Organizational Modeling in UML and XML in the context of Workflow Systems

W.M.P. van der Aalst
Faculty of Technology Management
Eindhoven University of Technology
Eindhoven, The Netherlands
+31 (40) 247 4295
w.m.p.v.d.aalst@tm.tue.nl

Akhil Kumar
Smeal College of Business
Penn State University
University Park, PA 16802, USA
+1 (814) 863-0034
akhilkumar@psu.edu

H.M.W. Verbeek
Faculty of Technology Management
Eindhoven University of Technology
Eindhoven, The Netherlands
+31 (40) 247 2181
h.m.w.verbeek@tm.tue.nl

ABSTRACT

Workflow technology plays a key role as an enabler in E-Commerce applications, such as supply chains. Until recently the major share of the attention of workflow systems researchers has gone to the exchange of information in cross-organizational processes. Increasingly the focus is shifting from the exchange of data to support for interorganizational workflow processes. One of the initiatives in this direction has been XRL (eXchangeable Routing Language), an extendible instance-based language having an XML syntax and Petri-net semantics. In this paper, we move to the next level by extending XRL with organizational entities, structures, and rules. Hence, we describe an organizational model first in UML and then convert it into an XML DTD. Our organizational model allows for the specification of non-human resources, collections of resources (e.g., departments, teams, etc.), availability of resources, delegation, and role inheritance. Additional features of our proposal are the tight integration of organizational concepts and routing concepts. An important goal of this work is to create standard for organizational modeling much like the X.500 standard for directories.

Keywords

workflow systems, organizational modeling, UML, XML, E-commerce applications, interoperability.

1. INTRODUCTION

With the rapid expansion seen in electronic commerce, there is a major need for infrastructures and frameworks that can be used to implement inter-organizational workflows. In particular it is essential to provide support for routing of documents across organizations in a standardized and yet flexible manner to enable open electronic commerce. Developing more homogeneous languages for various electronic commerce activities is one way to facilitate increased productivity and interoperability.

There are two important perspectives in a workflow system: the process flow perspective and the organizational perspective. The former describes the routing and coordination of tasks within a process, while the latter deals with realities of the organization. In [2,3], an architecture and a language called XRL (eXchangeable Routing Language) are described that provide support for routing of workflow among trading partners for Internet-based electronic commerce services. However, for the successful implementation of workflows in organizations, it is also necessary to model organizational policies carefully so that the workflow system can determine who will do the work.

The decision regarding who must perform a task is an important aspect of organizational policy in a workflow system. Moreover, there are two dimensions related to task assignment: a task assignment could be made based on role (as in role-based access control systems often referred to with the acronym RBAC [14]), or a task could be assigned to a specific employee or user. The second dimension relates to whether the task assignment is made to a team (i.e., a group of users or roles), or to a single individual.

Another aspect of organizational modeling relates to non-human resources needed to accomplish a task. Examples of such resources are equipment like machines, computers, videoconference setup etc., and physical space resources such as conference rooms, classrooms, meeting rooms, etc. Thus, an organization should be able to model a task to be performed by a team consisting of roles and/or individuals, at a specific location using certain kinds of equipment. For example, a personnel committee consisting of four professors and the department chairman must meet to perform the evaluation of candidates task in a conference room equipped with 5 web-enabled PCs and a slide projector.

Therefore, the important elements to be modeled are tasks, roles, users, teams, equipment, and space. While contemporary workflow systems can handle tasks, roles and users, and combine them with routing constructs to describe processes, they are seriously lacking in the handling of notions like teams, equipment and space. Moreover, it is also necessary to model organizational structure in the form of a role hierarchy, ability to make delegations (of various types such as role-role, user-role, role-user and user-user), and various organizational constraints. Some important examples of constraints are binding of duties and separation of duties constraints, where the same user may be required to do multiple tasks (in the case of binding) or prohibited from doing so (in the case of separation).

In this paper, we develop a formal organizational model to capture these requirements. The model is first described in UML and then converted into an equivalent DTD. The UML representation is intuitively appealing and easy to understand; however, XML offers additional advantages as a modeling paradigm, such as greater interoperability. In [2,3], XML was used to describe the routing semantics in the form of a DTD. Therefore, describing the organizational policy in XML also complements this work very well. Since XML has very fast become the lingua franca of the web, a

SAC 2003, Melbourne, Florida, USA.

model built in XML can easily be exchanged between different units of the same organization and also across organizations.

In addition to being a well-known language for modelling, XML also lends itself easily to querying. An organizational model needs to be queried all the time. Thus, in addition to simple queries like finding a user's phone number or skill set, there is also a need to find answers to queries like who is the most suitable delegate for Bill or Sue if they are not available to do a task. An organizational model constructed in XML can be queried easily using a query language like XPath [10] or XQuery [9].

The organization of this paper is as follows. Section 2 will briefly give background information on XRL and present an organizational model in UML. Section 3 shows how to convert the UML model into XML and gives a DTD for it. Section 4 will illustrate how the DTD can be used to create instances that correspond to the model. Section 5 will discuss other issues in the context of this modelling approach and compare our work with related work. Finally, Section 6 will conclude the paper.

2. ADDING THE ORGANIZATIONAL PERSPECTIVE TO XRL

2.1. Background

XRL (eXchangeable Routing Language) is an XML data type definition (DTD) for describing workflow instances. It consists of elements for describing tasks and also various coordination requirements between them such as sequence, parallel, any_sequence, etc. For a full DTD we refer the reader to [2,3]. Once a workflow instance has been described in XRL, then it can be parsed and executed by a workflow engine. Previous efforts towards building such a prototype engine called XRL/Flower have also been described in [2,3]. However, XRL is basically a process modelling language that focuses on routing while omitting organizational details. Therefore, we discuss the organizational model in the next section and later show how it can be incorporated into the process model of XRL.

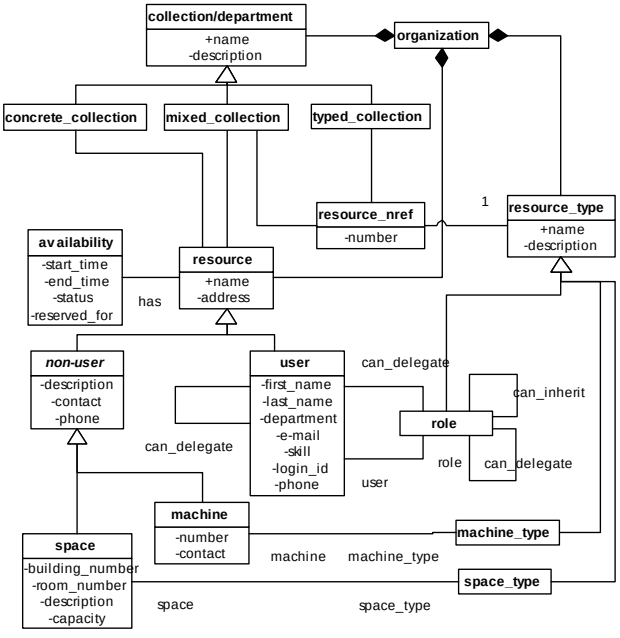

Figure 1. UML model for an organization

2.2. A UML model

We first describe the organization model in UML and then convert it into XML with an accompanying DTD. The UML organizational model is shown in Figure 1.

In this UML model, an organization entity is an aggregate of three other entities: (1) resources, (2) resource types, and (3) collection (or departments). The resources are subtyped into user and non-user entities, which in turn are sub-typed into machine and space entities. Resource_types are abstract entities that group concrete resources such as: role, machine_type, and space_type. A role is related to users because a typical role such as manager would comprise of many users like Bob, Sue and Hong. In general, one role can have many users and one user can have many roles. The can_inherit and can_delegate relationships show the lines of inheritance and delegation. It is important to note that the can_delegate relationship can operate at the role-role, user-user, user-role, and role-user levels. Just like roles are related to users, machine_type and space_type are related to machine and space, respectively. These resource types are introduced to structure the organization and to allow allocation of resources to work without referring to specific resources.

Most workflow management systems assume that only one resource is needed to execute a task. Therefore, it is not possible to specify that a team of users, a user and a machine, or a user and a space are required to execute a step in the process. To allow for the specification and enactment of tasks requiring a collection of users, machines, and spaces, the organization has a collections entity. This entity is sub-typed into concrete_collection, typed_collection, and mixed_collection sub-entities. A concrete_collection refers to specific resources, e.g., a team of workers specified by enumerating its members. A typed_collection refers to resource types, e.g., a selection committee consisting of two full professors and three associate professors (where full professor and associate professor refer to roles). A mixed_collection may contain both specific resources and resource types. The resource_nref entity is used to allow multiple resource_types to participate in a collection as in the above committee example (e.g., two full professors). The availability entity is related to the resource entity to represent the available time periods for each resource. Finally, the department entity is associated with the user entity to describe the details of the department (such as name, manager, etc.) each user belongs to.

3. CONVERTING UML INTO AN XML DTD

The next step is to convert the UML model into an XML DTD so that various details of an organization can be described in XML. This DTD called XRL_ORG.DTD is shown in the Appendix. In large part, the mapping from UML model to a DTD treats each entity in the UML diagram as an XML element. However, some additional elements also have to be introduced. In particular a relations element is added to capture various relationships present in Figure 1. The relations element is a child element of organization. It is used to describe various mappings such as user_role_mapping (i.e., users are mapped onto roles), machine_machine_type_mapping (i.e., machines are mapped onto machine types), and space_space_type_mapping (i.e., spaces are mapped onto space types). Second, role inheritance (can_inherit) and delegation (can_delegate) are also specified using the relations element.

XML documents conforming to the organization DTD can be used to describe organizational entities, structures, and rules (examples will be given shortly). As such they can be applied independent of any routing language, such as XRL. However, our goal is to have a close link between the process perspective of XRL (as described

by the process DTD) and the organizational perspective as described by the organization DTD. Therefore, we added the following constructs to the original XRL language to build a bridge with XRL_ORG.DTD.

```
<!ELEMENT task (event*, resource_binding*)>
<!ATTLIST task
    ...
    function CDATA #IMPLIED
    ...
>
<!ELEMENT resource_binding ((user_ref | machine_ref | space_ref | role_nref |
machine_type_nref | space_type_nref | named_collection_ref | concrete_collection
| typed_collection | mixed_collection)*) >
```

The above snippet shows that a task has two sub-elements. The event* sub-element gives the set of events to be generated on completion of the task. The resource_binding* sub_element represents one or more resources required to complete this task. Therefore, each task may contain multiple resource bindings. Each resource binding refers to a set of concrete resources (users, machines, and spaces), resource types (roles, machine types, and space types), named collections (predefined teams), and concrete/typed/mixed collections (ad-hoc collections of typed and/or untyped resources).

The two models in UML and XML are comparable in large part. However, there are some minor differences. The XML DTD for instance has attributes associated with the can_delegate element. The transitive attribute is used to specify whether the can_delegate relationship is a transitive one (i.e., if A can delegate to B, and B can delegate to C, then does it follow that A can delegate to C). Moreover, the restrictions attribute can be used to impose restrictions or constraints on the delegate relationship. Of course, this can be captured in UML also by introducing several new entities and assigning similar attributes to each one of them, but for simplicity it has been omitted from Figure 1.

4. USING THE DTD TO BUILD AN ORGANIZATIONAL MODEL

The organization is the highest level element in the organization DTD. As described above, it contains four sub-elements: resources, resource_types, collections and relations. These elements and their sub-elements will be described in this section in more detail along with examples.

4.1. Resources and Resource_types

We first give an example of resource_types and resources below along with the main subelements under them. The main sub-element under resource_types are role, machine_types and space_type. The main sub-elements under resources are user, machine and space.

```
<organization><resources>
    <user name="u.b_jones" first_name="Bob"
        last_name="Jones" department="d.finance"/>
    <user name="u.s_brown" first_name="Sue"
        last_name="Brown" depar tment="d.finance"/>
    <user name="u.j_donk" first_name="Jim"
        last_name="Donk" department="d.marketing"/>
    <user name="u.l_hong" first_name="Lee"
        last_name="Hong" depar tment="d.technology"/>
    <user name="u.b_boss" first_name="Big"
        last_name="Boss" department="d.marketing"/>
    <machine name="m.pc_1" description="PC 1"/>
    <machine name="m.projector_1" description="Projector 1"/>
    <space name="s.room_301" room_number="301" capacity="10"/>
    <space name="s.room_307" room_number="307" capacity="30"/>
    <space name="s.room_321" room_number="321" capacity="50"/>
</resources><resource_types>
    <role name="r.professor" description="Professor"/>
    <role name="r.associate_professor" description="Associate Professor"/>
    <role name="r.assistant_professor"  description="Assistant Professor"/>
    <role name="r.post_doc" description="Post Doc"/>
    <role name="r.vp_finance"  description="VP Finance"/>
    <role name="r.vp_marketing"  description="VP Marketing"/>
    <role name="r.vp_technology"  description="VP Technology"/>
    <machine_type name="mt.pc" description="PC"/>
    <machine_type name="mt.projector"  description="Projector"/>
    <space_type name="st.conference_room"
        description="Conference Room"/>
</resource_types></organization>
```

4.2. Creating a resource binding

Next we introduce the resource_binding element which plays a key role in connecting a task with the resources required to perform it. The example below shows how a resource binding can be associated with a specific task, in this case the task named ”review_staff”. Staff can be reviewed by a professor and an associate professor. Alternatively, two associate professors and one post doc may also review staff. Finally, it may also be done by Big Boss. Thus, these are three alternative ways of reviewing staff, hence the three bindings.

```
<task name="review_staff"><resource_binding><typed_collection>
    <role_nref name="r.professor"/>
    <role_nref name="r.associate_professor"/>
</typed_collection></resource_binding><resource_binding><typed_collection>
    <role_nref name="r.associate_professor"  number="2"/>
    <role_nref name="r.post_doc"/>
</typed_collection></resource_binding><resource_binding><concrete_collection>
    <user_ref name="u.b_boss"/>
</concrete_collection></resource_binding></task>
```

Note that one of the possible bindings requires two associate professors. Apparently, it is possible that a binding requires multiple items of some resource type. Therefore, we use role_nref, machine_type_nref, and space_type_nref in collections. These elements have an optional number attribute that denotes the number of items required. This attribute is not needed for resources, because they are unique (there is only one Big Boss.)

4.3. Including non-human resources in binding

The next example shows another kind of binding. It first expresses the idea that there must be two professors, two associate professors, and two assistant professors to hire staff. Then, it also specifies the requirements for a conference room, two PCs, and a projector.

```
<task name="hire_staff"><resource_binding><mixed_collection>
    <role_nref name="r.professor" number="2"/>
    <role_nref name="r.associate_professor" number="2"/>
    <role_nref name="r.assistant_professor" number="2"/>
    <machine_type_nref name="mt.pc" number="2"/>
    <machine_type_nref name="mt.projector"/>
    <space_type_nref name="st.conference_room"/>
</mixed_collection></resource_binding></task>
```

In this way it is possible to combine human resource needs with needs for other resources such as space and equipment.

4.4. Modeling Departments and Teams

Next we discuss how a team can be defined. A department or a team is modeled with a collections element, which consists of one or more named_collection sub-elements. Named_collection in turn has three sub-elements: concrete_collection, typed_collection and mixed_collection. An executive team may be defined as follows:

```
<named_collection name="t.executive"
    description="Executive Team"><typed_collection>
    <role_nref name="r.vp_finance"/>
    <role_nref name="r.vp_marketing"/>
    <role_nref name="r.vp_technology"/>
</typed_collection></named_collection>
```

The above team consists of the VPs for Finance, Marketing and Technology. Since there is no number attribute, its default value is 1.

Now, a resource binding may include the executive team as shown below.

```
<task name="hire_exec_staff"><resource_binding>
      <named_collection_ref name="t.executive"/>
      <machine_type_nref name="mt.pc" number="2"/>
      <machine_type_nref name="mt.projector"/>
      <space_type_nref name="st.conference_room"/>
</resource_binding></task>
```

The notion of teams makes it easier to specify resource bindings. Moreover, multiple teams can also be combined together in a resource binding as well.

4.5. Inheritance and Delegation Hierarchies

Inheritance and delegation are specified by means of can_inherit and can_delegate elements, which are sub-elements of the relations element.

```
<relations><can_inherit>
      <role_ref name="r.professor"/>
      <role_ref name="r.associate_professor"/>
</can_inherit><can_inherit>
      <role_ref name="r.associate_professor"/>
      <role_ref name="r.assistant_professor"/>
</can_inherit><can_delegate>
      <role_ref name="r.professor"/>
      <role_ref name="r.associate_professor"/>
</can_delegate><can_delegate>
      <role_ref name="r.professor"/>
      <user_ref name="u.b_jones"/>
</can_delegate></relations>
```

These hierarchies can be used to determine all the attributes a given role might inherit from other roles, and to also find the delegates for a particular role or a user. We will see later how such hierarchies can be queried in a straightforward manner.

Note that in these relations, we do not need an optional number attribute for resource types. Therefore, we use role_ref, machine_type_ref, and space_type_ref here.

4.6. Other mapping relations between elements

Above we saw two kinds of relations, which are sub-elements of the relations elements. Three other relations are required to associate a user with a role, a space with a space type, and a machine with a machine type. This is accomplished by use of user_role_mapping, space_space_type_mapping and machine_-machine_type_mapping elements. These elements are sub-elements of relations. These mappings would allow finding a user with a sepcific role, a space (e.g., a room) of a particular type, say a conference room, or a machine of a particular type, say a PC. The example below illustrates these mappings.

```
<relations><user_role_mapping>
      <user_ref name="u.b_jones"/>
      <role_ref name="r.associate_professor"/>
</user_role_mapping><space_space_type_mapping>
      <space_ref name="s.room_301"/>
      <space_type_ref name="st.conference_room"/>
</space_space_type_mapping><machine_machine_type_mapping>
      <machine_ref name="m.pc_1"/>
      <machine_type_ref name="mt.pc"/>
</machine_machine_type_mapping></relations>
```

4.7. Modeling Availability of Resources

Another feature of the model is handling availability. Availability is a sub-element of relations. The status of any resource can thus be represented with availability sub-elements as follows.

```
<relations><availability start_time="05-20-02:0800"
      end_time="05-20-02:1700" status="available">
      <user_ref name="u.b_jones"/>
      <machine_ref name="m.pc_1"/>
</availability></relations>
```

The above simply describes that both Bob Jones and PC 1 are available from 8 AM to 5 PM on May 20, 2002. The status field may have other values such as reserved, in-use, etc. More such elements can be added for tracking the status of other resources.

4.8. Function Separation and Binding

Function separation and binding requirements are specified using the new function attribute of a task. Thus, if a post doc may not be reviewed and appointed by the same individual, this could be expressed as follows.

```
<task name="review_post_doc"><resource_binding>
      <role_nref name="r.professor"/>
</resource_binding><resource_binding>
      <role_nref name="r.associate_professor"/>
</resource_binding></task>
<task name="appoint_post_doc"
      function="diff_user(review_post_doc)"><resource_binding>
      <role_nref name="r.professor"/>
</resource_binding><resource_binding>
      <role_nref name="r.associate_professor"/>
</resource_binding></task>
```

The above description says that though professors and associate professors may review and appoint post docs, the specific individuals of both tasks must not be the same. Similarly, other functions for this attribute could be:

- same_user(task name): The same user who did another task must do current task
- diff_role(task name, …, task name): The role that performed one or more other tasks must not perform current task
- same_role(task name): The role that performed another task must perform current task
- higher_role(task name, …, task name): The role that performs current task should be higher in the role hierarchy than the one which performed one or more other tasks.
- lower_role(task name, …, task name): The role that performs current task should be lower in the role hierarchy than the one which performed one or more other tasks.
- not(user name, …, user name): prohibit a list of users.

If two tasks are to occur in a sequence, then the later task should contain the function attribute that relates the two tasks; however, if two tasks are in parallel, then any one of them can contain the function attribute.

It is also possible to combine multiple functions. In the above example it would be reasonable to say,

```
function="diff_user(review_post_doc), same_role(review_post_doc)"
```

This would imply that a post doc should be appointed by somebody in the same role as the role of the individual who performed the review, however, it must be a different individual.

Finally, some functions may have multiple arguments. Thus, if the function attribute appointing post docs has a value of “higher_role(review_post_doc, check_post_doc)”, it means that a post doc can only be appointed by somebody who occupies a higher role in the inheritance hierarchy than the ones who reviewed him and checked his credentials.

5. DISCUSSION AND RELATED WORK

We described our approach for creating a detailed organization model. Although a detailed discussion of the implementation

architecture is beyond the scope of this paper, we expect that a workflow engine will have to be suitably modified to incorporate organizational policies. Thus, when a task has to be assigned, the workflow engine may consult a policy module to find the appropriate resource bindings. An assignment of a task may thus involve informing Bob, Sue and Chen that they have a meeting scheduled for them at 3 PM on May 31 in conference room C105, as opposed to informing each user that a task is waiting for them and leaving the remaining details for them to figure out.

This approach has several interesting features. First, it is relatively succinct and we showed that it could capture a lot of useful information about organizational structure and policy. Clearly, there is always a trade-off between simplicity and expressive power, and our approach gives considerable expressive power within an intuitive and easy to learn framework. Secondly, the model is expressed in XML, which is understood very widely. This can help in inter-organizational interactions in various ways. Two organizations planning to collaborate can send their models to each other, and this information can then be used to enhance understanding and exchange. For example, it can be used to create inter-organizational teams, which is a prerequisite before collaboration can commence. Thirdly, the approach is also extendible. Although we expect the DTD to become a standard (say, like the X.500 standard for directories [8]), different industries or even groups of organizations may wish to expand the DTD. Thus, the organization DTD would be a least common denominator, but it may be extended by adding attributes and elements. For example, additional attributes like salary could be added to the user attribute, and tasks may be assigned based on a least-salary-first rule to eligible employees. Similarly, other elements can be added under resource (e.g., raw materials in manufacturing) based on the needs of the organization. In this way, various enhancements are possible.

Research on organizational modeling in workflow systems has been relatively limited. The RBAC (Role-Based Access Control) model [11,12,14,15] represents one approach for determining suitable users for a task. The salient features of RBAC are that permissions are associated with roles and users are made members of roles, thereby acquiring the associated permissions. RBAC models are useful but they have limitations, in particular they are primarily permission oriented, from a security point of view, and neglect other aspects of the organization such as resource availability. They also do not handle delegation very well and lack general querying capabilities. Bussler and Jablonski [6] have also done some pioneering work in pointing out many limitations of workflow systems in modeling policy and organizational issues. Several researchers have developed so-called meta models, i.e., object models describing the relation between workflow concepts, which include work allocation aspects. Consider for example some of the papers by Zur Mühlen [18,19]. Other authors have examined the issues of security in workflow systems from various different perspectives. For a partial list of these efforts and their salient features see [4,5,7,13,16].

6. CONCLUSIONS

Support for routing of workflows and integrating the workflows with the organizational model are key to successful implementation of various E-Commerce applications. Many researchers have noted the need for such an integration [6,18,19]. However, to the best of the authors' knowledge, the present work represents the first effort to describe a detailed organizational model in XML and to integrate it with a workflow routing model also in XML, in order to tie the two together in a common framework. The organization model described by the organization DTD can be used independently, and it also complements the XRL process DTD very well. In particular, the DTD described in this paper captures four important elements in an organization in a systematic manner. These elements are: resources (such as users, machines and space), resource types (such as managers, vice-presidents, PCs, conference rooms, etc.), collections (such as teams consisting of resources and/or resource types), and various relationships between resources and resource types. We also showed how one can model issues of resource availability and express advanced constraints for separation and binding of duties. Therefore, we believe this framework is comprehensive.

The advantages of this approach are that it will lead to better, “organizationally aware” workflow systems, and thereby increase chances of successful deployment. Moreover, it will also facilitate interoperability, both between organizations, and across departments of the same organization. Finally, it will improve the ability to query an organizational model using an XML query language such as XPath [10] or XQuery [9]. In future work, we expect to implement this model using real data from an organization and measure its effectiveness. Further, a prototype implementation for a workflow engine based on the process DTD already exists [2], but it lacks an organizational model. Hence, a next logical step is to enhance it by incorporating the organization model. It should also be noted that organizational models contain valuable company information that many companies may like to keep confidential or share in a limited way. Hence, there is a need for research on views that allow a company to expose its model selectively depending upon how much it trusts the trading partner. Lastly, although two companies may have the same unde rlying DTD, yet there are differences in syntax. While one organization may have a role called director, another may refer to the same role as supervisor. Hence, methods may be required for describing the characteristics of roles in still more detail and mapping them appropriately into one another.

7. REFERENCES

[1] Aalst, W.M.P. van der, A.H.M. ter Hofstede, B. Kiepuszewski, en A.P. Barros. "Advanced Workflow Patterns," In O. Etzion en P. Scheuermann, editors, 7th International Conference on Cooperative Information Systems (CoopIS 2000), volume 1901 of Lecture Notes in Computer Science, pages 18-29. Springer -Verlag, Berlin, 2000.

[2] W.M.P. van der Aalst and A. Kumar. XML Based Schema Definition for Support of Interorganizational Workflow. Information Systems Research, 2002 (to appear).

[3] W.M.P. van der Aalst, H.M.W. Verbeek, and A. Kumar. XRL/Woflan: Verification of an XML/Petri-net based language for inter -organizational workflows (Best paper award). In K. Altinkemer and K. Chari, editors, Proceedings of the 6th Informs Conference on Inform ation Systems and Technology (CIST -2001), pages 30-45. Informs, Linthicum, MD, 2001.

[4] Atluri, V.; and Huang W.K. An extended Petri net model for supporting workflows in a multilevel secure environment. Proceedings of the 10th IFIP WG 11.3 Workshop on Database Security (1996), 199-216.

[5] Bertino, E.; Ferrari, E.; and Atluri, V. Specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security, 2, 1 (February 1999), 65-104.

[6] Bussler, C.; and Jablonski, S. Policy resolution for workflow management. Proceedings 28th Hawaii International Conference on System Sciences Conference (January 1995).

[7] Castano, S.; and Fugini, M. Rules and patterns for security in workflow systems. Proceedings of the IFIP WG 11.3 Working Conference on Database Security (August 1998), 59-74.

[8] Chadwick, D. W., Understanding X.500 - The Directory, © 1994, 1996. Available at http://www.isi.salford.ac.uk/staff/dwc/Version.Web/Contents.htm

[9] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and M. Stefanescu. XQuery: A Query Language for XML. W3C Working Draft. Available at http://www.w3.org/TR/xquery/, 2000.

[10] J. Clark, S. DeRose (ed.) "XML Path Language (XPath) Version 1.0", W3C , November 1999. (http://www.w3.org/TR/xpath)

[11] Ferraiolo, D. F.; and Kuhn, D.R. Role-Based Access Control. In 15th National Computer Security Conference. NIST/NSA, 554-563, 1992.

[12] Ferraiolo, D. F.; Cugini, J.; and Kuhn, D.R. Role-based access control: features and motiv ation. Proceedings of the 11th Annual Computer Security Applications Conference (1995). IEEE Computer Society Press, 241-248.

[13] Nyanchama, M.; and Osborn, S.L. The role graph model and conflict of interest. ACM Transaction on Information and System Security, 1 (1999), 3-33.

[14] Sandhu, R.S.; Coyne, E.J.; Feinstein, H.L.; and Youman, C.E. Role-based access control models. IEEE Computer, 29, 2 (1996), 38-47.

[15] Sandhu, R.S.; Bhamidipati V.; and Manuawer, Q. The ARBAC97 model for role-based administration of roles. ACM Transactions on Information and System Security, 2, 1 (February 1999), 105-135.

[16] Simon, R. and Zurko, M. E. Separation of duty in role-based environments. Proceedings of the 10th Computer Security Foundations Workshop (1997), 183-194.

[17] H. S. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. W3C Working Draft, Apr. 2000. http://www.w3.org/TR/xmlschema-1/.

[18] Zur Mühlen, M. Resource modeling in workflow applications. Proceedings of the 1999 Workflow Management Conference (November 1999), 137-153.

[19] Zur Mühlen, M. Evaluation of workflow management systems using meta models. Proceedings of the 32nd Hawaii International Conference on System Sciences (1999).

APPENDIX: THE ORGANIZATION DTD (XRL_ORG.DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT organization (resources?, resource_types?, collections?, relations?)>
<!ELEMENT resources ((user | machine | space)*)>
<!ELEMENT user EMPTY>
<!ATTLIST user
    name ID #REQUIRED first_name CDATA #IMPLIED
    last_name CDATA #IMPLIED department IDREF #IMPLIED
    e-mail CDATA #IMPLIED login_id CDATA #IMPLIED
    address CDATA #IMPLIED phone CDATA #IMPLIED
    skills CDATA #IMPLIED>
<!ELEMENT machine EMPTY>
<!ATTLIST machine
    name ID #REQUIRED description CDATA #IMPLIED
    number CDATA #IMPLIED contact CDATA #IMPLIED
    address CDATA #IMPLIED phone CDATA #IMPLIED>
<!ELEMENT space EMPTY>
<!ATTLIST space
    name ID #REQUIRED building_number CDATA #IMPLIED
    room_number CDATA #IMPLIED description CDATA
        #IMPLIED
    capacity CDATA #IMPLIED contact CDATA #IMPLIED
    address CDATA #IMPLIED phone CDATA #IMPLIED>
<!ELEMENT resource_types ((role | machine_type | space_type)*)>
<!ELEMENT role EMPTY>
<!ATTLIST role name ID #REQUIRED description CDATA #IMPLIED>
<!ELEMENT machine_type EMPTY>
<!ATTLIST machine_type name ID #REQUIRED description CDATA #IMPLIED>
<!ELEMENT space_type EMPTY>
<!ATTLIST space_type name ID #REQUIRED description CDATA #IMPLIED>
<!ELEMENT user_ref EMPTY>
<!ATTLIST user_ref name IDREF #REQUIRED>
<!ELEMENT machine_ref EMPTY>
<!ATTLIST machine_ref name IDREF #REQUIRED>
<!ELEMENT space_ref EMPTY>
<!ATTLIST space_ref name IDREF #REQUIRED>
<!ELEMENT role_ref EMPTY>
<!ATTLIST role_ref name IDREF #REQUIRED>
<!ELEMENT machine_type_ref EMPTY>
<!ATTLIST machine_type_ref name IDREF #REQUIRED>
<!ELEMENT space_type_ref EMPTY>
<!ATTLIST space_type_ref name IDREF #REQUIRED>
<!ELEMENT role_nref EMPTY>
<!ATTLIST role_nref name IDREF #REQUIRED number CDATA #IMPLIED>
<!ELEMENT machine_type_nref EMPTY>
<!ATTLIST machine_type_nref name IDREF #REQUIRED number CDATA
    #IMPLIED>
<!ELEMENT space_type_nref EMPTY>
<!ATTLIST space_type_nref name IDREF #REQUIRED number CDATA
    #IMPLIED>
<!ELEMENT collections (named_collection*)>
<!ELEMENT named_collection (concrete_collection | typed_collection |
    mixed_collection)>
<!ATTLIST named_collection
    name ID #REQUIRED description CDATA #IMPLIED>
<!ELEMENT named_collection_ref EMPTY>
<!ATTLIST named_collection_ref
    name IDREF #REQUIRED>
<!ELEMENT concrete_collection ((user_ref | machine_ref | space_ref)*)>
<!ELEMENT typed_collection ((role_nref | machine_type_nref |
    space_type_nref)*)>
<!ELEMENT mixed_collection ((user_ref | machine_ref | space_ref | role_nref |
    machine_type_nref | space_type_nref)*)>
<!ELEMENT relations ((user_role_mapping | machine_machine_type_mapping |
    space_space_type_mapping | can_inherit | can_delegate | availability)*)>
<!ELEMENT user_role_mapping (user_ref, role_ref)>
<!ELEMENT machine_machine_type_mapping (machine_ref, machine_type_ref)>
<!ELEMENT space_space_type_mapping (space_ref, space_type_ref)>
<!ELEMENT can_inherit (role_ref, role_ref)>
<!ATTLIST can_inherit transitive_flag CDATA #IMPLIED restrictions CDATA
    #IMPLIED>
<!ELEMENT can_delegate ((role_ref | user_ref), (role_ref | user_ref))>
<!ATTLIST can_delegate transitive_flag CDATA #IMPLIED restrictions CDATA
    #IMPLIED>
<!ELEMENT availability ((user_ref | machine_ref | space_ref)*)>
<!ATTLIST availability
    start_time CDATA #REQUIRED end_time CDATA #REQUIRED
    status CDATA #REQUIRED reserved_for CDATA #IMPLIED>
```