Policy-Based Resource Management

Yan-Nong Huan[1] and Ming-Chien Shan[2]

[1]Oracle Corporation, 600 Oracle Parkway, Redwood Shores, California 94065
yahuang@us.oracle.com
[2]Hewlett-Packard Laboratories, 1501 Page Mill Road, 1U-4A, Palo Alto,
California 94304, shan@hpl.hp.com

Abstract. This paper proposes a new method to handle policies in Resource Management of Workflow Systems. Three types of policies are studied including qualification, requirement and substitution policies. The first two types of policies map an activity specification into constraints on resources that are qualified to carry out the activity. The third type of policies intends to suggest alternatives in cases where requested resources are not available. An SQL-like language is used to specify policies. Policy enforcement is realized through a query rewriting based on relevant policies. A novel approach is investigated for effective management of large policy bases, which consists of relational representation of policies and efficient retrieval of relevant policies for a given resource query.

Keywords. Workflow, Resource, Policy, Interval-Based, Query Rewriting.

1. Introduction

An information system is composed of a database system and one or many applications manipulating the database. The database system is a common data repository shared among multiple applications. Besides data, people sometimes move components which seemly belong to applications into the database, so that multiple applications can share common components. In other words, the common components become part of the database's semantics. Active databases are an example of such kind, where dynamic characteristics of data are pushed down to the database, so the data can always behave the same way no matter what applications are.

We are interested in Workflow Management Systems (WFMS) [5], and particularly, in Resource Management (RM) [6] of WFMS. A WFMS consists of coordinating executions of multiple activities, instructing who (resource) do what (activity) and when. The „when" part is taken care of by the workflow engine which orders the executions of activities based on a process definition. The „who" part is handled by the resource manager that aims at finding suitable resources at the run-time for the accomplishment of an activity as the engine steps through the process definition.

Resources of different kinds (human and material, for example) constitute the information system of our interest, their management consists of resource modeling and effective allocation upon users' requests. Since resource allocation needs to follow certain general guidelines (authority, security, for example) - no matter who or what application issues requests: so those general guidelines are better considered as part of the resources' semantics. That is the reason why we are interested in resource

M. Jarke, A. Oberweis (Eds.): CAiSE'99, LNCS 1626, pp. 422-428, 1999.

policy management in RM. Resource policies are general guidelines every individual resource allocation must observe. They differ from process specific policies which are only applied to a particular process. The policy manager is a module within the resource manager, responsible for efficiently managing a (potentially large) set of policies and enforcing them in resource allocation.
We propose to enforce policies by *query rewriting*. A resource query is sent to the policy manager where relevant policies are first retrieved, then either additional selection criteria are appended to the initial query (in the case of requirement policies) or a new query is returned (in the case of substitution policies). Therefore, the policy manager can be seen as both a *regulator* and a *facilitator* where a resource query is either „polished" or given alternatives in a controlled way before submitted for actual resource retrieval. By doing so, returned resources can always be guaranteed to fully comply with the resource usage guidelines.

1.1 Design Goals for the Policy Management

Technical issues involved in this study can be presented as the following set of design goals.
1. A simple policy model allowing users to express relationships between an activity and a resource that can be used to carry out the activity;
2. A Policy Language (PL) allowing users to *define* policies; PL must be easy to use and *as close as possible* to SQL;
3. Resource query *enhancement/rewriting*: a major functionality of the policy manager is to enforce policies by enhancing/rewriting the initial resource query;
4. *Efficient* management of policy base: retrieving relevant policies applicable to a given resource query may become time-consuming when dealing with a large policy base, strategies are to be devised to achieve good performance.

1.2 Paper Organization

The rest of the paper is organized in the following way. In Section 2, we give a brief overview of our resource manager, with the purpose of showing how the present research is positioned in its context. In Section 3, three types of policies for resource management are presented and the policy language is illustrated with examples. Some conclusive remarks are drawn in Section 4.

2. Context

In this section, we briefly discuss the context of the present research.

2.1 Architecture

Two main components exist in our Resource Manager (Figure 1). One is the resource manager per se, responsible for modeling and managing resources; the other is the policy manager allowing the user to manage policies. Three interfaces are offered, each obviously requiring a different set of access privileges. The policy language interface allows one to insert new policies and consult existing ones. With the resource definition language interface, users can manipulate both meta and instance

resource data. Finally, the resource query language interface allows the user to express resource requests. Upon receiving a resource query, the query processor dispatches the query to the policy manager for policy enforcement. The policy manager first rewrites the initial query based on *qualification policies* and generates a list of new queries. Each of the new queries is then rewritten, based on *requirement policies*, into an enhanced query. The enhanced new queries are finally sent to the resource manager for resource retrieval. In the cases where none of the requested resources is available, the *initial* query is re-sent to the policy manager which, based on *substitution policies*, generates alternatives in the form of queries. Each of the alternative queries is treated as a new query, therefore has to go through both *qualification* and *requirement* policy based rewritings. Then, the policy manager once again sends a list of resource queries to the resource manager. If no relevant resources are found against the rewritten alternative queries, notify the user of the failure. Bear in mind that substitution policies should *not* be used *transitively*.

2.2 Resource and Activity Models

A role is intended to denote a set of capabilities, its extension is a set of resources sharing the same capabilities. In this regard, a role can be seen as a resource type. The resource hierarchy shows resources organized into roles whilst the activity hierarchy describes the classification of activity types.

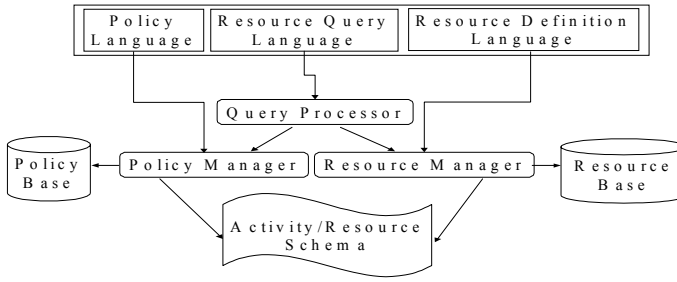

Figure 1: Architecture

Figure 2 shows an example of resource and activity hierarchies. A resource type as well as an activity type is described with a set of attributes, and all the attributes of a parent type are inherited by its child types. In addition to the resource classification, the resource manager holds relationships among different types of resources.

Two possible relationships between resources are exemplified in Figure 3. Note that, like attributes, relationships are inherited from parent resources to child resources. Views may be created on relationships to facilitate query expressions. For example, ReportsTo(Emp, Mgr) is defined as a join between BelongsTo(Employee, Unit) and Manages(Manager, Unit) on the common attribute Unit.

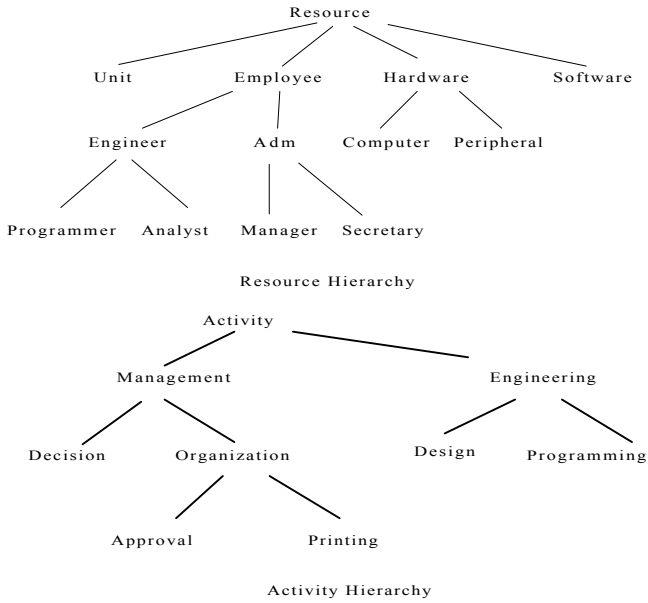

Figure 2: Resource and Activity Classifications

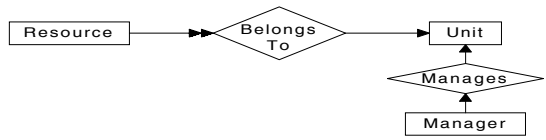

Figure 3: Entity-Relationship Model of Resources

2.3 Resource Query Language

Users can use the resource query language (RQL) to submit resource requests to the resource manager. The language is composed of SQL *Select* statements augmented with activity specifications.

```
Select   ContactInfo
From     Engineer
Where    Location = 'PA'
For      Programming
With     NumberOfLines = 35000 And Location = 'Mexico'
```

Figure 4: Initial RQL Query

```
Qualify   Programmer
For       Engineering
```

Figure 5: Qualification Policy

The query in Figure 4 requests ContactInfo of *Engineer* located in 'PA', for activity *Programming* of 35,000 line code and of location 'Mexico'. Since a resource request

is always made upon a known activity, the activity can and should be fully described; namely, each attribute of the activity is to be specified.

3. Policy Model and Language

Three types of policies are considered: qualification policies, requirement policies and substitution policies.

3.1 Qualification Policies

A qualification policy states a type of resources is *qualified* to do a type of activities. The policy in Figure 5 states the resource type *Programmer* can do the activity type *Engineering*. Sinceresources and activities are partially ordered, we allow qualification policies to be inherited from parent resources or activities to their children. Consider a general qualification policy „*Qualify* R *for* A", what this policy really means is any sub-type resource of R (including R itself) is qualified to do any sub-type activity of A (including A itself). All qualification policies in the policy base are *Or-related*, and they obey the *Closed World Assumption* (CWA). Namely, if the policy in Figure 5 is the only policy in the policy base, we may assume no resource types other than *Programmer* can do activity *Engineering*.

3.2 Requirement Policies

A requirement policy states that if a *resource* is chosen to carry out an *activity* with specified characteristics, the *resource* must satisfy certain conditions. Therefore, it expresses a necessary condition for a *resource* type and an *activity* type. All requirement policies in the policy base are *And-related*.
Here are examples of requirement policies:

Require	*Programmer*
Where	Experience > 5
For	*Programming*
With	NumberOfLines > 10000

Require	*Employee*
Where	Language = 'Spanish'
For	*Activity*
With	Location = 'Mexico'

Figure 6: Requirement Policies

The first policy in states that if a *Programmer* is chosen to carry out activity *Programming* of more than 10,000 line code, it is required that the *Programmer* have more than 5 year experience.
Given that both the set of resources and the set of activities are partially ordered, the scope of a requirement policy can stretch over resources and activities which are sub-types of the resource and the activity explicitly mentioned in the policy. For example, the second policy in Figure 6 requires the *Employee* be Spanish speaking if (s)he is engaged in activity *Activity* located in Mexico. Since both *Employee* and *Activity* have sub-types in their respective hierarchies (Figure 2), the policy is actually applicable to any pair of resource and activity as long as the resource is a sub-type of *Employee* (including *Employee* itself) and the activity is a sub-type of *Activity*

(including *Activity* itself). This gives a great deal of flexibility to expressing requirement policies. The syntax of a general requirement policy is as follows:

```
Require  R
Where    <Where>
For      A
With     <With>
```

Figure 7: General Requirement Policy

```
Substitute   Engineer
Where        Location = 'PA'
By           Engineer
Where        Location = 'Cupertino'
For          Programming
With         NumberOfLines < 50000
```

Figure 8: A Substitution Policy

There, *R* is a resource type and *A* is an activity type. <Where> is a SQL *where* clause which can eventually include nested SQL *select* statements. <With> is a restricted form of SQL *where* clause in which no nested SQL statements are allowed. Some more complex policy examples follow,

```
Require Manager
Where   ID = (
            Select   Mgr
            From     ReportsTo
            Where    Emp = [Requester]
        )
For     Approval
With    Amount < 1000
```

```
Require Manager
Where   ID = (
            Select       Mgr
            From         ReportsTo
            Where        level = 2
            Start with   Emp = [Requester]
            Connect by   Prior Mgr = Emp
        )
For     Approval
With    Amount > 1000 And Amount < 5000
```

Figure 9: Complex Requirement Policies

Both policies in Figure 9 relate resource *Manager* to activity *Approval*. The first states that if the amount requested for approval is less than $1,000, the authorizer should be the manager of the requester. The second policy (a hierarchical sub-query is used) requires that the authorizer be the manager's manager if the requested amount is greater than $1,000 and less than $5,000. Two points are worth mentioning,

1. Nested SQL statement can be used to construct more complex selection criteria.
2. Attributes of the activity can be referenced in constructing selection criteria. To distinguish an attribute of the activity from that of the resource, the former is enclosed between [and]. In the examples of Figure 9, Requester is an attribute of activity *Approval*.

3.3 Substitution Policies

A substitution policy is composed of three elements: a *substituting resource*, a *substituted res*ource and an *activity*; each eventually augmented with descriptions. It states that the substituting resource can replace the substituted resource in the unavailability of the latter, to carry out the activity. Multiple substitution policies are *Or-related*. The policy in Figure 8 states that *Engineers* in PA, in their unavailability, can be replaced by *engineers* in Cupertino to carry out activity *Programming* of less than 50,000 line code. Similar to the requirement policy, the scope of a substitution policy can stretch over resources and activities which are sub-types of the substituted

resource and the activity mentioned in the policy. Therefore, the policy in Figure 8 may eventually be applicable to a query looking for a *Programmer* for activity *Programming*.

4. Conclusion

We studied several issues related to resource policies in Workflow Systems. A policy language was proposed allowing users to specify policies of different types. To enforce the policy, a resource query is first rewritten based on relevant policies, before submitted to the resource manager for actual retrieval. The originality of the present work is on the resource policy model, the policy enforcement mechanism and policy management techniques including relational representation of, and efficient access to, a large policy set. A prototype was implemented in Java on NT 4.0, with experimental policies managed in an Oracle database.

References

[1] M. Blaze, J. Feigenbaum and J. Lacy, „Decentralized Trust Management“, Proc. of IEEE Symposium on Security and Privacy, Oakland, CA, May 1996.
[2] C. Bufler, „Policy resolution in Workflow Management Systems“, Digital Technical Journal, Vol. 6, No. 4, 1994.
[3] C. Bufler and S. Jablonski, „Policy Resolution for Workflow Management Systems“, Proc. Of the Hawaii International Conference on System Sciences, Maui, Hawaii, January 1996.
[4] Desktop Management Task Force, „Common Interface Model (CIM) Version 1.0 (Draft)“, December 1996.
[5] J. Davis, W. Du and M. Shan, „OpenPM: An Enterprise Process Management System“, IEEE Data Engineering Bulletin, 1995.
[6] W. Du, G. Eddy and M.-C. Shan, „Distributed Resource Management in Workflow Environments“, Proc. of Database Systems for Advanced Applications (DASFAA), Melbourne, Australia, April, 1997.