of underlying table structures.

```java
// get Proxies for organizational objects

RelationShipProxy plays = orgInst.relShip("PLAYS");
RelationShipProxy belongsTo = orgInst.relShip("BELONGS_TO");
AgentProxy rolePlayer = plays.fromAgent("POSITION");
NonAgentProxy role = plays.toNonAgent("ROLE");
AgentProxy groupMember = belongsTo.fromAgent("POSITION");
NonAgentProxy group = belongsTo.toNonAgent("GROUP");

//Conditions

Condition cond1 = new Relation( group.instAttr("NAME"), ",=", "Specification" );
Condition cond2 = new Relation( role.instAttr("NAME"), ",=", "Designer" );
Condition cond3 = new Relation( rolePlayer.id(), ",=", groupMember.id() );
Condition cond  = new AndCondition( cond1, cond2, cond3 );

//prepare agent-selector

AgentSelector selector = new AgentSelector( rolePlayer.id() ); selector.setCondition( cond );

// perform query

AgentId agentId[] = orgDB.agentIds( selector );
```

Fig. 4: Assignment Rules in Java Code
Implementing Assignment Rules

4.3

Fig. 4 shows the example of the query mentioned above. We confine ourselves to a representation without considering the data aspect, thus implementing the query: Give me all agents, belonging to the group „Specification“ and playing the role „Designer“.

Every assignment rule is organized in four sections:

• Definition of the proxy-objects to be applied (e.g. plays, belongs_to, etc.). Notice that a programmer needs to know only the professional structure of an organization.

• Specifying the conditions applicable to assignment rules. Proxy-objects defined above are denoted by predefined interfaces (e.g. group.instAttr).

• Calling the selection operation for agents consisting of proxy objects and conditions (selector.setCondition).

• Perform the selection of agents (orgDB.agentIds(selector)).

Return values are the identifications (Ids) of the selected agents. These values are provided for the application area, e.g. a WfMS.

Conclusion: Looking into the future

Looking into the future with the advent of electronic commerce in a Business-to-Business (B2B) mode the specified and implemented operator getAgent ( ) will be of outstanding importance. In B2B contract negotiations between buyers and sellers are emphasized. A buyer is looking for a relevant seller in an organization using getAgent ( ), a seller is investigating a potential buyer’s organization likewise. Negotiations in any way are controlled by persons or agents, qualified by their position within an organization and the rule systems thereto pertaining.

References


