Policy Resolution in Workflow Management Systems

by Christoph J. Bussler

ABSTRACT

One crucial function of a workflow management system (WFMS) is to assign tasks to users who are eligible to carry them out. Except in simple workflow scenarios, roles such as secretary and manager are not a sufficient basis for determining eligibility. Additionally, WFMSs are deployed not only in group settings by small companies but also worldwide by large enterprises. Since local laws and business policies have to be followed, task assignment policies for the same task generally differ from country to country and, therefore, must be specified locally. The Policy Resolution Architecture (PRA) model provides more generality and expressiveness than role models do and at the same time supports the independent specification of task assignment policies in different parts of an enterprise. PRA can be used to model arbitrary organization structures and to define realistic task assignment (eligibility) rules by means of precisely defined organizational policies. Thus, PRA provides real-world organizations with a precise, simple means of expressing their complex task assignment policies.

INTRODUCTION

A workflow management system (WFMS) is a software system that manages the flow of work between participants or users according to formal specifications of business processes called workflows. A workflow specifies tasks to be performed and their execution order. Additionally, a workflow specification defines the internal flow of data between tasks as well as all applications required to carry out the tasks. For example, a travel expense reimbursement workflow specifies the tasks of filling, checking and signing a form, and reimbursing an amount. This workflow specifies that the form must be signed before an amount is reimbursed. The workflow specification also defines the flow of the expense form between tasks and the required spreadsheet application. Finally, for each task of a workflow, some rule has to be in place that specifies the users who are eligible to carry out the task. This set of eligible users is determined at run time, and the task is subsequently assigned to them.

One of the key issues in successfully deploying WFMSs in an enterprise is the correct assignment of a given task to eligible users. An eligible user is one who is capable of and responsible for carrying out an assigned task. This distinction is important because not every user who is capable of performing a task is necessarily responsible for it. The successful completion of a

task, however, often requires that crucial, irreversible decisions be made by a person who is responsible for the task. Making the right decisions and then carefully and responsibly carrying out the task is essential to conducting business successfully.

The criteria used to determine an eligible user for a task are manifold. A user must have a specific set of capabilities to be able to carry out the task. Additionally, the position of a user in the organization hierarchy and/or the reporting structure of the organization can determine if the user is responsible for the task. Furthermore, limits placed on a user's decision-making authority can affect eligibility. For example, not every salesperson is authorized to accept an order that leads to a significant increase in manufacturing output. Such an order requires special attention and internal coordination by a senior sales representative. When cost-optimized task assignments are made, the experience of the user as well as the user's skill set has to be taken into consideration. Highly experienced users are in most cases expensive resources, but usually they can complete tasks faster than users with average experience. Although users with either level of experience may have sufficient experience to carry out a specific task, if deadlines are involved or extreme caution with respect to quality is necessary, a highly experienced user might be appropriate. In such cases, the additional cost would be justified.

The previous discussion demonstrates the necessity of a precise definition of eligible users for a given task. Such a definition, i.e., set of task assignment rules, should contain all the criteria used to determine eligible users for the task. Early in the development of Digital's ObjectFlow WFMS product, the concept of roles was considered sufficient to model the assignment of tasks to users.[1] However, an analysis of distributed enterprise-wide production workflows clearly showed that using roles as the only assignment mechanism has limited value in determining eligibility.[2] The need for a far more expressive, general, and flexible approach became obvious. The analysis also revealed that workflows are often reused in different parts of an enterprise. A prominent example is the travel expense reimbursement workflow, which is discussed throughout this paper. Although a workflow is reused, however, the task assignment policies may differ greatly in the various parts of an enterprise. This difference is due to the need to adhere to local laws and/or to business-related deviations from the general rules.

Based on the requirements derived from several case studies of complex workflows, the Policy Resolution Architecture (PRA) was developed to provide a comprehensive way of specifying task assignment rules.[2] To support the fact that different parts of an organization may require different assignment rules, PRA and its implementation were designed as separate components. PRA incorporates three major elements and thus provides

- Concepts that enable the modeling of any organization structure (not just roles and groups) without prescribing structures that are application dependent.

- Task assignment rules as entities in themselves, separate from a workflow specification. This makes it possible for each of the different parts of an enterprise to have its own set of task assignment rules for the same workflow.

- A language that enables the explicit specification of organization schemas and task assignment rules. Specifications are processed by a component called the policy resolution engine during workflow execution.

Before explaining PRA in detail and providing the rationale for its development, the paper introduces the key concepts of workflow management. This introduction presents a seemingly simple workflow that specifies travel expense reimbursement, which is later used to introduce the design objectives of PRA. Note that a real travel expense reimbursement workflow for production is by far more complex than the example used in this paper. A large distributed enterprise endeavors to reuse the same workflow in all of its parts because reuse facilitates administration and leverages the development investment. At the same time, such an enterprise probably sponsors numerous business trips, which makes the travel expense reimbursement workflow an excellent candidate to use as an example.

WORKFLOW MANAGEMENT

This section introduces a model of workflow management. The discussion begins with a survey of preliminary work. The survey suggests the motivation for workflow management and enumerates some areas in which workflow management is deployed. The key concepts of the workflow model are then used to model a workflow example, i.e., the travel expense reimbursement workflow. The section concludes with a definition of workflow management systems.

Historical Survey

Looking back in history reveals that workflow management has many roots. The most important are office automation, software process management, manufacturing, and transaction processing. The following short survey of achieved results is given to help the reader understand the motivation for workflow management. The discussion also explains the choice of workflow management concepts. The list of previous and related works indicates the range of literature that exists.

Office Automation. One of the primary roots of workflow management is undoubtedly office automation. Early research led to the development of models and tools to support office workers.[3-9] What emerged were not only desktop applications that imitate concepts such as in basket, out basket, forms, and documents but also models of the procedures that the office workers follow while doing their jobs.[10,11] Furthermore, systems were developed that execute the office procedures to actively manage the flow of work within offices.[12,13]

Software Process Modeling. A second major root of workflow management is software process modeling and execution.[14-25] The focus of research in this area is the automated support of software development processes. Concepts comprise process models like the waterfall model or the spiral model, deliverable code, installation and operation manuals, requirements documents, and test cases.[26,27]

Manufacturing. Traditionally, formalized procedures that are executed repeatedly are inherent to manufacturing, another root of workflow management. Manufacturing involves not only production processes but also preproduction procedures starting from, for example, the release of computer-aided design (CAD) drawings to the preparation of shop floor schedules.[28-31]

Transaction Processing. Another important area that influenced the development of workflow management is transaction processing. After the concept of atomicity, consistency, isolation, and durability (ACID) transactions was developed, researchers proposed more advanced transaction models for processing several interdependent tasks that must be transactional and recoverable.[32-39]

Coordination Theory, Enterprise Modeling, and Speech Act Theory. Another area of research that contributed to the idea of workflow management is coordination theory.[40,41] This area looks at processes as one form of coordination and tries to apply interdisciplinary research results to it. The research area of enterprise modeling focuses on the modeling of the whole multifaceted enterprise.[42-49] Enterprise activities are one part of an enterprise that drives the enterprise processes. The speech act theory is an attempt to model the conversation between humans.[50] Some research follows the direction that a workflow is an interwoven chain of speech acts.[51]

Early Application-independent Approaches. In addition to the application-specific roots of workflow management, early approaches that modeled processes independent of application areas provided motivation for workflow management.[52-54]

The term process appears in all the areas of work mentioned above. Also, all these research areas deal with data, e.g., documents, CAD drawings, and orders. Most approaches have some notion of subject or agent. The question arose among researchers, Does each area need its own definition of terms, modeling language, and execution mechanism, or is it possible to provide general concepts that need to be customized only for a specific area of application? This question triggered the development of the concept of workflow, whose goal it is to serve as the general and customizable concept.

Workflow Management Concepts

After the specific application semantics (e.g., documents, office workers, release procedures, and CAD drawings) have been abstracted, the basic concepts of workflow management can be distilled from the various approaches mentioned above. Although workflow management is independent of specific application semantics, it does support all the application areas cited. It provides an integrated set of underlying concepts that can be customized to model the semantics of each application area. Workflow management is analogous to relational database systems. Such systems know how to model and implement tables and how to process queries; however, they do not know about the specific concepts of an application area that are implemented by user-defined tables, e.g., addresses and orders.

The following list introduces the basic concepts of workflow management by enumerating the major aspects that make up a workflow specification:[14]

- Functional aspect. The functional aspect describes what has to be done, without saying how, by whom, and with which data. The functional aspect provides two concepts: elementary workflows and composite workflows. Elementary workflows are tasks that can be carried out by one person, program, or machine. For brevity, elementary workflows are called steps. Composite workflows bundle either elementary workflows or other composite workflows to higher-level tasks. In this way, a reuse hierarchy is built, since the bundled workflows may very well stand by themselves. Generally, these higher-level tasks can no longer be achieved by a single person, program, or machine but require several such entities. A workflow that bundles other workflows references them. As a naming convention, a workflow that is referenced by some other workflow is called a subworkflow. The referencing workflow is called the superworkflow. The topmost workflow of a reuse hierarchy is called the top-level workflow.

- Behavioral aspect. The behavioral aspect describes the

execution order of the subworkflows of a workflow. Constructs that describe the order include sequence, conditional branching, parallel branching, and the looping and/or joining of parallel or conditional execution paths.

- Informational aspect. The informational aspect is twofold: first, it describes the local variables of a workflow and the external data referenced; second, it describes the flow of data from subworkflow to subworkflow.

- Organizational aspect. The organizational aspect describes who is eligible to carry out a step. The "who" can be a human (e.g., an office worker), a program (e.g., a compiler in a software process), or a machine (e.g., a cell in a shop floor). The term user was chosen to represent all three. Most available WFMSs offer the concept of roles to model the organizational aspect. A role usually groups a set of users. At run time, tasks are assigned to roles and all users grouped by these roles are assigned the task. Although this method of task assignment is adequate for certain workflows such as departmental workflows, as shown later in the section Task Assignment in a Travel Expense Reimbursement Workflow, roles are not sufficient to handle workflows that are deployed in an enterprise-wide or international setting.

The literature discusses additional aspects, e.g., a historical aspect and a technological aspect.[55] The historical aspect is used to specify the kind of information to be stored in a historical database during the execution of a workflow, e.g., starting times or values of variables. Instead of having the default strategy of saving all data, the workflow specifies in the historical aspect only the important data that must be stored. The technological aspect allows the definition of which application program or programs are available to carry out a step. At run time, these application programs are made available to the user. In principle, it is not possible to enumerate all necessary aspects completely in advance. Depending on the application area to be modeled, additional aspects might appear and require support.

The paper now shows how the key concepts of workflow management can be applied, i.e., customized, to model a specific workflow type. The example used is a sample travel expense reimbursement workflow.

Travel Expense Reimbursement Workflow

Figure 1 shows the graphical representation of a simplified workflow for the reimbursement of travel expenses. (Examples of

workflow language can be found in the literature.[55,56]) The workflow consists of four steps: (1) fill, (2) check, (3) sign, and (4) reimburse. The graphical representation shows the functional aspect (task structure) as ovals and the behavioral aspect (control flow) as solid arrows. The informational aspect (data flow) is displayed as forms; dotted arrows indicate the direction of the flow of data. The organizational aspect is omitted since the paper will focus later on this topic. The technological aspect is represented by icons of the software applications that are available to carry out the steps. The historical aspect is represented by icons that symbolize logs in which information must be recorded.

[Figure 1 (Travel Expense Reimbursement Workflow) is not available in ASCII format.]

Step 1 of the travel expense reimbursement workflow, the fill step, enables a user to enter the relevant expenses incurred during a business trip into an electronic travel expense form. After a user has finished entering the data, validation must take place. The check step enables a user to look at the contents of the travel expense form. This user is prompted to validate the contents but cannot change entries. If the user who checks the form detects an error, the form is sent back to the user who initially filled it out, with a note that explains the reason for rejection. Otherwise, the form is forwarded to the next user who has to sign the form to approve the amount. After the sign step is complete, the amount can be reimbursed. The last step, reimburse, enables a user to add the amount spent to the next paycheck of the user who requested reimbursement.

This sample workflow is intentionally kept simple because beginning with the next section, the paper focuses solely on task assignment rules. In a real organizational setting, the workflow would involve more steps and additional execution paths. For example, a user who has to sign the form might detect an error. In this case, as in the check step, the form would be sent back to the user who initially filled it out.

Workflow Management Systems

Managing the flow of work among users is done by a software system called a workflow management system (WFMS). A WFMS contains all the specifications of the workflow types (e.g., a travel expense reimbursement or a capital equipment order) that are modeled and released for production. If a user issues a request to start a workflow (e.g., if, after a business trip, a traveler starts a travel expense reimbursement workflow), the WFMS creates an instance of the requested workflow type. Of course, more than one instance of the same workflow type can exist simultaneously. A WFMS assigns the steps of a workflow to users according to the specified order of the behavioral, functional, and organizational aspects.

In general, a WFMS performs the following actions to execute a workflow instance:

- Determine the next steps to be executed.
- Determine the eligible users for these steps.
- Assign steps to eligible users.
- Wait for the result of each step.
- Transfer the result back to the step's superworkflow and record the step as complete.

The WFMS repeats these actions until all steps of a workflow are executed.[55,57-59] This list of actions has to be slightly modified if, in addition to steps, a workflow contains composite workflows in its list of subworkflows. In this case, the subworkflow is not assigned to users and the list of actions is applied to each of the subworkflows.

Each user who can potentially be involved in a workflow is connected to a WFMS by a private worklist, which is a graphical representation of a list of steps assigned to the user. Each entry in a user's worklist represents a task the user is eligible to carry out. A user can participate in more than one workflow at the same time. Normally, the user is free to choose from the worklist any item on which to start. In well-designed systems, the WFMS automatically starts the application programs that the user will require to accomplish the work. In this way, the user can begin work immediately.

Almost all prototype implementations or product developments allow the modeling of the four main aspects described previously. The list of workflow management systems is growing rapidly, and references to relevant literature are readily available.[37,57-64] References to literature that describes the deployment of workflow management systems in an application area are rare, however.[51,61,65-67]

The reminder of the paper focuses on the organizational aspect of workflow management. The paper discusses the derivation of the requirements that concepts of this aspect must meet and then introduces PRA as the model whose concepts address the requirements. An analysis of the travel expense reimbursement workflow illustrates some of these requirements. Additional requirements are also described to provide a more complete set.

TASK ASSIGNMENT IN A TRAVEL EXPENSE REIMBURSEMENT WORKFLOW

The requirements that must be fulfilled by the concepts of the organizational aspect were derived from the travel expense

reimbursement workflow example, the author's project work experiences, and Marshak's "Characteristics of a Workflow System -- Mind Your P's and R's."[68] The following list describes task assignment rules for each step of the travel expense reimbursement workflow:

- Fill. The fill step can be executed by anyone in an organization who has the potential to travel. This assignment rule enables an employee to fill in a travel expense reimbursement form after a business trip. (An employee who did not travel can also fill in a form and claim expenses; however, the check and sign steps are intended to detect such misbehavior and to reject the form.) The user who fills in the form is referred to as the applicant and is known at run time.

- Check. The check step must be executed by a user who is able to play the role of secretary. To be able to validate the contents of the form, a user in this role is expected to know how a travel expense reimbursement form is structured and how to correctly fill in the form. This user is also expected to know the destination and the travel dates, and if the travel actually took place. Not all secretaries in an enterprise have this knowledge, but the secretary of the applicant's manager can be expected to know the information. This secretary usually plans the trip and often the meetings of the traveler. If the user who is able to play the role of secretary determines that the contents of the travel expense reimbursement form are sound, the form is forwarded to the next step; otherwise it is sent back to the applicant.

  The overall task assignment rule is therefore: Everyone who is able to play the role of secretary and reports to the same manager as the applicant is eligible to execute the check step. (Note that the term manager means a user who is able to play the role of manager.)

- Sign. The sign step has to be executed by a manager of the applicant because the manager normally has to approve spending by subordinates. Usually, there is only one user to whom the applicant reports and who is able to play the role of manager. If there are two such users, either can be responsible for signing the form and only one has to sign it.

  The overall task assignment rule is: Everyone who is able to play the role of manager and to whom the applicant reports is eligible to execute the sign step.

- Reimburse. The reimburse step must be executed by a financial clerk who is responsible for the group to which the applicant belongs.

The overall task assignment rule is: Everyone who is able to play the role of financial clerk and who is responsible for the applicant's group is eligible to execute the reimburse step.

The requirements thus far derived from the example are

- Organization structure dependencies. To select one user relative to another (e.g., a user playing the role of secretary reporting to a user playing the role of manager) requires describing the users, the roles, and the dependencies (relationships). This description is called an organization structure. An organization structure contains all organizational object types like "user," "group," or "role," and the relationships among them like "reports to" or "supervises." Given such a structure, users can be selected based on their relationships to others. Users can also be selected based on attributes such as their absence status (i.e., whether they are on vacation or on a business trip) or their workload.

- Historical access. In some cases, the eligible user for a step cannot be determined locally, and historical information is required. For example, determining the user who can play the role of manager in one step might require knowing which user started the workflow. Therefore, it must be possible to query a log of the history of a workflow to derive the information necessary to make task assignments.

The following are additional requirements:

- Data dependency. In the travel expense reimbursement example used in this paper, the manager to whom the sign step is assigned can sign for any amount. In other cases, however, this signatory power may have limitations. For instance, if the amount exceeds a certain value, a vice president and not the manager of the applicant must sign the travel expense reimbursement form. As this last example shows, task assignment may depend on data in the workflow.

- Delegation. A manager who is out of the office may want to delegate his/her tasks to keep business operations running smoothly. The appropriate task assignment rule would then have to be extended to incorporate the delegation of tasks. Depending on the status of the manager (e.g., on a business trip or on vacation), the work would be assigned to someone else (i.e., delegated). However, task assignment rules that incorporate delegation can be complex. Consider the situation in which a manager leaves on a business trip after work has already been assigned. In this situation (and also in the

case where a manager has an excessive amount of work to accomplish), the manager must be able to dynamically delegate some or all of the already assigned tasks. Further consider that a manager may want to delegate different types of tasks not to the same user but to different users, depending on the type of task. To avoid leaking information or making an inexpedient assignment, the task assignment rule must make sure that the target users are eligible to receive the delegated task assignment.

- Separation of duty. Some scenarios require a separation of duty, i.e., two tasks must be performed by different users. For example, in the transfer of a large amount of money, two managers must sign the transfer form to double-check the transaction. Regarding the travel expense reimbursement workflow, a user who fills out the claim form should not also sign it. Task assignment rules must ensure that there is a separation of duty.

- Responsibility. As previously stated, a subworkflow can be either a step or a group of steps that may be a reuse of building blocks for larger workflows. A second use of a composite workflow is to explicitly express responsibility for workflows. Sometimes an application domain requires a user to take responsibility for a set of tasks even though the user does not actually execute the tasks. For example, consider a workflow that implements the start of a new product development. The investment plan depends on the development plan, which is based on a market analysis. A manager or a vice president is usually responsible for these three complex tasks (market analysis, development plan, investment plan) but not involved in the detailed work. In a WFMS, this situation would be modeled as a workflow called Product Development Start, which contains the three complex tasks as subworkflows. The Product Development Start workflow could then be assigned to a manager or a vice president to model responsibility. The assignment to this user means only that the user must acknowledge the start of the assigned workflow and therefore accept responsibility for it. The assignment does not imply that the user has to perform the detailed work. Thus, a WFMS must be able to assign not only steps to users but also composite workflows.

- Early/late allocation. Often, the application semantics clearly indicates the single user who should execute a task. In such cases, the related task assignment rule (e.g., the role of manager of applicant) passes to this user at run time. In other scenarios, however, successful execution of a task requires some capability that more than one user possesses. This capability is often expressed through a role (e.g., financial clerk, which is

a role usually played by more than one user in large enterprises). In the single-user case, the task is assigned to that user regardless of the user's workload; this process is called early allocation. The user must carry out the task unless it is feasible to delegate it. In the multiple-user case, the task appears on the worklist of all users able to play the role. One user starts the task; in most cases, this user would not have the highest workload. Therefore, the final allocation of the task is made not by the WFMS but by the set of eligible users themselves. This process is called late allocation. In this case, if one user starts work on a step, the other users are no longer allowed to begin the task.[5,59] Subsequently, their assignment must be revoked. "Implementing Agent Coordination for Workflow Management Systems Using Active Database Systems" describes a general mechanism for handling the revocation of assignments.[69]

The travel expense reimbursement workflow is used in the following discussion about the limitations of roles as a basis for task assignment rules. These limitations influenced the major design objectives of PRA, which are then discussed.

Roles As Task Assignment Rules

As stated earlier, roles have limited use as task assignment rules. Applying the role concept to the task assignment rules introduced above illustrates the limitations. Certainly, the term role has many definitions. In this paper, a role is an abstraction of a set of users. The abstraction criteria are the set of capabilities of a user. Whether or not a particular user belongs to the set of users abstracted by a role is defined by an explicit relationship between a user and a role called the "plays" relationship. A user who has a plays relationship with a role has the capabilities defined by that role, i.e., the user is able to play the role. For example, if both Ann and Joe are users who are able to play the role of clerk, then each one has the capabilities defined by this role and each is capable of executing the task. A user might have a wide range of capabilities and be able to play several roles at the same time. E.g., a user might be able to play the role of employee and the role of manager simultaneously. Although this definition of role is not the only one, it is very common and often applied.[6,14,51,52,62,63,70,71]

For each task assignment rule that was introduced in the travel expense reimbursement example, a discussion follows about the extent to which roles support the requirements.

- Fill. The task assignment rule for the fill step is the only rule of the example that can be modeled completely with a role. Assume that every user is able to play the

role of employee. If the fill step is assigned to the role of employee, every user can execute the step, thus modeling exactly the task assignment rule of the fill step.

- Check. Assigning the check step to the role of secretary does not model the full semantics of the desired task assignment rule. Such an assignment models only the requirement that a user has to be able to play the role of secretary to carry out the step. The assignment does not model the additional requirement that only those users who report to the same manager as the applicant are eligible.

- Sign. Analogous to the situation in the check step, assigning the sign step to the role of manager does not model that only a user to whom the applicant reports is eligible but that any manager is eligible.

- Reimburse. Assigning the reimburse step to the role of financial clerk ensures only that the step is assigned to a capable user. The assignment does not fulfill the additional requirement that this user must also be responsible for the group to which the applicant belongs.

The discussion of the last three task assignment rules demonstrates two tightly coupled limitations of using roles to model requirements.

1. The concept of roles cannot express organizational dependencies, such as relationships between users (e.g., "reports to" and "responsible for"). It only relates users to roles by a plays relationship. Furthermore, roles do not provide a means of introducing additional objects of organization structures like "group" and "department." The only two objects the concept of roles provides are "role" and "user."

2. The concept of roles, therefore, does not provide a sufficiently sophisticated language to express, for instance, that a user not only has to play a certain role but also has to relate to some other user in a particular way (e.g., "reports to" a particular user).

In addition, the other requirements like historical access, delegation, and separation of duty cannot be modeled at all using roles.

To overcome these limitations, PRA introduces the concepts of organization schema and organizational policy and the Policy Definition Language. A brief introduction follows. Details are presented in the section Policy Resolution Architecture.

Organization Schema

One of the fundamental concepts of PRA is a freely definable organization schema. An organization schema contains all types of organizational objects and relationships that are available for modeling a particular organization. Figure 2a gives an example of an organization schema. If a defined schema is instantiated, it contains an organization structure. Since other objects besides roles are required to model an organization, relationships other than "plays" must be available. Some necessary additional relationships are "reports to," which relates two users, and "is responsible for" and "belongs to," which relate a user and a group. A freely definable organization schema, such as the one provided by PRA, allows designers to define roles as required by the workflow application.

[Figure 2 (Sample Organization Schema and Organization Structure for the Travel Expense Reimbursement Example) is not available in ASCII format.]

Such a freely definable organization schema may seem to be a luxury, and a fixed organization schema that provides the most relevant objects and relationships may seem sufficient. An analysis of various organization structures in different enterprises clearly shows, however, that a single organization schema is not adequate for all situations in which WFMSs can be deployed. An enterprise that deploys a schema in which the semantics of the modeled objects are fixed has to follow the semantics completely. Consequently, such a schema does not meet enterprise-specific needs.

Figure 2a shows a graphical representation of a sample schema for the travel expense reimbursement example. Although this schema may appear general and an adequate alternative to an all-embracing schema, it does not contain required organizational objects such as task forces with a limited life span, committees, and departments. Also, this sample schema does not consider objects or relationships necessary for modeling delegation and relocation of employees. Figure 2b displays a superficial organization structure, i.e., an instantiation of the schema. Objects like user and role are depicted as icons, and relationships are depicted as arcs and solid, dashed, and dotted lines between the icons.

Approaches that go beyond using roles as a basis for task assignment commonly provide organizational objects in addition to roles and users, usually group and/or department objects.[2,6,8,59,72] The literature contains evidence that the schemas and the task assignment rules are fixed and have to be used as they are. Additionally, these approaches do not separate the workflow from the workflow specification, which makes the reuse of a workflow in a different organizational setting very difficult.

Organizational Policies As Task Assignment Rules

A second fundamental PRA concept is that of an organizational policy, which up to this point has been called a task assignment rule. An organizational policy specifies all the eligible users for a task by stating the criteria a user must meet. These criteria can include a role or roles that a user has to be able to play and relationships that a user has to have with other users or groups.

Figure 3a shows an example of an informal organizational policy for the sign step. This organizational policy specifies that if the WFMS is to assign the sign step, it will assign the step to the manager of the applicant if the amount is less than $1,000. Otherwise, it will assign the step to the vice president responsible for the applicant's group. A more advanced rule would not fix the amount at $1,000 but would make this amount dependent on the authorization level of the manager, as illustrated in Figure 3b.

Figure 3 Informal Organizational Policies for the Sign Step of the Travel Expense Reimbursement Workflow

(a)

```
WORKFLOW TravelExpenseReimbursement
STEP     sign
CRITERIA IF amount < 1000
         THEN manager of applicant
         ELSE VP responsible for applicant's group
         ENDIF
```

(b)

```
WORKFLOW TravelExpenseReimbursement
STEP     sign
CRITERIA IF amount < authorization level of applicant's manager
         THEN manager of applicant
         ELSE VP responsible for applicant's group
         ENDIF
```

The Policy Definition Language is PRA's formal language for specifying organizational policies. Policies written in this language are precise and executable by an execution engine called the policy resolution engine. Each time the WFMS is about to assign a step, the system evaluates the corresponding organizational policy to determine the set of users who can execute the task.

POLICY RESOLUTION ARCHITECTURE

WFMSs operate in global, open, and distributed environments and in group, department, enterprise, and multiple-enterprise settings. The enterprise-level deployment of workflows is possible only if the underlying concepts and systems are developed appropriately. PRA is therefore based on several design principles that ensure a general approach that supports enterprise-level deployment.

Design Principles

The PRA design principles are reusability, security, generality, dynamics, and distribution.

Reusability. In the travel expense reimbursement example, the sign step was modeled to approve travel expenses. Other workflows, like capital equipment orders, can reuse the sign step for similar tasks, e.g., to approve an order. If an organizational policy were attached to the step type itself, this assignment rule would serve to determine eligible users independent of the workflow in which the step is reused. Viewed from an organizational perspective, however, the reuse of steps in different workflows requires several policies. For example, the signing of a travel expense reimbursement form is carried out by a manager of the applicant, whereas the signing of a capital equipment order for an amount that exceeds a certain value is carried out by an appropriate vice president. Therefore, the sign step in the context of a travel expense reimbursement workflow has an organizational policy that defines the manager of the applicant to be eligible, whereas the sign step in the context of the capital equipment order workflow has a different policy, one that defines an appropriate vice president as eligible for the task.

The observation that a policy for a step depends not only on the step itself but also on the workflow in which the step is reused led to the decision to make organizational policies objects in themselves, independent of a workflow specification. Organizational policies name not only the step in which they are used but also the surrounding workflow. The design of

organizational policies for a step depends on the context in which the step is to be reused.

As mentioned earlier, making organizational policies independent objects allows different organization structures to reuse a workflow. To achieve such reuse, each organizational setting has its own set of organizational policies for the workflow to be reused. These organizational policies are tailored to the specific needs and circumstances of the organizational setting.

Organizational policies can themselves be reused. Different steps may require the same set of eligible users, and, therefore, one policy would be sufficient for more than one kind of step (e.g., sign and fill) or for more than one use of the same kind of step. For example, a manager signs not only travel expense forms but also capital equipment orders. In both workflows, the organizational policy that defines the manager of the applicant depends on the authorization level. Both workflows can reuse the sign step, as can be seen in the policy shown in Figure 4a. If the authorization level depends on the workflow, the policy changes to take into consideration the specific kind of workflow, as shown in Figure 4b.

Figure 4 Informal Organizational Policies Showing Reuse of the Sign Step

(a)

```
WORKFLOW TravelExpenseReimbursement | CapitalEquipmentOrder
STEP     sign
CRITERIA IF amount < authorization level of applicant's manager
         THEN manager of applicant
         ELSE VP responsible for applicant's group
         ENDIF
```

(b)

```
WORKFLOW TravelExpenseReimbursement | CapitalEquipmentOrder
STEP     sign
CRITERIA IF amount < authorization level of applicant's
                     manager depending on workflow type
         THEN manager of applicant
         ELSE VP responsible for applicant's group
         ENDIF
```

Security. Because changing an organizational policy may affect daily business operations, all users should not be able to make changes at will. For example, a user (applicant) should not be able to approve his/her own travel request. Organizational policies are therefore objects that must be properly secured to prevent users from performing unauthorized tasks. The decision to design organizational policies as objects makes it easier to secure the policies, because security mechanisms such as access control lists (ACLs) can be applied directly to objects.[73]

Designers considered and rejected the alternative approach of securing the workflow specification and, consequently, the organizational policies included in the specification. Workflow types do have to be secured to prevent unauthorized changes; however, securing the workflow specification would allow those who are eligible to change the workflow type to also change the associated organizational policies. Such an all-encompassing security design inhibits the separation of duty between workflow designers who care about how a business process is implemented by a workflow and organization designers who care about the organization structure and the user capabilities and responsibilities. Protecting workflows independently of organizational policies allows users to modify a workflow without allowing them to modify organizational policies and thus gain or grant unauthorized eligibility. Similarly, organization schemas and organization structures must be secured independently to prevent users from changing roles or relationships to gain or grant unauthorized authority.

Generality. Although several standard organization structures prevail -- strong hierarchical, matrix-shaped, function-oriented, and networked -- hybrid organization structures exist, which contain a myriad of anomalies and exceptions. Independent of their organization structure, most enterprises have business processes that are potential candidates for a WFMS implementation. A WFMS that claims to be able to implement business processes in all kinds of enterprises must therefore be able to support all possible organization structures. A fixed organization schema is inadequate for such a universal implementation capability. Consequently, PRA supports the modeling of arbitrary organization schemas and allows WFMSs to implement any organization that might exist.

Following this general approach, it is apparent that a fixed set of assignment rules is also inadequate. The PRA design hence provides a language that enables users to define task assignment rules (organizational policies) as required by the workflows of an enterprise.

Dynamics. Organizations change for many reasons, e.g., employee numbers fluctuate, restructuring takes place, groups join or split because of new product strategies, etc. Business operations and therefore workflows, however, must continue uninterrupted. To do so, the organization structure and the organizational policies of a WFMS must change to reflect the changes in the real organization. The decision to separate workflows from organization structures and organizational policies enables users to change versions independently. For example, an organizational policy can change while a workflow that uses it is running. If the change takes place before the WFMS assigns the step to a user, the WFMS will use the new version of the organizational policy instead of the old version. Policy changes result in neither the shutting down of the WFMS nor the stopping and restarting (from the beginning) of the workflow. This independence allows WFMSs to deal with the dynamics of an organization and make correct task assignments while changes are taking place.

Distribution. Not only are enterprises becoming more distributed, but they are also increasing their worldwide operation. Nations have different local laws and policies because they decide autonomously on these issues. A local subsidiary has to adhere to local law, even though it belongs to a company that operates worldwide. For example, U.S. companies have a position called vice president. A U.S. company may have the rule that contracts with external suppliers of manufacturing parts must be signed by the vice president of manufacturing. If the U.S. company has a German subsidiary, by German law, this subsidiary is a company in itself and must have a person called "Geschaftsfuhrer" who is responsible for the operations of the company. If the subsidiary wants to enter into a contract with a supplier, German law requires the Geschaftsfuhrer to sign the contract even though the U.S. corporate organizational policy requires the vice president of manufacturing to sign. Although the same type of workflow is running in both countries, e.g., the contract with external supplier workflow, the organizational policies for the approval step differ. The U.S. version of the organizational policy specifies the vice president of manufacturing is the only eligible user, and the German version specifies that the Geschaftsfuhrer the only eligible user.

Domains were introduced to deal with the issue of autonomous policies. A domain is an abstract entity of management. Organizational policies as well as workflows are related to domains. The previous example might involve two domains: "USA" and "GERMANY." (The domains could be further subdivided.)

The principles just discussed guided the PRA design. As mentioned in the previous section, PRA defines the concepts of organization schema, organizational policy, and a formal language to model policies. In addition, PRA defines interfaces for an execution engine and their use by a WFMS. A detailed discussion of the PRA

components follows.

Organization Schema and Organization Structure

The PRA organization schema is a set of objects and relationships that can be freely defined, thus enabling users to model arbitrary organizations. Each member of the set can be instantiated to populate an organization schema, that is, to produce an organization structure. PRA allows users to define constraints on the organization structure to avoid erroneous structures. For example, if an enterprise has the policy that an employee must not report to more than two people, PRA enables the user to define a constraint that specifies that one person can be related to only two others through a "reports to" relationship. If a modeler adds a third reporting line, the system detects the violated constraint.

Organizational Policy

An organizational policy specifies a set of eligible users for a given workflow, which can be either elementary (a step) or composite. A set of users is not stable and therefore fixed but specified through an expression called an organizational expression. An organizational expression specifies the selection of users with particular properties from an organization structure. For example, an expression might enumerate users, select all users able to play a particular role, or select a user related to some other in a specific way. Additionally, organizational expressions can refer to the history of a workflow or to its internal data, such as local variables, and thus be dependent on the workflow state. Consequently, the set of users for the same step in two different instances of the same workflow might be different. Consider, for example, the travel expense reimbursement workflow, with the user selection for the sign step dependent on the authorization level. In two instances of the workflow, the amounts to be reimbursed might differ such that different people, e.g., the manager and the vice president, must execute the two sign steps.

To provide a general mechanism for determining a set of eligible users for a workflow, PRA organizational policies accommodate operations in addition to executing a step or taking responsibility for a composite workflow. Delegating a workflow and undoing a workflow are two examples. To delegate a workflow, an organizational policy has to ensure that both the person who delegates the workflow and the person to whom the workflow is assigned are eligible users. The operation of undoing a workflow (i.e., to undo the results achieved thus far) and starting again can result in wasted effort and unrecoverable work. Therefore, a WFMS must carefully choose eligible users for this operation.

To deal with various workflow operations, a PRA organizational

policy relates a workflow type and one of its operations in a given domain to an organizational expression. An organizational policy is defined as the tuple <workflow type, operation, domain, organizational expression>. For example, the organizational policy for the fill step in the travel expense reimbursement example is <TravelExpenseReimbursement.Fill, execute, USA, `every user who plays the role of employee'>. Since an applicant should be able to undo the step and start again, the WFMS must also specify the organizational policy <TravelExpenseReimbursement.Fill, undo, USA, `the user who started fill'>. (The next section describes PRA's formal language for specifying organizational policies.)

When a WFMS determines that a workflow in a particular domain is to be executed, it calls the policy resolution engine, which looks for the appropriate organizational policy and evaluates its organizational expression. The engine returns the results of the evaluation, i.e., the set of eligible users, to the WFMS, which subsequently assigns the workflow to those users. One organizational policy can be reused for several workflow types, domains, etc., by entering a set in the appropriate element of the tuple. For example, if the organizational policy for the fill step of the travel expense reimbursement workflow is the same in the U.S. as it is in Europe, the policy could be modeled as <TravelExpenseReimbursement.Fill, execute, {USA, EUROPE}, `every user who plays the role of employee'>.

Policy Definition Language

From the organizational viewpoint, the following elements are necessary to run a workflow: an organization schema together with its instantiation, the organizational policies for this workflow, and the relevant organizational expressions. To describe these elements in a formal way, PRA defines a language called the Policy Definition Language (PDL), which consists of several parts. The first part enables the definition of an organization schema and its population. The second part is concerned with organizational expressions. Finally, the third part supports the definition of organizational policies.

The following figures illustrate the PDL for a sample organization schema and organization structure, some organizational expressions, and some organizational policies for the travel expense reimbursement workflow. Figure 5 shows the PDL for the organization schema displayed in Figure 2a. The PDL for the instantiation displayed in Figure 2b appears in Figure 6.

Figure 5 Policy Definition Language for the Sample Organization Schema Shown in Figure 2a

```
ORGANIZATION_TYPE Role
        ATTRIBUTES name: String
                   authorization_level: set(task, amount);
        KEYS name;

ORGANIZATION_TYPE Group
        ATTRIBUTES name: String
        KEYS name;

ORGANIZATION_TYPE User
        ATTRIBUTES name: String
                   office_tel_#: String
                   e_mail: String
                   absence: {vacation, ill, business, available}
        KEYS name;

RELATIONSHIP_TYPE Reports_to
        FROM User
        TO   User
        ATTRIBUTES kind: {line, functional, none}

RELATIONSHIP_TYPE Plays
        FROM User
        TO   Role
        ATTRIBUTES duration_from: date
                   duration_to: date

RELATIONSHIP_TYPE Responsible_for
        FROM User
        TO   Group

RELATIONSHIP_TYPE Belongs_to
        FROM User
        TO   Group
```

Note that, for simplicity, we assume user names to be unique. In reality, this is not the case and the modeling must deal with nonunique names.

Figure 6 Policy Definition Language for the Sample Organization Structure (Instantiation) Shown in Figure 2b

```
Role "Employee",  {}
     "Manager",   {(TravelExpenseReimbursement.Sign, 1000),
                   (CapitalEquipmentOrder.Sign, 5000)}
     "FinancialClerk", {}
     "Secretary", {}
     "Engineer",  {}
     "VP",        {(TravelExpenseReimbursement.Sign, *),
                   (CapitalEquipmentOrder.Sign, *)}

Group "Sales"
      "Manufacturing"
      "Engineering"
      "Administration"

User "Al",     "[1] 125-5589", "al@center.com",      available
     "Nina",   "[1] 125-5590", "nina@center.com",    available
     "Ken",    "[1] 125-5601", "ken@center.com",     available
     "Susan",  "[1] 125-5609", "susan@center.com",   business
     "Matt",   "[1] 125-4499", "matt@center.com",    available
     "Charles","[1] 125-4580", "charles@center.com", available
     "Mike",   "[1] 125-0101", "mike@center.com",    available

Reports_to "Al",      "Nina", line
           "Ken",     "Nina", line
           "Nina",    "Mike", line
           "Susan",   "Matt", line
           "Charles", "Matt", line
           "Matt",    "Mike", line
           "Mike",    "",     none

Plays "Al",      "Employee",  01-02-88, 0-0-0 (* open ended *)
      "Al",      "FinancialClerk", 01-02-88, 0-0-0
      "Nina",    "Employee",  01-02-90, 0-0-0
      "Nina",    "Manager",   01-02-90, 0-0-0
      "Ken",     "Employee",  01-02-91, 0-0-0
      "Ken",     "Secretary", 01-02-91, 0-0-0
      "Susan",   "Employee",  01-02-92, 0-0-0
      "Susan",   "Secretary", 01-02-92, 0-0-0
      "Matt",    "Employee",  01-02-88, 0-0-0
      "Matt",    "Manager",   01-02-88, 0-0-0
      "Charles", "Employee",  01-02-88, 0-0-0
      "Charles", "Engineer",  01-02-88, 0-0-0
      "Mike",    "Employee",  01-02-90, 0-0-0
      "Mike",    "VP",        01-02-93, 12-31-97

Responsible_for "Al",   "Sales"
                "Al",   "Manufacturing"
```

```
                "Al",   "Engineering"
                "Mike", "Sales"
                "Mike", "Manufacturing"
                "Mike", "Engineering"

Belongs_to "Al",      "Administration"
           "Nina",    "Engineering"
           "Ken",     "Administration"
           "Susan",   "Administration"
           "Matt",    "Engineering"
           "Charles", "Engineering"
           "Mike",    ""
```

The organization schema definition part of the PDL looks like a data definition language (DDL) in a relational database. Two differences exist, though: (1) PDL distinguishes organizational object types from organizational relationship types, and (2) PDL allows complex data types (e.g., sets as attributes). If a policy resolution engine is built on top of a relational database, a compiler or a translator within the engine translates the organization schema definition part of PDL into a set of DDL statements.

Figure 7 lists the organizational expressions required to formulate the organizational policies for the travel expense reimbursement workflow. Note that the organizational expression for employees selects all users who play the role of employee. The RETURNS statement indicates the search for users. The definition of the plays relationship type in Figure 5 indicates that the employee is of the type role. This information is sufficient to formulate a query to the underlying database system in an implementation of a policy resolution engine.

Figure 7 Organizational Expressions for the Travel Expense Reimbursement Example

```
ORGANIZATIONAL_EXPRESSION employees()
    RETURNS User: user
        user plays employee

ORGANIZATIONAL_EXPRESSION secretaries()
    RETURNS User: user
        user plays secretary

ORGANIZATIONAL_EXPRESSION manager_of(User: a_user)
    RETURNS User: user
        a_user reports_to user

ORGANIZATIONAL_EXPRESSION subordinates_of(User: a_user)
    RETURNS User: user
        user reports_to a_user

ORGANIZATIONAL_EXPRESSION group_of(User: a_user)
    RETURNS Group: group
        a_user belongs_to group

ORGANIZATIONAL_EXPRESSION VP_responsible_for_group_of(User: a_user)
    RETURNS User: user
        user plays VP
        INTERSECTION
        user responsible_for group_of(a_user)

ORGANIZATIONAL_EXPRESSION executing_agent(Workflow: a_workflow)
    RETURNS User
        (* provided by the historical services of WFMS *)
```

The PDL for the organizational policies for the travel expense reimbursement example appears in Figure 8. The WFMS applies the first organizational policy when assigning the fill step in a travel expense reimbursement workflow. The policy is valid in three domains, USA, EUROPE, and ASIA, for the execute operation, which has no parameters. The policy engine returns a set of all users who are able to play the role of employee. The second policy listed in Figure 8 returns a set of all users who play the role of secretary and who report to the same user as the applicant.

Figure 8 Organizational Policies for the Travel Expense Reimbursement Example

```
ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Fill
        OPERATION Execute()
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION employees()

ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Check
        OPERATION Execute()
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION
                  secretaries()
                    INTERSECTION
                  subordinates_of(
                    manager_of(
                      executing_agent(
                        TravelExpenseReimbursement.Fill)))

ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Sign
        OPERATION Execute()
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION
                  manager_of(
                    executing_agent(
                      TravelExpenseReimbursement.Fill))

ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Reimburse
        OPERATION Execute()
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION
                  financial_clerks()
                    INTERSECTION
                  User: user responsible_for
                  group_of(
                    executing_agent(
                      TravelExpenseReimbursement.Fill))
```

Independent from the travel expense reimbursement example are the sample separation of duty and delegation policies shown in Figures 9 and 10. The organizational policy that specifies separation of duty ensures that the user who signs the expense form is different from the user who fills out the form. The policy that models the delegation operation contains a parameter that specifies to which person the sign step is to be delegated. Only the manager of the applicant can call this operation and then only if the parameter specifies either the next higher manager or the responsible vice president. The step can be delegated only to one of these two users.

Figure 9 Organizational Policy for the Separation of Duty

```
ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Sign
        OPERATION Execute()
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION
                  manager_of(
                    executing_agent(
                      TravelExpenseReimbursement.Fill))
                  DIFFERENCE
                  executing_agent(
                      TravelExpenseReimbursement.Fill)
```

Figure 10 Organizational Policy for the Delegate Operation

```
ORGANIZATIONAL_POLICY
        WORKFLOW  TravelExpenseReimbursement.Sign
        OPERATION Delegate (User: a_user)
        DOMAIN    USA, EUROPE, ASIA
        ORGANIZATIONAL_EXPRESSION
                  IF a_user IN
                     (manager_of(
                        manager_of(
                          executing_agent(
                            TravelExpenseReimbursement.Fill)))
                      OR
                      VP_responsible_for_group_of(
                        executing_agent(
                          TravelExpenseReimbursement.Fill)))
                  THEN
                     manager_of(
                       executing_agent(
                         TravelExpenseReimbursement.Fill))
```

Since the PDL is well defined, it can be used not only by designers to model organizations and policies but also by developers of graphics-oriented tools. Such tools could present graphical symbols to users to be manipulated. When a user decides to commit the changes, the tool generates a PDL script, which is fed into the policy resolution engine.

Approaches like the ones mentioned earlier in the paper provide a fixed set of types for modeling an organization or a fixed set of functions, such as "role player" or "supervisor," from which to select users for a workflow. None of these approaches provides a language like PDL that can freely define the organizational aspect as the application semantics requires.

Policy Resolution Engine

The policy resolution engine is a mechanism that evaluates organizational policies for a WFMS. Serving as a base service, the policy resolution engine manages organizational policies and organizational expressions, as well as the organization schema and its population. The engine also provides interfaces for the definition, modification, and evaluation of these objects. The interfaces are distinguished by the kind of service they provide. There are basically two kinds of interfaces: evaluation interfaces and management interfaces.

Evaluation Interfaces. Policy resolution engine clients use evaluation interfaces to evaluate organizational policies or organizational expressions when necessary. The engine provides four evaluation interfaces: two for organizational policies ("resolve" and "conform to") and two for organizational expressions (also "resolve" and "conform to"). The resolve operation for organizational policies expects a workflow reference and one of its operations as input values. This operation selects an appropriate organizational policy, evaluates it, and returns a set of users eligible to execute the given task of the workflow. The conform to operation for organizational policies expects a workflow reference, one of its operations, and a user as input values. This operation resolves the appropriate organizational policy for the workflow and checks whether the user is contained in the set of results for that organizational policy (i.e., if the user conforms to the policy). If the user is contained in the set of results, the conform to operation returns the value "true"; otherwise it returns the value "false." Policy resolution engine clients use this operation to validate a request by a user to execute a certain task of a workflow.

The resolve and conform to operations for organizational expressions work analogously. Instead of a workflow reference, the operations expect the name of an organizational expression as

input. The operations evaluate the named organizational expression and return the set of results, which is used if the resolve operation is called. The conform to operation returns true and false values as described in the previous paragraph.

Management Interfaces. Management interfaces are used to define, modify, or delete organizational policies, organizational expressions, or organization schemas and their populations. These interfaces look like the following operations that are provided for organizational policies: create, delete, modify, list, get. The create operation creates an organizational policy; the delete operation deletes a policy; the modify operation allows users to change an organizational policy to adjust to new requirements; the list operation returns the identifiers of all policies; and the get operation returns the complete description of a policy.

Designers do not call these management interfaces directly, since they communicate their changes through user-friendly interfaces or tools. These tools are either graphics oriented or language oriented. In a graphics-oriented tool, a designer manipulates icons and graphical symbols, which in turn results in calls to the appropriate management interfaces. Alternatively, a graphics tool can generate a PDL script according to the manipulations of a user and submit this script to the policy resolution engine. In this case, the engine interprets the submitted script and changes its internal state accordingly. Language-oriented tools enable a designer to directly express changes using PDL. These tools take specifications and translate them into management interface calls. Of course, they can also submit the language specifications directly as PDL scripts to the policy resolution engine, as described above.

Legacy Databases. Many large enterprises have developed databases that contain some or all of the organizational data the policy resolution engine needs to evaluate organizational policies. These databases, called legacy databases, might be self-implemented or based on standards efforts like those related to providing directory services on networks, i.e., X.500.[74] In general, organizations must deal with one of the following scenarios:

- No legacy database exists. No existing database has to be considered, and the policy resolution engine can use its own database to build up organizational knowledge.
- Legacy databases contain all relevant data. To use the policy resolution engine, the database must provide a sufficiently expressive query interface, on top of which queries issued from the engine can be evaluated. The only additional information that has to be stored is organizational policies and organizational expressions. The organization has to choose whether to extend the

legacy databases or to use the database within the policy resolution engine.

- A legacy database contains some relevant data. In addition to organizational policies and organizational expressions, organizational objects and relationships must be stored in either the legacy database or the database of the policy resolution engine.

If the relevant data is stored in several databases, the querying interface must be built in such a way that the policy resolution engine can issue the necessary queries, which might span several databases. Furthermore, semantics issues have to be dealt with in heterogeneous environments.[75,76]

Architectural Considerations -- Clients of a Policy Resolution Engine. From an architectural point of view, there are two possible ways to design a policy resolution engine:

1. Incorporate the policy resolution engine into a WFMS. The engine would be a module, whose operations are hidden by the exported interfaces of the WFMS. All calls to the engine operations would be made through the interface of the WFMS.

2. Make the policy resolution engine an independent component. The engine would be a server with a WFMS system as one of its clients. All clients of the engine, including the WFMS, would be able to directly access the exported operations of the engine.

PRA recommends the implementation of a policy resolution engine as an independent base service, which can be used by clients other than a WFMS. For example, an electronic mail system can be a client of the policy resolution engine. Since electronic mail is sent to users, rather than enumerate the electronic mail addresses of the recipients by hand, organizational expressions can provide the addresses. For example, a manager could send an electronic mail message to "all my subordinates" or an engineer could send an electronic mail message to "all my colleagues who are engineers." The sample operational expression shown in Figure 11 returns all electronic mail addresses of all subordinates of a given user.

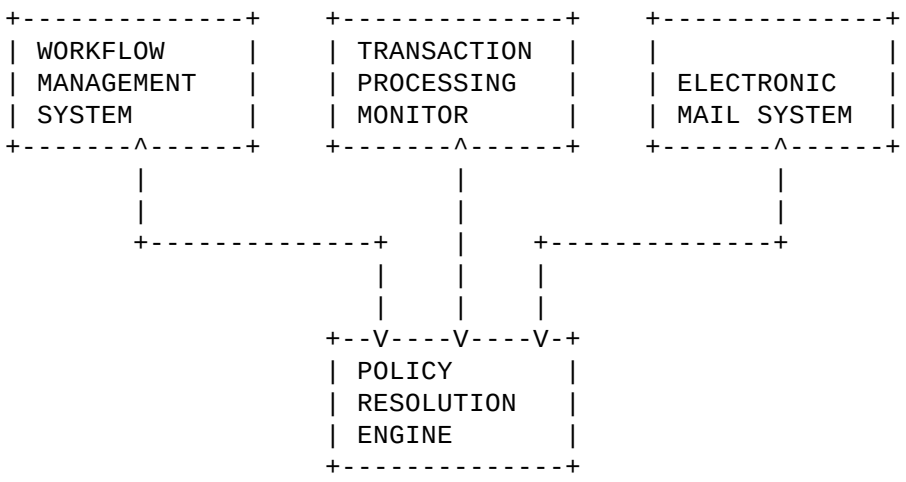
Figure 11 Organizational Expression for Electronic Mail

```
ORGANIZATIONAL_EXPRESSION subordinates(User: a_user)
        RETURNS String: user.e_mail
                user reports_to a_user
```

Another possible client is a transaction processing monitor, which incorporates workflow management.[77] Dayal et al. reference a service called role resolution, which is an earlier development of policy resolution.[78]

Figure 12 shows a schematic representation of a policy resolution engine with three clients -- a WFMS, a transaction processing monitor, and an electronic mail system.

Figure 12 Client-server Structure of a Policy Resolution Engine



SUMMARY

The sample workflow discussed in this paper, that is, the travel expense reimbursement workflow, illustrates that roles are sufficient as task assignment rules for only the simplest scenarios. Since workflow management systems are deployed in situations where complex workflows are modeled and executed, a more general and powerful model called the Policy Resolution Architecture (PRA) was developed. PRA provides the concept of an organizational policy. An organizational policy is more general than a role in that it relates a workflow type to an organizational expression that determines the set of eligible users for the workflow. Because they state all criteria a user has to fulfill and do not limit the selection based on their properties or interrelationships, organizational policies specify all eligible users. Since an organizational expression is related to a workflow type by an organizational policy, task assignment through organizational policies is a very general approach. Organizational policies are evaluated based on organization schema and their populations (organization structures). Since PRA provides a way to model arbitrary complex organization schemas, arbitrary organizations can be modeled and subsequently populated. This generality, in conjunction with organizational policies, provides a powerful and flexible approach to task assignment in workflow management.

ACKNOWLEDGMENTS

I want to thank the anonymous referees whose remarks helped me a great deal in revising this paper. Susan Thomas assisted me by improving my English and thus making the paper more readable. Kathy Stetson was always very helpful in coordinating the writing and revision processes.

REFERENCES

1. T. May, "Know Your Work-Flow Tools," BYTE (July 1994).

2. T. Kreifelts and P. Seuffert, "Addressing in an Office Procedure System," Message Handling Systems, State of the Art and Future Directions: Proceedings IFIP WG 6.5 Working Conference on Message Handling Systems, R. Speth, ed. (Amsterdam: North-Holland, 1987).

3. S. Chang and W. Chan, "Transformation and Verification of Office Procedures," IEEE Transactions on Software Engineering, vol. SE-11, no. 8 (August 1985).

4. W. Croft and L. Lefkowitz, "Task Support in an Office System," ACM Transactions on Office Information Systems, vol. 2, no. 3 (July 1984).

5. C. Ellis and G. Nutt, "Office Information Systems and Computer Science," Computing Surveys, vol. 12, no. 1 (March 1980).

6. C. Ellis and M. Bernal, "Officetalk-D: An Experimental Office Information System," First SIGOA Conference on Office Information Systems (1982).

7. C. Ellis, "Formal and Informal Models of Office Activity," Information Processing 83, R. Mason, ed. (Amsterdam: North-Holland, 1983).

8. B. Karbe and N. Ramsperger, "Concepts and Implementation of Migrating Office Processes," Verteilte Kunstliche Intelligenz und Kooperatives Arbeiten: 4. Internationaler GI-Kongress Wissensbasierte Systeme, Informatik Fachberichte 291, W. Brauer and D. Hernandez, eds. (Munich: Springer-Verlag, October 1991).

9. T. Kreifelts, "Coordination of Distributed Work: From Office Procedures to Customizable Activities," Verteilte Kunstliche Intelligenz und Kooperatives Arbeiten: 4. Internationaler GI-Kongress Wissensbasierte Systeme, Informatik Fachberichte 291, W. Brauer and D. Hernandez, eds. (Munich: Springer-Verlag, October 1991).

10. C. Cook, "Streamlining Office Procedures -- An Analysis Using the Information Control Net Model," AFIPS Conference Proceedings of the 1980 National Computer Conference, Anaheim, California (May 1980).

11. I. Ladd and D. Tsichritis, "An Office Form Flow Model," AFIPS Conference Proceedings of the 1980 National Computer Conference, Anaheim, California (May 1980).

12. L. Baumann and R. Coop, "Automated Workflow Control: A Key to Office Productivity," AFIPS Conference Proceedings of the 1980 National Computer Conference, Anaheim, California (May 1980).

13. M. Zisman, "Representation, Specification and Automation of Office Procedures," Ph.D. dissertation (Philadelphia: University of Pennsylvania, Wharton School, 1977).

14. B. Curtis, M. Kellner, and J. Over, "Process Modeling," Communications of the ACM, vol. 35, no. 9 (September 1992).

15. W. Deiters and V. Gruhn, "The Funsoft Net Approach to Software Process Management," International Journal of Software Engineering and Knowledge Engineering, vol. 4, no. 2 (1994).

16. W. Deiters, V. Gruhn, and H. Weber, "Software Process Evolution in MELMAC," The Impact of CASE Technology on Software Processes Series on Software Engineering and Knowledge Engineering, vol. 3, D. Cooke, ed. (Singapore: World Scientific Publishing, 1994).

17. D. Harel et al., "STATEMATE: A Working Environment for the Development of Complex Reactive Systems," Proceedings of the Tenth International Conference on Software Engineering (1988).

18. W. Humphrey and M. Kellner, "Software Process Modeling: Principles of Entity Process Models," Proceedings of the Eleventh International Conference on Software Engineering (May 1989).

19. M. Jaccheri and R. Conradi, "Techniques for Process Model Evolution in EPOS," IEEE Transactions on Software Engineering (December 1993).

20. T. Katayama, "A Hierarchical and Functional Software Process Description and Its Enaction," Proceedings of the Eleventh International Conference on Software Engineering (May 1989).

21. P. Mi and W. Scacchi, Operational Semantics of Process Enactment and Its Prototype Implementations (Los Angeles: University of Southern California, Computer Science Department, April 1991).

22. P. Mi and W. Scacchi, Modeling Articulation Work in Software Engineering Processes (Los Angeles: University of Southern California, Computer Science Department, April 1991).

23. P. Mi and W. Scacchi, "A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes," IEEE Transactions on Knowledge and Data Engineering, vol. 2,

no. 3 (September 1990).

24. L. Osterweil, "Software Processes Are Software Too," Proceedings of the Ninth International Conference on Software Engineering (March/April 1987).

25. L. Williams, "Software Process Modeling: A Behavioral Approach," Proceedings of the Tenth International Conference on Software Engineering (1988).

26. W. Royce, "Managing the Development of Large Software Systems," Proceedings of the Ninth International Conference on Software Engineering (March/April 1987).

27. B. Boehm, "A Spiral Model of Software Development and Enhancement," ACM Software Engineering Notes, vol. 11, no. 4 (August 1986).

28. C. Hegarty and L. Rowe, "Control Loops and Dynamic Run Modifications Using the Berkeley Process-Flow Language," Proceedings of the Third International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Lyons, France (1992).

29. S. Jablonski, "Data Flow Management in Distributed CIM Systems," Proceedings of the Third International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Lyons, France (1992).

30. Proceedings of the Third International Conference on Data and Knowledge Systems for Manufacturing and Engineering, Lyons, France (1992).

31. H. Yoshikawa and J. Goossenaerts, eds., Information Infrastructure Systems for Manufacturing (Amsterdam: North-Holland, 1993).

32. T. Harder and A. Reuter, "Principles of Transaction-oriented Database Recovery," ACM Computing Surveys, vol. 15, no. 4 (December 1983).

33. P. Attie, M. Singh, A. Shet, and M. Rusinkiewicz, "Specifying and Enforcing Intertask Dependencies," Proceedings of the Nineteenth International Conference on Very Large Databases (VLDB), Dublin, Ireland (1993).

34. Y. Breitbart, A. Deacon, H. Schek, and G. Weikum, "Merging Application-centric and Data-centric Approaches to Support Transaction-oriented Multi-system Workflows," SIGMOD Record, vol. 22, no. 3 (September 1993).

35. U. Dayal, M. Hsu, and R. Ladin, "A Transactional Model for Long-Running Activities," Proceedings of the Seventeenth International Conference on Very Large Databases (VLDB),

Barcelona, Spain (September 1991).

36. H. Garcia-Molina and K. Salem, "Sagas," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (1987).

37. Bulletin of the Technical Committee on Data Engineering, vol. 16, no. 2 (June 1993).

38. S. Jablonski, "Transaction Support for Activity Management," Proceedings of the Workshop on High Performance Transaction Processing Systems (HPTS), Asilomar, California (1993).

39. H. Wachter and A. Reuter, "The ConTract Model," in Transaction Models for Advanced Database Applications, A. Elmagarmid, ed. (San Mateo, California: Morgan Kaufmann, 1992).

40. T. Malone and K. Crowston, "The Interdisciplinary Study of Coordination," ACM Computing Surveys, vol. 26, no. 1 (March 1994).

41. T. Malone, K. Crowston, J. Lee, and B. Pentland, "Tools for Inventing Organizations: Toward a Handbook of Organizational Processes," CCS WP #141, Sloan School WP #3562-93 (Cambridge: Massachusetts Institute of Technology, Center for Coordination Science, May 1993).

42. R. Burkhart, "Process-based Definition of Enterprise Models," Proceedings of the First International Conference on Enterprise Integration Modeling Technology (ICEIMT), Hilton Head, South Carolina (June 1992).

43. C. Bussler, "Enterprise Process Modeling and Enactment in GERAM," Proceedings of the International Conference on Automation, Robotics and Computer Vision (ICARCV '94), Singapore (November 1994).

44. M. Fox, "The TOVE Project: Towards a Common-Sense Model of the Enterprise," Proceedings of the First International Conference on Enterprise Integration Modeling Technology (ICEIMT), Hilton Head, South Carolina (June 1992).

45. Proceedings of the First International Conference on Enterprise Integration Modeling Technology (ICEIMT), Hilton Head, South Carolina (June 1992).

46. R. Katz, "Business/enterprise Modeling," IBM Systems Journal, vol. 29, no. 4 (1990).

47. J. Sowa and J. Zachman, "Extending and Formalizing the Framework for Information Systems Architecture," IBM Systems Journal, vol. 31, no. 3 (1992).

48. F. Vernadat, "Business Process and Enterprise Activity Modelling: CIMOSA Contribution to a General Enterprise Reference Architecture and Methodology (GERAM)," Proceedings of the International Conference on Automation, Robotics and Computer Vision (ICARCV '94), Singapore (November 1994).

49. T. Williams, "Architectures for Integrating Manufacturing Activities and Enterprises," Information Infrastructure Systems for Manufacturing, H. Yoshikawa and J. Goossenaerts, eds. (Amsterdam: North-Holland, 1993).

50. F. Flores and T. Winograd, Understanding Computers and Cognition (Reading, MA: Addison-Wesley, 1987).

51. R. Medina-Mores, R. Winograd, T. Flores, and F. Flores, "The Action Workflow Approach to Workflow Management Technology," Proceedings of the ACM 1992 Conference on Computer Supported Cooperative Work (CSCW), Toronto, Ontario, Canada (November 1992).

52. T. Danielsen and U. Pankoke-Babatz, "The Amigo Activity Model," in Research into Networks and Distributed Applications, R. Speth, ed. (Munich: North-Holland, Elsevier Science Publishers B.V., 1988).

53. R. Fehling, K. Joerger, and D. Sagalowicz, Knowledge Systems for Process Management (Palo Alto, CA: Teknowledge Inc., 1986).

54. J. Guyot, "A Process Model for Data Bases," SIGMOD Record, vol. 17, no. 4 (December 1988).

55. C. Bussler and S. Jablonski, "An Approach to Integrate Workflow Modeling and Organization Modeling in an Enterprise," Proceedings of the Third IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE), Morgantown, West Virginia (April 1994).

56. S. Jablonski, "MOBILE: A Modular Workflow Model and Architecture," Proceedings of the Fourth Working Conference on Dynamic Modelling and Information Systems, Noordwijkerhout, Netherlands (September 1994).

57. M. Hsu, A. Ghoneimy, and C. Kleissner, "An Execution Model for an Activity Management System," Proceedings of the Workshop on High Performance Transaction Systems (1991).

58. M. Hsu and M. Howard, "Work-Flow and Legacy Systems," BYTE (July 1994).

59. F. Leymann and W. Altenhuber, "Managing Business Processes as an Information Resource," IBM Systems Journal, vol. 33, no. 2 (1994).

60. Workflow Management Software: The Business Opportunity (Ovum Reports, December 1991).

61. T. White and L. Fischer, "New Tools for New Times: The Workflow Paradigm (Alameda: Future Strategies Inc., Book Division, 1994).

62. J. Bair, "Contrasting Workflow Models: Getting to the Roots of Three Vendors," Proceedings of the Groupware '93 Conference, San Jose, California (1993).

63. S. Sarin, K. Abbot, and D. McCarthy, "A Process Model and System for Supporting Collaborative Work," Proceedings of the ACM SIGOIS Conference on Organizational Computing Systems (November 1991).

64. M. Shan, "Pegasus Architecture and Design Principles," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C. (May 1993).

65. M. Ansari, L. Ness, M. Rusinkiewicz, and A. Sheth, "Using Flexible Transactions to Support Multi-System Telecommunication Applications," Proceedings of the Eighteenth International Conference on Very Large Databases (VLDB), Vancouver, British Columbia, Canada (1992).

66. D. Evans, "Putting Elves to Work: Workflow Technology in a Law Firm," Proceedings of the Groupware '93 Conference, San Jose, California (1993).

67. D. Sng, "A National Information Infrastructure for the 21st Century Collaborative Enterprise," Proceedings of the International Conference on Automation, Robotics and Computer Vision (ICARCV '94), Singapore (November 1994).

68. R. Marshak, "Characteristics of a Workflow System -- Mind Your P's and R's," Proceedings of the Groupware '93 Conference, San Jose, California (1993).

69. C. Bussler and S. Jablonski, "Implementing Agent Coordination for Workflow Management Systems Using Active Database Systems," Proceedings of the Fourth International Workshop on Research Issues in Data Engineering: Active Database Systems (RIDE-ADS '94), Houston, Texas (February 1994).

70. C. Ellis, S. Gibbs, and G. Rein, "Groupware -- Some Issues and Experiences," Communications of the ACM, vol. 34, no. 1 (January 1991).

71. L. Lawrence, "The Role of Roles," Computers and Security, vol. 12, no. 1 (1993).

72. L. Aiello, D. Nardi, and M. Panti, "Modeling the Office Structure: A First Step towards the Office Expert System," Second ACM SIGOA Conference on Office Information Systems (ACM SIGOA), vol. 5, nos. 1 and 2 (1984).

73. D. Denning, Cryptography and Data Security (Reading, MA: Addison-Wesley, 1983).

74. Blue Book, Volume VIII, Fascicle VIII.8, Data Communication Networks: Directory, Recommendations X.500-X.521 (Study Group VII), Comite Consultatif Internationale de Telegraphique et Telephonique.

75. S. Ceri and J. Widom, "Managing Semantic Heterogeneity with Production Rules and Persistent Queues," Proceedings of the Nineteenth Conference on Very Large Databases (VLDB), Dublin, Ireland (1993).

76. W. Kent, "Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language," Proceedings of the Seventeenth International Conference on Very Large Databases (VLDB), Barcelona, Spain (September 1991).

77. U. Dayal et al., "Third Generation TP Monitors: A Database Challenge," Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C. (May 1993).

78. C. Bussler, "Capability Based Modeling," Proceedings of the First International Conference on Enterprise Integration Modeling Technology (ICEIMT), Hilton Head, South Carolina (June 1992).

BIOGRAPHY

Christoph J. Bussler Christoph Bussler is a faculty member at the Technical University of Darmstadt, Germany, where he is pursuing a Ph.D. degree. His research work is in workflow and organization modeling, with a focus on organizational embedding of workflow management, and in architectures for enterprise-wide deployment of workflow management systems. While at Digital from 1991 to 1994, Christoph developed the Policy Resolution Architecture and its prototype implementation. He holds an M.C.S. (1990) from the Technical University of Munich and has published numerous papers on workflow management and enterprise modeling.

TRADEMARKS

Digital and ObjectFlow are trademarks of Digital Equipment Corporation.