Role and Rights Management 10

Alexander Lawall, Thomas Schaller and Dominik Reichelt

Abstract

Role and rights management of today's IT landscape is a challenging task that causes problems concerning the redundancy of organizational knowledge. This knowledge is the basis for specifying access rights and task assignment. As a consequence, the widespread technological methods are prone to inconsistencies on organizational changes, such as employees leaving, joining or moving within the organization. For this purpose, an approach is needed that offers both a comprehensive organizational meta-model and a declarative organization query language. The central meta-model helps to partially overcome the redundancy problem. In conjunction with the organization query language, the problems caused by redundancy is minimized. A query language expression describes formally characteristics of agents that are assigned to access rights, or tasks. Accordingly, this new approach uses a *descriptive* approach instead of *total enumeration* as required by other approaches. Thus, query expressions stay unmodified even if the organization changes.

A. Lawall (✉) · T. Schaller · D. Reichelt
Institut für Informationssysteme der Hochschule für Angewandte Wissenschaften Hof,
Alfons-Goppel-Platz 1, 95028 Hof, Germany
e-mail: alexander.lawall@hof-university.de

T. Schaller
e-mail: thomas.schaller@iisys.de

D. Reichelt
e-mail: dominik.reichelt@hof-university.de

A. Fleischmann et al. (eds.), *S-BPM in the Wild*,
DOI 10.1007/978-3-319-17542-3_10

10.1 Role and Rights Management

This section describes what role and rights management in business process management is about and focuses on typical problems that companies have to face in this area. We will show the reasons for these problems and discuss solutions. At the end a novel, S-BPM-like organization server for role and rights management is presented.

10.2 Motivation

Looking at the various applications and systems needed for running a business, one thing becomes obvious: In almost every application, there is the need to maintain a model of the organizational structure including roles the agents. This model is required in order to define access rights or assign tasks to agents (in the case of a workflow management system). These redundancies lead to a great maintenance overhead that—even for small businesses—can grow to a great burden.

Another issue is that almost all applications try to model an organization as hierarchy or tree. But organization theory literature reveals that companies tend to be multidimensional graphs rather than just trees. In practice, this leads to a lot of workarounds within the used software components that are also not easy to maintain.

A general security problem is the result. Because of the complexity of the management task, nobody is able to guarantee that the agents only see the data that they are supposed to see. Often, the process of granting access to data items is well organized in companies. However, a proper process for revoking access rights in case an agent is transferred to a new position or is leaving the organization is missing.

10.3 What Role and Rights Management Does

Role and Rights Management is involved, within a business process, (1) when a subject has to be mapped to a concrete agent and (2) for determining access rights to data objects.

10.3.1 Business Processes

Figure 10.1 shows the typical process of a business trip approval. When it is filled out, the request has to be sent to the subject “supervisor”. The role and rights management has to determine which agent can take over the task of the subject “supervisor”. At first glance, this seems to be easy. In reality, however, it is often the case that the boss is not available and a deputy has to be determined.

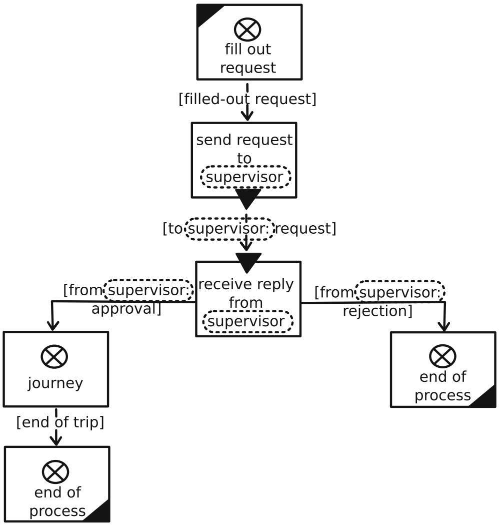
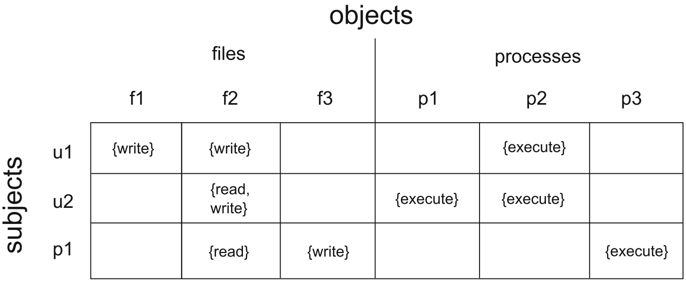

fill out request
[filled-out request]
send request to supervisor
[to supervisor: request]
receive reply from supervisor
[from supervisor: approval]
[from supervisor: rejection]
journey
end of process
[end of trip]
end of process

Fig. 10.1 Approval of a business trip (adapted from Fleischmann et al. 2011), subject behavior diagram. © 2011 Hanser Munich, reproduced with permission

Which agent actually is the supervisor can also depend on the context a subject is acting in. If an employee is working within different projects at the same time, the request has to be approved by the leader of the project that defines the context for the business trip.

10.3.2 Data Access

There are different approaches for defining access rights. The most widespread are the access control matrix and the role-based access control (RBAC) model. The access control matrix simply describes which subjects have access rights to what data objects. Subjects can be agents like users, processes or even hardware components (e.g., a printer or a fax machine). Data objects can be files, tables, processes and so on (Fig. 10.2).



	objects					
	files			processes		
subjects	f1	f2	f3	p1	p2	p3
u1	{write}	{write}			{execute}	
u2		{read, write}		{execute}	{execute}	
p1		{read}	{write}			{execute}

Fig. 10.2 Access control matrix (Seufert 2001). © 2002 Steffen Seufert

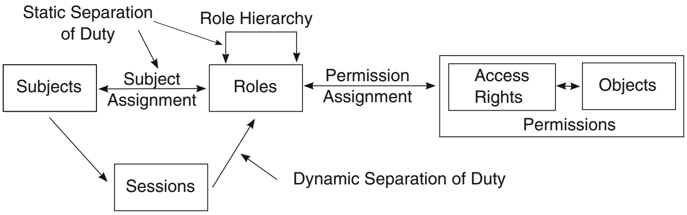


Fig. 10.3 Role-based access control (adapted from Ferraiolo et al. 2001). © 2001 ACM Transactions on Information and System Security

In general, there are two variations of access rights:

- All subjects have all access rights on all objects, except for rights that are explicitly denied.
- No subject has any access rights on any object until the access right is explicitly defined. This is the more common case.

In role-based access control (RBAC), permissions are not directly assigned to agents, but are instead accumulated in roles (Ferraiolo et al. 2001). Users are then assigned to these roles, thereby acquiring the roles permissions. A role typically contains all clearances needed in an organizational unit or for a specific job function. As the number of roles is usually assumed to be considerably lower than the number of agents, the number of administrative tasks required for maintaining the permissions can be reduced.

There are several extensions of the presented core RBAC. In Chen (2011) and Chen and Zhang (2011), the extensions of RBAC include role hierarchies, constraints and the combination of role hierarchies and constraints (cf. Fig. 10.3). Role hierarchies are used to inherit access rights. For example, a head of a department is superior to his clerk and has also access to all objects which the clerk is assigned to. With constraints, subject assignment and role relations can be restricted via the use of predicates.

10.4 Current Problems and Possible Solutions

10.4.1 Redundancy

Independently of the question of which access control mechanism to use, the problem remains that the role subject assignment has to be done in all application systems. This can grow to a great burden and is a source of security problems if not all policy definitions are kept up to date. We think that it is a good idea to think about an organization server that centralizes the task. This way, all organizational and policy definitions are defined and maintained in one location, reducing

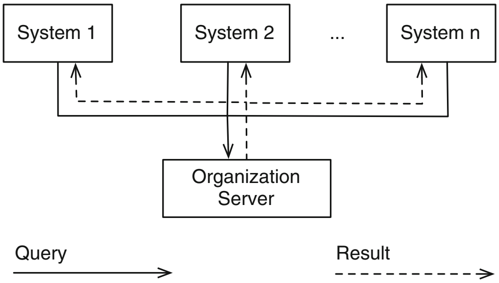
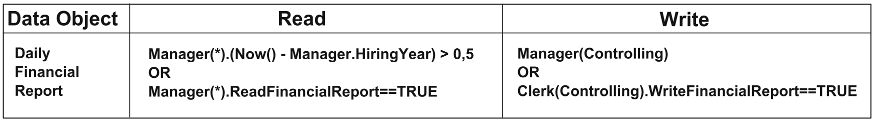


Fig. 10.4 Outside view of an organization server (Schaller 1998). © 1998 Thomas Schaller, reproduced with permission



Data Object	Read	Write
Daily Financial Report	Manager(*).(Now() - Manager.HiringYear) > 0,5 OR Manager(*).ReadFinancialReport==TRUE	Manager(Controlling) OR Clerk(Controlling).WriteFinancialReport==TRUE

Fig. 10.5 Access control matrix (Lawall et al. 2012). © 2012 Springer-Verlag Berlin Heidelberg, reprinted with permission

redundancy and enabling a higher level of security. Since the early 1990s, science has brought up different ideas of how to implement such a server.[1] It is funny, however, that this issue is not recognized in industry. Instead, the users are left alone with their maintenance problem.

Figure 10.4 shows the embedding of such a server. From an outside view, the server fulfills two tasks. First, it maintains the agents of the company, such as users, applications or systems. Secondly, it provides a language that makes it possible to "talk" with the server using an Organizational Query Language (OQL). As a simplified example, an expression in OQL could look like "clerk(claims department).(Now() − clerk.HiringYear) > 10". This means that we are looking for all clerks in the claims department of an insurance company that have been on the job for more than ten years. This language enables clients to specify access rights or task assignments according to the real-world needs. Let us first examine a simple policy definition scenario. In Fig. 10.5, OQL expressions are used for defining access permissions. Let us look at the read policy. The general rule is that all managers that have been with the company for more than half a year can read the daily financial report. The "OR" term of the expression defines an additional policy exception rule by referring to a specific "ReadFinancialReport" flag. At the moment a user would like to have access to the secured data object "daily financial report", the client application passes the OQL statement to the organization server. The server resolves the expression to a subset of matching agents, which is passed back to the calling application (client). The client will grant access if the agent is an element of the returned subset.

[1] For further reading please see Bussler (1997) or Schwab (1998).

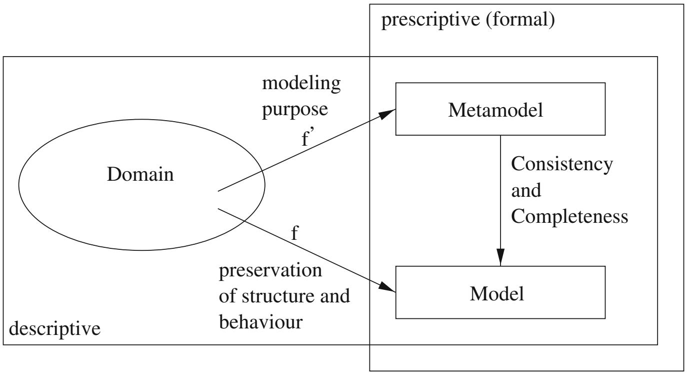


Fig. 10.6 Model and meta-model (Schaller 1998). © 1998 Thomas Schaller, reproduced with permission

The case of task assignment is very similar. In S-BPM, the subjects fulfilling a task are specified by an OQL expression.[2] This expression is passed to the organization server when the task has to be executed. The organization server returns a set of agents that satisfy the specification. Based on additional information, e.g., the employees current workload, the workflow management system decides which members of this set the task will be assigned to.

10.4.2 Wrong Models and Meta-Models

Previously, we argued that it is good to have **one logically central** organization server that is responsible for policy resolution and based on an organizational model. In order to work properly, this model has to be semantically and syntactically correct. Semantic correctness means that the structure and the behavior of the organization is represented correctly in the model, according to a defined modeling purpose. Syntactic correctness means that the model is set up consistently according to a given meta-model that defines how the building blocks of the model can be combined.

According to Fig. 10.6 there are two problem domains.

- Maintaining a wrong model (arc f).
  This situation often happens when the model is set up wrong initially or if the model is not kept up to date. Especially the second point is an issue for companies. As already described, they have to deal with multiple role and rights management systems and it is very hard to keep all of them up to date.

[2] In Fig. 10.1 the receiver of the send request task can easily be specified using the expression *supervisor* (*initiator*).

- Choosing the wrong meta-model for the modeling task (arc f').
  The widespread stereotype for modeling organizations is the hierarchy (mathematically a tree).[3] This may come from the early days of computer science, when a lot of things were organized as trees, like the file systems of a computer. Another influence may be the fact that companies often represent their organizational structure as hierarchies. Consequently, it is not a big step to use these representations as a basis for the role and rights model. Organization theory reveals that companies are often not structured as hierarchies. This is because of things like projects, councils, divisions and so on. The elements of this so-called *shadow organization* lead to the fact that an employee can have multiple supervisors, depending on a given context. The result is not a tree but a general graph that is—due to different relationship types like deputyship, supervision and so on—multidimensional. Due to the wrong meta-model, the administrators have to build workarounds to map the organization to the tree structure. The result is an unsophisticated representation of the company that is a source of security issues and business process exceptions.

10.5 Requirements for an Organization Server—A Case Study

Let us have a look at a real-world scenario within an insurance company. According to the organization handbook, a claims department has a manager, a number of clerks and a lawyer. Generally the lawyer is the deputy of the department head, cf. Fig. 10.7.

We examined two concrete departments: One being responsible for "Car Damages", the other for "House Damages". Compared to the general structure and policies, we observed some differences (cf. Fig. 10.8). At "Car Damages", there was an additional secretary position. In the absence of the manager, organizational tasks were assigned to the secretary position. There was a change in the deputyship between the department head and the lawyer as well. Byron, the lawyer, had been working in the department for only three weeks and therefore was not very experienced. The clerk Winter had been working in the department for over ten years. Based on that constellation, the department head Smith decided that Winter should be his general deputy. Hinton was as well a deputy for Smith, but only depending on some constraint information, such as for instance the cash value of a claim (constrained deputy relation in Fig. 10.8).

Looking at these two departments, we also found an interesting mutual deputyship between the lawyers of the two departments (cf. Fig. 10.8). This observation becomes important when thinking about dividing the organization

[3] This can especially be found in RBAC-based approaches.

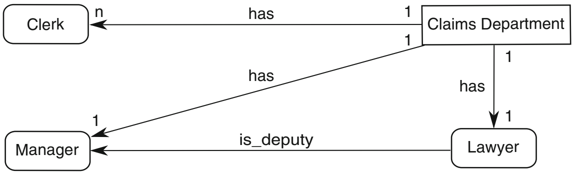


Fig. 10.7 Claims department in general. © 1998 Thomas Schaller, reproduced with permission

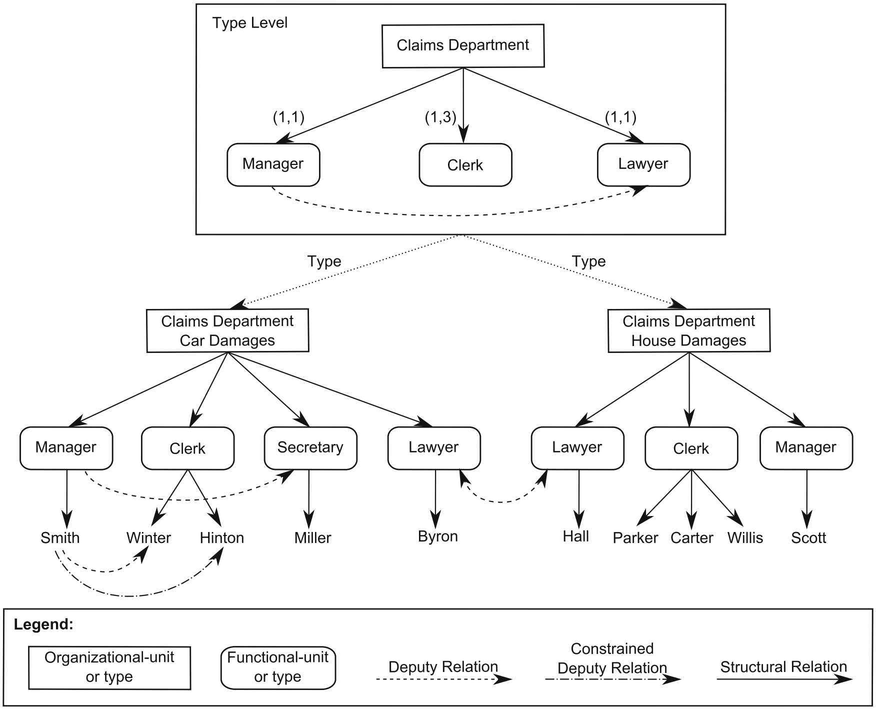


Fig. 10.8 Type and instance level of the example (adapted from Lawall et al. 2014a, Fig. 3). © 1998 Thomas Schaller, reproduced with permission

system into types or classes on the one hand and instances on the other. Please note that the relationships defined until now are specified on different levels of abstraction (roles and agents).

The observation made gives us some insights about the requirements an organization server and its meta-model has to meet.[4]

[4] A complete overview can be found in Schaller (1998).

10.5.1 Knowledge Hierarchy

As we have seen, there are different levels of organizational knowledge. On the top level, general structural assertions like “a department consists of one to three clerks” are dominant. We call this level the *type* or *template* level. Knowledge on this level is based on experience and is changed seldom as time goes by. Looking at real-world departments—we will call them *instances*—things become more concrete and specialized. There are concrete positions and relationships between them. Finally, agents are assigned to the concrete positions. The organizational structures on this level are changing more frequently according to the demands of the daily business.

10.5.2 Relationships

An organizational structure is formed by elements and relationships between them. It is important to realize the existence of several relationship types like “is_part_of”, “is_deputy”, “is_supervisor”, “reports_to” and so on.

Positions[5] are abstractions of agents having a defined skill set fulfilling specific tasks. These abstractions help to define a more stable model of the organization that is independent of employee turnover. Relationships can be defined between abstract positions or on the concrete agent level.

Relationships are rarely of a general nature. As discussed in our example, relationships depend on specific constraint information like the cash value of a car claim. Even the “is_deputy”-relationship can depend on projects or products if you think in the terms of a matrix organization. They can also be valid only for a fixed time period.

10.5.3 Intelligent Subject Resolution

If a client system asks the organization server to resolve the OQL expression “Manager(Claims Department Car Damages)”, an intelligent resolution algorithm has to be applied that uses the described knowledge hierarchy (Lawall et al. 2014a, b). By traversing the graph in Fig. 10.8, the algorithm moves to the department “Claims Department Car Damages”, looking for a position “Manager”. After that, the algorithm determines all the agents assigned to that position, finding manager Smith. If Smith is on the job, his identification is handed back to the client system and the search ends. In the case that Smith is not available (e.g., due to vacation or sickness), the algorithm searches for deputy relations between Smith and other agents. Obviously, there are two relations. Whether Winter and Hinton both appear in the search result depends on the constraint on the relation to Hinton and whether they are on the job. In the case of an empty set, the algorithm moves to the position “Manager”,

[5] We make no difference between the term “role” and the term “position”.

looking for a deputy relation and finds the position “Secretary” assigned to Miller. If Miller is on the job, her identification will be returned to the client system. If not, the algorithm has the alternative of determining a valid deputy on the type level. Let us assume that the department is linked to the department type as depicted in Fig. 10.8. Within this type, the algorithm finds the lawyer as a deputy. It moves back to the instance “Claims Department Car Damages”, and checks if there is a position with this name and an agent assigned to that position who is available. If Byron is on the job, his identification is returned. Otherwise, the lawyer of the “Claims Department Car Damages” has a two-way deputy relation with the lawyer of the “Claims Department House Damages”. If this position has an agent assigned to itself and the agent is available, the algorithm will hand back his identification (here Hall, the lawyer of the “Claims Department House Damages”). Otherwise the returned set is empty. In this case, the client has to postpone the execution of the task.

10.5.4 Multidimensional Organizations

Even in organizations that are structured hierarchically at first glance, there are structures belonging to the so-called secondary (“shadow”) organization comprising committees, commissions, boards and so on. The positions and functions of the secondary organization are assigned to the employees. This leads to a multidimensional organization in every case.

10.6 The Organization Server C-Org

Within the S-BPM Research an Organization Server called C-Org was developed. It implements the requirements discussed in the foregoing section and offers central role and rights management to arbitrary clients (see Fig. 10.9). The system was developed at Hof University in Germany. Up to now, C-Org has been connected to several systems like

- Metasonic S-BPM Suite
- Microsoft’s Active Directory
- Bonita Workflow
- Process Maker
- The telephone private branch exchange Asterisk
- Database management systems via an adapted JDBC driver (prototype stage)

Thanks to a small interface, the integration of C-Org into an existing IT environment is simple. Clients send OQL expressions to the server and receive the identities of agents that fulfill the expressions. The test drive has been used successfully to demonstrate how consistent role and rights management can look like in the future. If the organization changes, only the central model has to be altered and from that moment on all systems are up to date.

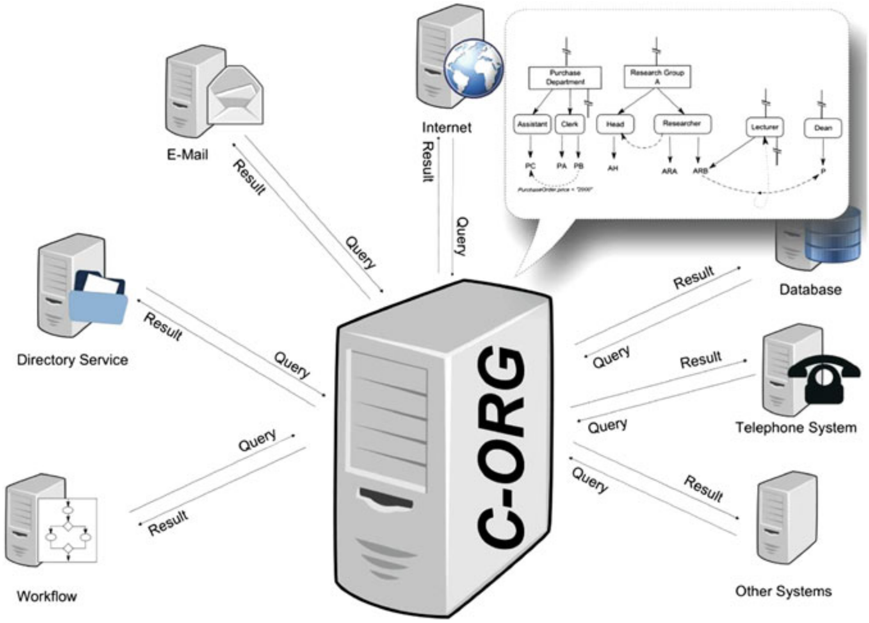
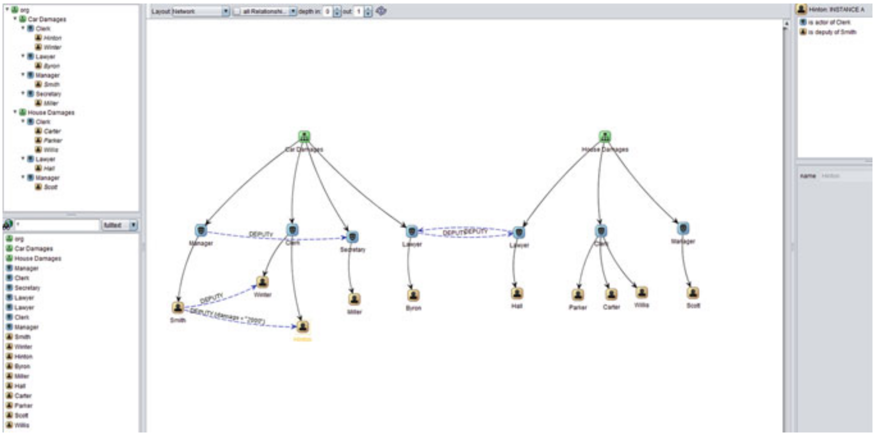


Fig. 10.9 C-ORG as central organization server

10.6.1 Implementation

Figure 10.10 shows the administrator user interface of C-Org (see also Lawall et al. 2014a). It contains a *model* editor, a *search* area, a *tree-navigation* as well as an *attribute pane* and a *relation list* for a selected organizational element.



Layout: Network all Relationshi... depth in: 0 out: 1

Hinton: INSTANCE A
is actor of Clerk
is deputy of Smith

name Hinton

Fig. 10.10 Screenshot: administration view

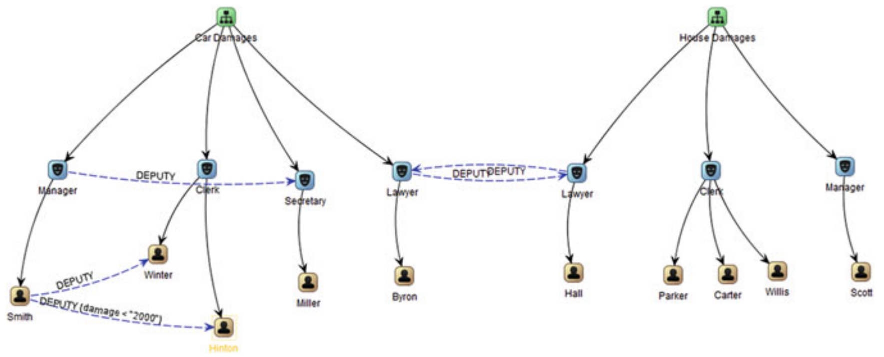


Fig. 10.11 Model region of C-Org

The *model editor* provides a graph-based view on the organizational structure. Organizational elements are represented as nodes and their relations as edges. It provides means to navigate the model by centering on selected nodes. As the central component of the user interface, it is discussed below in more detail.

The *search area* can be used to retrieve a list of organizational elements. It has two modes of operation:

1. It provides a simple text index search for attribute values, e.g., entering "Wi*" will yield Winter and Willis.
2. It can also be used to evaluate OQL expressions. An expression is entered and the result set for the current state of the organizational model is shown.

The *tree-navigation* maps the concrete organizational structure on a tree. Consequently, entities are duplicated in the projection if they can be reached on different paths.

The *attribute pane* in the bottom right section shows the attributes of the currently selected node or relation. It allows a quick modification, e.g., the assignment of a predicate to a relation.

The *relation list* lists all relations of the currently selected node, independently of the relation-types hidden in the model editor. This allows access to connected nodes and significantly reduces the time required to alter existing relations.

For quick access, elements can be dragged from any of the outer GUI sections and dropped into the model editor. If the elements have existing relations to the nodes already shown in the model editor, these relations will be shown as well. Otherwise, the elements are represented as unconnected nodes.

Figure 10.11 provides an enlarged view of the model editor. It contains the instance part of the example model of Fig. 10.8 with the desired[6] relations. The editor also shows concrete constraints (predicates) on relations, e.g., the deputy

[6] The relation types to be shown can be selected.

relation with *damage* < "2000" between *Smith* and *Hinton*. Users perform most modifications of the organizational model via this component. In addition to navigating the model, they can create, modify and delete organizational elements and their interconnections.

10.6.2 Usage of C-Org

From an architectural point of view C-Org can be used as a dedicated server within a company network. Another possibility is to use it as a cloud service within the IBM Bluemix environment.

10.6.3 C-Org from the Viewpoint of S-BPM

C-Org focuses on the subjects and their relationships rather than the organization's hierarchy. The end users are able to specify policies and roles according to their daily needs in a decentralized manner (if they are allowed to). In routine cases, like defining a deputy, a manager can react agilely without involvement of the central IT department. The specific resolution algorithm guarantees that if there is a specific policy on the instance level, it will be used (Lawall et al. 2012). Overall, the approach reduces the workload of the administrators and makes the life of the business people easier.

10.6.4 Additional Features

Despite the task of role and rights management C-Org offers some additional features.

- It can replace the classical mailing lists that have to be maintained by hand. Instead of returning identities, C-Org returns the mail addresses of the agents specified by an OQL expression. In place of enumerating the recipients, the client just describes which persons to write to. Because of the central organization database, the description is always up to date—which is not always true for hand-maintained mailing lists.
- Another nice function is the connection of a telephony server. For example, if a called agent is not available, his deputy can be called instead. This redirection can be fine-tuned by using context information added to the deputy relationship. Another idea is to implement a group call functionality, where all phones of the group members are called and the call will be routed to the first responding agent.

- Discussions with several banking companies revealed that C-Org is also very interesting for compliance management. Based on the central model, C-Org offers a new way for the management and documentation of policies.

10.7 Conclusion and Takeaway

This section is directed at IT architects, system administrators and CIOs who want to have a consistent way to reference organizational elements. This approach can be used to specify organization-wide access rights and policies with minimal maintenance effort. The reference, expressed by the organization query language, remains unchanged in the case of organizational changes. There is no need to alter existing role assignments.

It is also relevant for process owners and modelers who want to find a more descriptive way to define process stakeholders. This allows for a more flexible task assignment based on organizational relations. There is no need for technical workarounds to describe such relations, e.g., a table for supervisors, but the organization server can be asked for the specific case.

In addition to access rights and task assignment, the organization server can also be used for content generation (e.g., intra- and internet pages, customer relationship management systems, etc.). The contents can just be described using the organization query language and is resolved to the current values (e.g., team members, phone numbers, e-mail addresses, etc.).

Similarly to task assignments in processes, recipients of messages (e.g., e-mails) can be described using OQL. This can replace mailing lists and their maintenance. Not just functional e-mail addresses, like mailing lists, can profit from this, but also functional phone numbers.

In all of these application cases, the organizational information is current and does not have to be maintained manually in the individual systems.

References

Bussler C (1997) Organisationsverwaltung in Workflow-Management-Systemen. Ph.D. thesis, Universität Erlangen (in German)

Chen L (2011) Analyzing and developing role-based access control models. Ph.D. thesis, University of London

Chen L, Zhang Y (2011) Research on role-based dynamic access control. In Proceedings of the 2011 iConference. ACM, New York, USA, pp 657–660

Ferraiolo David F, Sandhu R, Gavrila S, Kuhn DR (2001) Chandramouli Ramaswamy: proposed nist standard for role-based access control. ACM Transactions on Information and System Security (TISSEC), pp 224–274

Fleischmann A, Schmidt W, Stary C, Obermeier S, Börger E (2011) Subjektorientiertes Prozessmanagement. Mitarbeiter einbinden, Motivation und Prozessakzeptanz steigern. Springer, Heidelberg (in German)

Lawall A, Schaller T, Reichelt D (2012) An approach towards subject-oriented access control. In: Stary C (ed) S-BPM one—scientific research. LNBIP, vol 104, Springer, Berlin, pp 33–43

Lawall A, Schaller T, Reichelt D (2014a) Enterprise architecture: a formalism for modeling organizational structures in information systems. In: Barjis J, Pergl R (eds) 10th International workshop, CAiSE 2014, LNBIP, vol 191, Thessaloniki

Lawall A, Schaller T, Reichelt D (2014b) Local-global agent failover based on organizational models. In: 2014 IEEE/WIC/ACM international joint conferences on web intelligence (WI), intelligent agent technologies (IAT), brain informatics and health (BIH) and Active media technology (AMT), pp 420–427

Schaller T (1998) Organisationsverwaltung in CSCW-Systemen. Ph.D. thesis, Universität Bamberg (in German)

Schwab K (1998) Konzeption und Implementierung von computergestützten Kooperationsmanagementsystemen. Habilitationsschrift, Universität Bamberg (in German)

Seufert S (2001) Die Zugriffskontrolle—Eine Bestandsaufnahme relevanter Ansätze und deren Weiterentwicklung zu einem Konzept für die Ableitung von Zugriffsrechten aus der betrieblichen Organisation. Ph.D. thesis, Universität Bamberg (in German)