

Authorization and Access Control in Adaptive Workflows

Dulce Domingos¹, António Rito-Silva², and Pedro Veiga¹

¹ Informatics Department,
University of Lisbon, Faculty of Sciences
{dulce, pmv}@di.fc.ul.pt

² INESC-ID Software Engineering Group
Technical University of Lisbon
Rito.Silva@inesc-id.pt

Abstract. In recent years we have witnessed the development of adaptive workflow management systems. These systems offer an extended set of features to support both ad-hoc and evolutionary changes, while ensuring correctness of process definition and their running instances. Ad-hoc and evolutionary changes impose new access control requirements, which have been neglected by adaptive workflow research and are not met by existing models for traditional workflow management systems (WfMSs). In this paper, we extend the role-based access control model for adaptive workflows. This extension is done by defining authorizations for adaptive WfMSs and adaptive authorizations.

Keywords: Access control, adaptive workflow, adaptive authorizations

1 Introduction

Workflow management systems (WfMSs) are being used to re-engineer, streamline, automate, and track organizational processes [2]. There is a growing acceptance of workflow technology in numerous application domains such as telecommunications, software engineering, manufacturing, production, finance and banking, healthcare, shipping, and office automation [2]. However traditional WfMSs suffer from the lack of flexibility [1], which is a limitation for supporting more demanding applications, more dynamic environments and better human involvement in organizational activities [2].

Indeed, one of the most important requirements emerging from new application areas of WfMSs is the ability to handle adaptable and dynamic changes. As a consequence, in the last years we have witnessed the development of new types of WfMSs supporting adaptive workflows [2, 3]. In this way, it is possible to change process definitions as well as process instances while they are being executed.

However, this flexibility raises new access control requirements, which are not met by traditional WfMSs access control models. In traditional WfMSs all instances are assumed to be alike, with no special handling of specific instances. Moreover, users' participation is always passive; it is assumed that users only

need to know about the activity they are personally involved in. Workflow participants get a work item from their worklist, handle it and check it in when it is finished. Indeed, it is assumed they do not need to know about the workflow as a whole. There are workflow administrators that deal with process definition. Consequently, access control requirements of traditional WfMSs focus mainly on task assignment.

Considering access control requirements, adaptive workflow literature agrees [2, 14, 29] that it must be possible to control at a very fine level of granularity who is allowed to perform which kind of changes and under which conditions. Therefore, the model should support the definition of who have the authority to introduce changes in the workflow and where. It should also distinguish the scopes of the adaptation, which can be done at instance level or at process definition level. Another important aspect stated by these authors points out that the workflow should be flexible enough to be able to adapt to changes in the organizational model, such as changes in persons or resources for a given role and changes in roles for given tasks.

In this paper, we adapt and extend the well-accepted role-based access control model [20] for adaptive WfMSs in two directions. One direction deals with the definition of authorizations considering the objects of adaptive WfMSs, such as: process definition, process instances, activities and activity instances; and operations on them like execute, change, read and grant. Moreover, subjects can be defined as dynamic roles using expressions that could refer to external resource information systems. This mechanism makes the model more adaptive to dynamic changes in organizations, by separating the role model from the organizational model. Implication rules are also defined, which are used to derive implicit authorizations from explicit ones.

The second direction extends the model in order to deal with adaptive authorizations. Our approach to support the evolution of authorizations is based on the following three mechanisms: (1) the definition of a set of operations that allows the change of authorizations, (2) the definition and enforcement of a correctness criterion, which guarantees the correctness of the access control aspect within a process definition, without neglecting side effects of change operations on other workflow aspects and (3) the definition and enforcement of a migration condition, which ensures the correctness of process instances in terms of access control.

The remainder of this paper is structured as follows: section 2 presents a brief overview of adaptive WfMSs. Section 3 surveys some related work in this area. In section 4 we discuss access control requirements of adaptive WfMSs and we illustrate them with an application scenario. Our access control model for adaptive WfMSs is presented in section 5 and section 6 describes the application of this access control model in the WorkSCo WfMSs. Finally, the last section presents some conclusions and provides a brief look for our future work.

2 Adaptive WfMS

This section reviews basic workflow concepts, according to the model and terminology defined by the Workflow Management Coalition [5], introduces adaptive workflows and classifies the two types of workflow adaptations.

A workflow management system is a system that supports the modelling and enactment of workflows. Workflow modelling creates a workflow process definition, which is a computerized representation of a business process and, normally, comprises a number of activities, whose chronological and logical order is given through the control flow. These activities can be atomic (basic work steps) or composite. Composite activities include one or more activities (designated by sub-activities) defining a hierarchy of activities. Workflow modelling requires a workflow meta-model that comprises a set of modelling concepts. A concrete workflow process definition is expressed in a workflow modelling language, which is a formal language offering constructs for the modelling concepts of the meta-model.

Workflow enactment involves the process definition interpretation by the WfMS, which creates and controls process instances, scheduling the various activities and invoking the appropriate human and application resources. Each process instance represents one individual enactment of the process definition, using its own process instance data. The next figure illustrates these concepts and their relationships, using the Unified Modelling Language (UML).

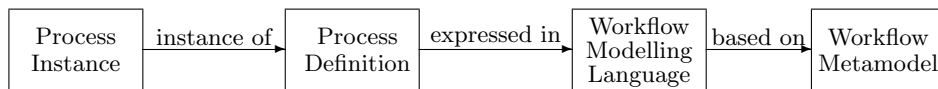


Fig. 1. Workflow modelling concepts

In recent years, we have witnessed the development of new types of WfMSs, which deal with adaptive workflows. Typically, adaptive WfMSs consider two types of changes:

- ad-hoc changes affect only one workflow instance or a selected group of instances. Changes occur on an individual or selective basis. The change is the result of an error, a rare event, or a special demand from the customer.
- evolutionary changes are of a structural nature: from a certain moment in time, the workflow definition changes for all new instances. Existing instances may also be influenced by an evolutionary change. The change can be the result of a new business strategy, reengineering efforts, or a permanent alteration of external conditions (e.g. a change of law).

Accordingly to Casati et al. [11] the problem of evolutionary changes has two facets:

- static evolution facet refers to the issue of modifying the process definition. In order to support the change of process definitions, adaptive WfMSs provide a complete and minimal set of primitives that allows changing the various aspects of the process definition. Typically, these primitives guarantee that the new version of the workflow definition is syntactically correct [11, 15, 14].
- dynamic evolution refers to the problem of managing running instances of a process, whose definition has been modified. The dynamic evolution facet requires mechanisms to support the migration of existing instances to new process definitions, without affecting their correctness. A process instance is correct if it is compliant with its definition, i.e., if it has been executed according to this new version up to the moment of the migration [11]. A simple but inefficient approach to determine whether a process instance can be migrated is to check its history for compatibility with the destination process definition [11, 15]. A more efficient approach, presented by Kradolfer [15], defines a migration condition for each of the provided model change operations and determines whether migration is possible by considering the migration conditions of the operations by which the destination process definition is derived from its source. To deal with process instances that cannot be migrated to the new process definition and have to remain with the old one, some adaptive WfMSs support different versions of process definitions [15, 12].

Note that ad-hoc changes are typically supported by adaptive WfMSs as particular cases of evolutionary changes.

As far as we know, research work in adaptive workflow focuses their approaches on specific workflow aspects (mainly the behavioural aspect) and, despite their consensual opinion about the need of access control, this aspect has not been considered yet.

3 Access Control in WfMSs

In recent years, role-based access control (RBAC) has gained a great acceptance. RBAC models have been developed as an alternative to traditional approaches to handling access control in information systems and their main purpose is to facilitate security administration and review [20]. The central notion of RBAC is that authorizations are associated with roles, and users are assigned to appropriate roles. Roles are created for the various job functions in an organization and users are assigned to roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Authorizations can be granted or revoked to roles as needed.

RBAC models have been widely applied to both commercial and research workflow systems [9, 24]. Several extensions to the basic models have been presented in order to increase expressiveness and to propose appropriate tools and mechanisms to support it. These extensions include team-based authorizations [4, 22, 28], history constraints such as separation of duties [8, 9, 19], temporal and

instance-based authorizations [9], mechanisms for inter-organizational workflows [13] and task-based authorizations [27, 23], as well as workflow data access control [25, 18].

However, to our knowledge, no access control model can be found in the literature that addresses access control requirements of adaptive workflows. In addition, only Casati et al. [10] propose in their work authorization rules that allow the definition of who is authorized to perform given types of changes. However, these authorization rules only apply to process instances and they are not assembled in an access control model.

4 Access Control Requirements in Adaptive Workflow

In this section we discuss some access control requirements of adaptive WfMSs in the context of a loan workflow application.

The top-level activities of our simplified loan workflow are illustrated in Figure 2. The activities *receive loan request* and *evaluate loan* are manual activities, while the others are automatic activities. Within this process definition, the activity *receive loan request* can be done by *clerks* and the activity *evaluate loan* can be done by *account managers* if the requestor is a bank client, or by *bank manager* otherwise.

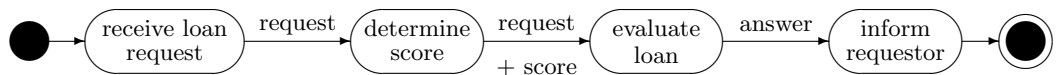


Fig. 2. A simplified loan workflow application

4.1 Process definition authorizations

Adaptive WfMSs provide operations to handle process definitions such as: read, change and instantiate. Therefore, for security reasons, adaptive WfMSs should support the definition of authorizations in order to ensure that only authorized users can perform those operations on process definitions. Authorizations should also be defined at activity definition level.

4.2 Process instance authorizations

Maria's process instance of the loan process definition is illustrated in Figure 3 (grey activities have already been done). Adaptive WfMSs also provide operations to handle process instances. This instance can be executed, read and changed. As we have stated for process definition, adaptive WfMSs should also support the definition of authorizations in order to ensure that only authorized users can perform those operations on process instances and on activity instances.

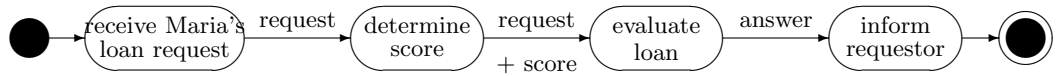


Fig. 3. An instance of the loan workflow application

4.3 Adaptive authorizations

Within a WfMS, process definitions specify all the behaviour of their process instances. Therefore, to comprise the access control aspect, the process definition has to specify, not only its activities, control flow and data flow, but also the authorizations that will be applied to the process instances. Consequently, we have to deal with adaptive authorizations, i.e., we have to support evolutionary and ad-hoc changes of authorizations.

To illustrate the need to support evolutionary changes of authorizations we use the simplified loan workflow illustrated in Figure 2, by considering that, as a result of the definition of more restricted rules for loan process definitions, authorizations defined for activity *evaluate loan* have to be changed: the activity *evaluate loan* can be done by *bank manager* if the score is less than 5 or by *loan manager of central bank* otherwise. Therefore, adaptive WfMSs should also ensure that only authorized users can change authorizations and that correctness of authorizations within the process definition is maintained (static evolution of authorizations facet).

Considering the process instance illustrated in Figure 3, if it needs to be migrated to the new process definition, then the adaptive WfMSs has to ensure the process instance correctness by evaluating if it is compliant with the new definition, taking into account its execution history (dynamic evolution of authorizations facet).

Finally, we point out that, ad-hoc changes are typically supported by adaptive WfMSs as particular cases of evolutionary changes, by creating and defining a specific process definition and migrating the instance to it.

4.4 Conclusions

From the loan workflow application, we conclude that an access control model for adaptive WfMSs needs to protect not only run-time objects (process and activity instances) but also build-time objects (process and activity definitions). The set of access modes that can be exercised on these objects needs to be identified, by taking into account the set of operations provided by adaptive WfMSs.

Finally, within an adaptive WfMSs, authorizations should also be adaptive. To deal with adaptive authorizations, our model should identify who is authorized to change authorizations. Additionally, changing authorizations needs special support in adaptive workflows in order to maintain the correctness of the process definition and its process instances.

5 An Access Control Model for Adaptive WfMSs

In this section we present an access control model for adaptive WfMS. This model is based on the RBAC model. This option is justified because RBAC provides flexibility and efficiency over traditional approaches, as stated before. The RBAC model is extended and adapted by providing the definition of authorization states (authorizations) and then by describing mechanisms to support adaptive authorizations.

5.1 Authorizations

An authorization is represented as a quadruple (s, o, a, p) [26] and states that a subject s has an access mode a to perform an operation on an object o if the predicate p holds true (where p is optional). In this section we map the general notion of each component of an authorization to the adaptive workflow domain.

We also define rules for the derivation of implicit authorizations. Implication rules determine how implicit authorizations are derived from authorizations explicitly defined. The concept of implicit authorizations simplifies the definition of authorizations because it makes unnecessary to define and store all authorizations explicitly.

Subject In RBAC models, subjects are defined as roles rather than individual users. This way, if the predicate p holds true, a user u has an access mode a on object o , if there exists the authorization (r, o, a, p) , such that $u \in r$. Roles are created for the various activities of the process definition and users are assigned to roles based on their responsibilities and qualifications. However, since we are in the presence of a system that should be flexible enough in order to respond to changes efficiently, the workflow role model should be as independent as possible from the organizational model, in such a way that changes in the organizational model do not affect the workflow role model and vice-versa.

Therefore, subjects can be defined as dynamic roles. Roles and their hierarchy are defined locally to a process definition and roles are defined dynamically, i.e., instead of enumerating for each role the users that belong to it, the user to role assignment can be done as an expression on user attributes (designed by role definition rule). This expression is used, at runtime for user's session role resolution and activation, which can be used to query an external information system, such as, a directory service or a human resource application. Therefore, changes in the organizational model are reflected in the role model transparently and automatically.

Formally, let R be the set of the roles of a process definition, S the set of subjects of the organization model and $r_i dr$ the role definition expression for $r_i \in R$. Our role to user mapping is defined by the function:

- $role_resolution_function(r_i dr) \rightarrow 2^S$, this function maps each role r_i to a set of subjects by querying the information system that supports the organizational model.

Roles are hierarchically organized in the workflow role model. Let $r_i, r_j \in R$ be roles. We say that r_i dominates r_j in the hierarchy ($r_i > r_j$) if r_i precedes r_j in the ordering.

Object The second component is object. Authorizations can be applied to each of the following objects: the workflow meta-model, process definitions, process instances, activity definitions and activity instances.



Fig. 4. Workflow access control objects

We explicitly define the relationships existing between these objects, as shown in Figure 4, since they imply implicit authorizations. The process definition includes activity definitions, which are hierarchically organized. We use AD to indicate the set of activity definitions with a relation $>_{AD}$ to indicate inclusion relationship, PD to indicate the set of process definitions and the relation $>_{PD}$ between PD and AD to indicate inclusion relationship. Let $ad_k, ad_j \in AD$ be activity definitions, we say that ad_k includes ad_j ($ad_k >_{AD} ad_j$) if ad_j is a sub-activity of ad_k and we say that $pd \in PD$ includes ad_k ($pd >_{PD} ad_k$) if ad_k is an activity of pd .

Similarly, process instances include activities instances, which are hierarchically organized. We use AI to indicate the set of activity instances with a relation $>_{AI}$ to indicate inclusion relationship, PI to indicate the set of process instances and the relation $>_{PI}$ between PI and AI to indicate inclusion relationship. Let $ai_k, ai_j \in AI$ be activity instances and $pi \in PI$ be a process instance, we say that ai_k includes ai_j ($ai_k >_{AI} ai_j$) if ai_j is a sub-activity of ai_k in $pi \in PI$ and we say that pi includes ai_k ($pi >_{PI} ai_k$) if ai_k is an instance activity of pi .

Access Modes The set of access modes AM we consider in this model are: read, change, execute and grant. These access modes correspond to the read, write, execute and owner access modes widely used in operating systems. The read access mode allow read-only accesses, as follows, considering each object of our model, (1) read access mode for the workflow meta-model authorizes users to list process definitions, (2) read access mode for process definitions and for activity definitions authorizes users to read the definition, and (3) read access

mode for process instances and for activity instances authorize users to see their states.

The execute access mode is used to create new objects or to execute instances. Therefore, we have the following semantics for the execute access mode: (1) the execute access mode for the workflow meta-model authorizes users to create process definitions, (2) the execute access mode for process and activity definitions authorizes users to create (instantiate) instances of them (process instances and activity instances, respectively), and (3) the execute access mode for process instances and for atomic activity instances authorizes users to execute these instances (the execute access mode cannot be applied to composite activity instances because the unit of execution is an atomic activity instance).

Of special importance to adaptive workflow are the change access modes, which cover the following authorizations: (1) change access mode within the workflow meta-model authorizes users to remove process definitions, (2) the change access mode for process definitions authorizes users to define the process and to perform evolutionary changes, i.e., to change the process definition by, for example, adding or deleting activities (the same is applied for activity definitions), and (3) the change access mode for process instances authorizes users to perform ad-hoc changes, i.e., to change the process definition of this particular process instance (the same is applied for activity instances). Finally, the grant access mode is used to authorize users to change authorizations within the same object, i.e., the grant access mode for the workflow meta-model authorizes users to change the authorizations of the workflow meta-model, the grant access mode for a process definition authorizes users to change the authorizations of this process definition (the same is applied for activity definitions) and the grant access mode for a process instance authorizes users to change the authorizations of this process instance (the same is applied for activity instances).

In section 6, we exemplify how operations provided by an adaptive WfMSs can be mapped to this set of access modes.

Predicates Predicates can be associated with authorizations to restrict their applicability. Predicates can use the following variables:

- Attributes of the system, such as time and location of access,
- Attributes of the user, for instance, his age,
- Attributes of the objects that differ according to the objects, such as instantiation time of process instances and activity instances.

Predicates used in authorizations defined for activity instances can also use the following variables:

- Workflow history, i.e., information about activity instances already executed, such as the username of the executor,
- Values of their input data.

Predicates have the following syntax:

```

<predicate> ::= <conjunctive-predicate> {OR <conjunctive-predicate>}
<conjunctive-predicate > ::= <compare-predicate>
                             {AND <compare-predicate>}
<compare-predicate> ::= <left-value> <operator> <right-value>
<left-value> ::= <variable>
<right-value> ::= <constant> | <variable>
<operator> ::= '=' | '!=' | '<' | '>' | '<=' | '>='

```

Implication rules Implication rules are used to simplify the definition of authorizations. By using implication rules to derive implicit authorizations from explicit ones, the number of authorizations that need to be specified can decrease significantly.

Within our model we define the following implication rules:

Rule 1: the implication authorization rule on roles states that an authorization given to a role r propagates to all roles, which precede r in the role hierarchy (that is, to all roles r' such that $r' > r$).

Rule 2: the implication authorization rule on access modes states that change access mode implies read access mode.

Rule 3: the implication authorization rule on objects states that authorizations are propagated to the included activities, according to the inclusion relationships defined in section 5.1.2 ($>_{PD}$, $>_{AD}$, $>_{PI}$, $>_{AI}$). However, the authorization to execute a process instance should not imply authorizations to execute all its activity instances, therefore, the implication rule 3 does not apply to execute access mode defined for process instances.

Rule 4: the implication authorization rule on grant access mode states that change access mode implies grant access mode according to the inclusion relationships defined in section 5.1.2 ($>_{PD}$, $>_{AD}$, $>_{PI}$, $>_{AI}$). To define this implication rule, we take into account that, for example, the change access mode for process definition authorizes users to define the process by creating and defining activities or changing them. When these users are creating and defining activities, they also specify the authorizations of these activity definitions, exercising the grant access mode on them. The same can be applied to the change access mode for process instances, considering that the change access mode for process instances authorizes users to perform ad-hoc changes, i.e., to change the process definition of a particular instance.

5.2 Adaptive authorizations

Within our access control model, process definitions specify not only the behavioural and informational aspects of the workflow, but also the access control aspect. Therefore, adaptive workflows that comprise the access control aspect

cannot neglect mechanisms to support adaptive authorizations. Considering the access modes defined in section 5.1.3, evolutionary changes of authorizations can be done by exercising the change access mode of process definitions (or of activity definitions). Actually, the authorization to define or change the process definition also comprises the access control aspect. Similarly, ad-hoc changes of authorizations can be done by exercising the grant access mode of process instances as well as the change access mode of process instances (or of activity instances).

Subsequently, we describe how our model supports adaptive authorizations while maintaining both process definition and process instances correctness.

Static evolution of authorizations As stated by Casati [11] the problem of evolutionary changes has two facets: the static evolution facet and the dynamic evolution facet. The static evolution of authorizations facet refers to the issue of modifying the authorizations defined in the process definition, while maintaining its syntactical correctness.

In order to support this facet, the model must provide primitives to allow change operations and must guarantee that the new version of the workflow definition is syntactically correct.

By this way, our model provides two types of primitive operations, which allow access control information to be changed, as follows: (1) operations to add and delete authorizations and (2) operations to manage roles (add and delete workflow roles, add and delete hierarchical relationships between roles, and add and delete role definition rules).

We define the notion of access control aspect model correctness based on model invariants that must be preserved (this is similar to the approach taken in [15], defined for the other workflow aspects). Invariants are conditions over the process definition that have to hold before and after change operations. Following is a list of invariants for the access control aspect:

- Authorization invariant: an authorization can only specify roles defined in the workflow role model, access modes defined in the model (section 5.1.3), objects of the system and valid predicates (see predicate invariant).
- Predicate invariant: requires that predicates can only refer to existent workflow relevant data and, in case of history predicates, defined workflow activities. This invariant can be violated by a change operation on other workflow aspect if that operation affects variables used by the predicate. Therefore, we have identified the operations that can have side effects on predicate validity, which are: the operation that removes an input data of an activity and the operation that removes an activity.
- Role invariant: role hierarchical relationships and role definition can only refer to roles defined in the workflow role model.

Dynamic evolution of authorizations The dynamic evolution facet refers to the problem of managing running instances of a process, whose access control

aspect definition has been modified. As stated before, the dynamic evolution facet requires mechanisms to support the migration of existing instances to another process definition, without affecting their correctness, by ensuring their compliance with the new process definition, i.e., by ensuring that the process instance history is compliant with the new process definition.

Considering the access control aspect, the process instance history comprises information about the user that has initiated the execution of the process instance and the users that have executed activity instances already performed. Therefore, a process instance is compliant with a process definition if (1) the user that has initiated the execution of the process instance is valid taking into account authorizations to execute process instances and if (2) the users that have performed already executed activity instances are valid taking into account authorizations to execute these activities.

Following the approach presented by Kradolfer [15], we have defined migration conditions for each of the access control change operations that can affect instances compliance, as follows:

Condition 1: delete an authorization specifying execute access mode for process instances: the process instance can be migrated to the new process definition if the subject that instantiated it is still authorized considering the remaining authorizations.

Condition 2: delete an authorization specifying execute access mode for a manual activity instance: the process instance can be migrated if (1) this activity instance has not been executed yet or, (2) it has already been executed and the subject that has executed it is still authorized considering the remaining authorizations.

6 Implementation in WorkSCo

In this section we overview the WorkSCo project³ architecture and we describe how operations provided by this adaptive WfMS can be mapped to the set of access modes defined in section 5. This description is illustrated using the simplified loan workflow application example.

The Workflow with Separation of Concerns (WorkSCo) project is being developed based on the new workflow architecture presented in [6] designated by micro-workflow. The WorkSCo framework was developed using techniques specific to object systems and compositional software reuse. It targets software developers and provides the type of workflow functionality necessary in object-oriented applications. WorkSCo has a lightweight kernel that provides basic workflow functionalities and offers advanced workflow features as components that can be added to the kernel. Software developers select the features they need and add the corresponding components to the kernel through composition.

The evolution component uses a meta-model approach. It explicitly supports process definition versioning and workflow instances migration [15]. The idea

³ <http://www.esw.inesc-id.pt/workscsco>

behind process definition versioning is not to update process definitions in place, but version them. Those workflows that cannot be migrated can continue their execution under the old version.

Within this approach, evolutionary changes are supported by creating a new process definition version and performing changing operations on this new version. Ad-hoc changes to specific process instance are supporting by creating a variant of the process definition and migrating the process instance to this variant.

In the following we illustrate the application of the access model presented in the previous section to the WorkSCo evolution component, using the simplified workflow application example described in section 4 and shown in Figure 2. Initially, the WorkSCo WfMS is configured with the workflow meta-model authorizations, which states who can create, remove and list process definitions and who can change these authorizations.

Considering a life cycle of process definitions and process instances, next we exemplify (1) the creation of the process definition, (2) the definition of a process definition, (3) an instantiation of a process, (4) an evolutionary change on the process definition and (5) a migration operation.

(1) Firstly our bank decides to define a new process for the loan workflow application. The WfMS's manager creates a new process definition, by exercising the execute access mode of the workflow meta-model. This process definition will be defined by the role *loan workflow designer*. Therefore, the process definition has to be created with the authorization (*loan workflow designer*, definition of *loan workflow*, *change*).

(2) Next the process is defined. A user playing the role *loan workflow designer* creates and defines its four activities, the control flow, the data flow and the access control aspect information. The access control aspect includes the roles of the process: *clerk*, *account manager*, *bank manager*, *loan manager of central bank* and *loan workflow designer* and its authorizations: (*clerk*, instances of *loan workflow*, *execute*). A user playing the role *loan workflow designer* can also define and change the activities that he will create without needing more authorizations. Indeed, implication rules defined in our model let these authorizations be omitted. This user also defines the following authorizations: (*clerk*, instances of *receive loan request*, *execute*), (*account manager*, instances of *evaluate loan*, *execute*, "requestor is a bank client") and (*bank manager*, instances of *evaluate loan*, *execute*, "requestor is not a bank client").

(3) Execution of process instances can be initiated by role *clerk* as stated by the authorization (*clerk*, instances of *loan workflow*, *execute*). Instances of activity *receive loan request* can be performed by role *clerk*, defined by authorization (*clerk*, instances of *receive loan request*, *execute*), while instances of activity *evaluate loan* can be performed by *account managers* if the requestor is a bank client or by *bank manager* otherwise, defined by authorizations: (*account manager*, instances of *evaluate loan*, *execute*, "requestor is a bank client") and (*bank manager*, instances of *evaluate loan*, *execute*, "requestor is not a bank client").

(4) As a result of the definition of more restricted rules for loan process definitions, authorizations defined for activity *evaluate loan* have to be changed. This change should be done by users playing role *loan manager of central bank*. Therefore, firstly, the authorization (*loan manager of central bank*, definition of *evaluate loan*, *change*) should be added. Users playing the role *loan workflow designer* can add this authorization exercising the authorization (*loan workflow designer*, definition of *loan workflow*, *change*), which, applying implication rule 4, derives the implicit authorization (*loan workflow designer*, definition of *evaluate loan*, *grant*). Then, a user playing role *loan manager of central bank* can remove the two authorizations defined within the activity *evaluate loan* and can add the two new authorizations: (*loan manager of central bank*, instances of *evaluate loan*, *execute*, “score greater or equal than 5”) and (*bank manager*, instances of *evaluate loan*, *execute*, “score less than 5”). The operation that adds authorizations evaluates their invariants, ensuring correctness.

(5) Finally, a user playing role *loan manager of central bank* has to migrate running process instances to the new process definition, such as the Maria’s process instance shown in Figure 3. Therefore, two authorizations have to be defined: (*loan manager of central bank*, instances of *evaluate loan*, *change*) or (*loan manager of central bank*, instances of *evaluate loan*, *grant*), and (*loan manager of central bank*, definition of *evaluate loan*, *execute*). Users playing role *loan workflow designer* can add these authorizations by exercising the authorization (*loan workflow designer*, definition of *loan workflow*, *change*) and applying implication rules 3 and 4, respectively.

Both authorizations (*loan manager of central bank*, instances of *evaluate loan*, *change*) and (*loan manager of central bank*, instances of *evaluate loan*, *grant*) state that the role *loan manager of central bank* can change authorizations of *evaluate loan* instances, while the other authorization (*loan manager of central bank*, definition of *evaluate loan*, *execute*) states that the same role can instantiate the *evaluate loan* activity definition, i.e., can migrate instances to it.

Next, the migration operation has to ensure instance correctness by ensuring its compliance with the new process definition. Considering Maria’s instance process, it is compliant with the new process definition because all the migration conditions hold. If, for example, the instance of the activity *evaluate loan* had already been executed by an *account manager*, it would not be compliant with the new process definition and the migration could not be performed.

7 Conclusions and Future Work

Recently, WfMSs have been extended in order to support both ad-hoc and evolutionary changes. These features raise new access control requirements that are not met by traditional workflow access control models.

In this paper, we identify access control requirements of adaptive WfMSs and present an access control model that meets these requirements. Our model extends the RBAC model by defining authorizations and mechanisms that support their evolution. Authorizations are defined taking into account the objects

of adaptive WfMSs that need to be protected and the access modes that users can exercise on them. The presented model also supports dynamic roles and defines implication rules to derive implicit authorizations. To support the evolution of authorizations, this model defines an administrative policy and mechanisms to maintain the correctness of process definition as well as process instances.

The development of our model has been influenced by the following works:

- Adaptive WfMSs [11, 15, 14] – these works propose mechanisms to support adaptive workflows, by ensuring process definition correctness and process instances correctness. Our approach adapts these mechanisms in order to deal with access control aspects.
- Access control models for object oriented databases [26] - access control models for object oriented databases differentiate two types of protected objects: class and set of instances and define rules for the implication between authorizations along each of the three dimensions. These concepts have been adapted to the adaptive WfMSs domain.
- Dynamic roles [7] - within this work, dynamic role resolution is done when authorizations are evaluated. In our approach, dynamic role resolution is done when users establish sessions.

At present, we are also working on testing the applicability of our model in large workflow applications, such as health care workflow applications and emergency planning workflow applications for civil protection.

References

1. Agostini, A., Michelis, G.: A Light Workflow Management System Using Simple Process Models. In *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, Kluwer Academic Publishers, Vol.9, No.3-4 (2000), 335-363.
2. Atluri, V., Huang, W.: An authorization Model for workflows. In *Proceedings of the 5th European symposium on research in computer security*, Rome, Italy (1996) 44-64.
3. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraint in workflow management systems. In *ACM Transactions on Information and System Security*, Vol.2, No.1 (1999) 65-104.
4. Botha, R.A., Eloff, J.H.P.: Separation of Duties for Access Control in Workflow Environments. In *IBM Systems Journal* Vol. 40, No. 3 (2001) 666-682.
5. Casati, F., Castano, S., Fugini, M.: Managing Workflow Authorization Constraints through Active Database Technology. *Information Systems Frontiers*, Vol.3, No.3 (1999).
6. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and Dynamic Service Composition in eFlow. In *Proceedings of CAISE 2000*, Stockholm, Sweden (2000).
7. Casati, F.: *Models, Semantics, and Formal Methods for the design of Workflows and their Exceptions*. Ph.D.Thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano (1998).
8. Castano, S., Fugini, M., Martella, G., Samarati, P.: In *Database Security*. Addison-Wesley (1994).

9. Cheng, E.C.: An Object-Oriented Organizational Model to Support Dynamic Role-based Access Control in Electronic Commerce Applications. In Proceedings of the 32nd Hawaii International Conference on System Sciences (1999).
10. Georgiadis, C.K., Mavridis, I., Pangalos, G., Thomas, R.K.: Flexible team-based access control using contexts. In Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies (2001), 21-27.
11. Han, Y., Sheth, A., Bussler, C.: A Taxonomy of Adaptive Workflow Management. In Workshop on Adaptive Workflow Systems at the 1998 Conference on Computer-Supported Cooperative Work, Seattle, USA (1998).
12. Holbein, R., Teufel, S., Morger, O., Bauknecht, K.: Need-to-Know Access Control for Medical Systems. In: Information Security - the Next Decade IFIP/SEC '97 (1997).
13. Hollingsworth, D.: The Workflow Reference Model. Document Number TC-00-1003. Issue 1.1. 19 January 1995.
14. Joeris, G., Herzog, O.: Managing Evolving Workflow Specifications with Schema Versioning and Migration Rules. TZI Technical Report 15, Center for Computing Technologies, University of Bremen (1999).
15. Kandala, S., Sandhu, R.: Secure Role-Based Workflow Models. In Database Security XV: Status and Prospects, Kluwer (2002).
16. Kang, M.H., Park, J.S., Froscher, J.N.: Access Control Mechanisms for Inter-Organizational Workflow. In Proceedings of the 6th ACM Symposium on Access Control Models and Technologies SACMAT 2001, Chantilly, VA (2001), 66-74.
17. Kradolfer, M.: A Workflow Metamodel Supporting Dynamic, Reuse-Based Model Evolution. Dissertation, Institut für Informatik, Universität Zürich (2000).
18. Manolescu, D.: Micro-workflow: a workflow architecture supporting compositional object-oriented software development. PhD Thesis. University of Illinois at Urbana-Champaign (2001).
19. Miller, J.A., Fan, M., Wu, S., Arpinar, I.B., Sheth, A.P., Kochut, K.B.: Security for the METEOR Workflow Management System. Uga-cs-Isdis technical report, University of Georgia (1999).
20. Reichert, M., Hensinger, C., Dadam, P.: Supporting Adaptive Workflows in Advanced Application Environments. In Proceedings of EDBT Workshop on Workflow Management Systems, Valencia, Spain, (1998).
21. Samarati, P., Vimercati, S.: Access Control: Policies, Models, and Mechanisms. In Focardi, R., Gorrieri, R. (eds): Foundations of Security Analysis and Design, LNCS 2172, Springer-Verlag (2001).
22. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models. IEEE Computer, Vol. 29, No. 2 (1996).
23. Sheth, A., Kochut, K.J.: Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems. Eds Doga, A., Kalinichenko, L., Ozsu, M.T., Sheth, A., NATO Advanced Study Institute, Istanbul, Turkey (1997).
24. Sheth, A.: From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Co-ordination and Collaboration. In Proceedings of the Workshop on Workflow Management in Scientific and Engineering Applications, Toulouse, France (1997).
25. Siebert, R.: An Open Architecture for Adaptive Workflow Management Systems. In International Workshop on Issues and Applications in Database Technology (IADT'98), Berlin (1998).

26. Thomas, R. K.: Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments. In Proceedings of the second ACM Workshop on Role-based Access Control (1997) 13-19.
27. Thomas, R., Sandhu, R.: Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management. In Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California (1997).
28. Wang, W.: Team-and-Role-Based Organizational Context and Access Control for Cooperative Hypermedia Environments. In Proceedings of the tenth ACM Conference on Hypertext and Hypermedia (1999) 37-46.
29. Wu, S., Sheth, A., Miller, J., Luo, Z.: Authorization and Access Control of Application Data in Workflow Systems. In Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies (JIIS), Vol. 18, No. 1 (January 2002) 71-94.