AW-RBAC: Access Control in Adaptive Workflow Systems

Maria Leitner, Stefanie Rinderle-Ma
University of Vienna
Faculty of Computer Science
Vienna, Austria
{*maria.leitner,stefanie.rinderle-ma*}*@univie.ac.at*

Juergen Mangler
SBA Research
Vienna, Austria
jmangler@sba-research.org

***Abstract*—Flexibility is one of the key challenges for Workflow Systems nowadays. Typically, a workflow covers the following four aspects which might all be subject to change: control flow, data flow, organizational structures, and application components (services). Existing work in research and practice shows that changes must be applied in a controlled manner in order to avoid security problems. In this context, attempts have been made to manage administrative or operative changes using role-based access control (RBAC) models. However, most approaches focus on either administrative changes such as role updating and administration or operative changes, for example, inserting a new activity into a running workflow instance. The distinct handling of certain changes is cumbersome and hence should be reduced by introducing a RBAC model that pays attention to all kinds of possible workflow changes. Hence, in this paper, we present an extended RBAC model for adaptive workflow systems (AW-RBAC) that includes change operations and a variety of objects that are subject to change within workflow systems. Under such a model supervised administrative and operative changes can be enforced on a set of objects in workflow systems. Doing so, the AW-RBAC model improves security during workflow changes and reduces administration costs. The AW-RBAC model is evaluated by means of practical examples and a proof-of-concept implementation.**

***Keywords*-RBAC; Access Control; Process-Aware Information Systems;**

I. Introduction

Adequate support of business processes constitutes one of the most important challenges for enterprises nowadays. Workflow Systems (WfS) enable the execution of enterprise-wide and cross-organizational processes, coordinate the process-oriented integration of application components, support users in a process-oriented way, monitor process progress, and document the process execution in a comprehensive way [1]. Business processes and workflows both are multi-faceted. Typically, a workflow consists of control flow and data flow that administrate which activities are executed and which data is passed between these activities. Furthermore, access to activities is granted based on an organizational model that reflects the corresponding organizational structure of the enterprise. Finally, workflow activities are connected with services (i.e., application programs) that are invoked during runtime by passing data from the workflow activity to the service and vice versa.

Research [2], [3] shows that security is of importance in WfS. Consequently, providing proper security mechanisms at all levels of workflow systems is hence a non-trivial task. The issue of providing proper security mechanisms in WfS is aggravated when considering adaptive WfS where changes and modification might be required at all different levels. Typically, the workflow control and data structures are frequently subject to change due to exceptional handling or evolutionary changes. Current research prototypes partly allow changes while commercial systems enable only very basic change operations [4]. In addition, organizational structures might be adapted due to organizational changes such as deleting organizational hierarchies or merging departments. Moreover, for repairing workflows at runtime the replacement of assigned services might be also required. Controlling such changes is particularly important since uncontrolled change might lead to undesired effects and security violations, for example, injection of undesired behavior into the workflow structures and granting access to workflow data to unauthorized users.

Most existing approaches to control access in WfS are based on Role Based Access Control (RBAC). Here, organizational structures related to a workflow are modeled within a RBAC model. During runtime, access rights / permissions based on either access rules or task-based assignment (Task-Based RBAC) are determined and passed to the WfS to restrict access of activities only to authorized users. Access control for workflow execution is crucial. However, proper access control for workflow change poses an equally important question. Only few approaches have tackled this issue so far. In [5], [6], administrative changes of the RBAC model are handled with the focus on the correct and consistent application of such changes. Though the basic idea of administrative changes can be used for control organizational changes in WfS, no other workflow changes are considered. The work presented in [7] provides an extended RBAC model to authorize control flow changes in WfS. This constitutes a first milestone on the way to secure adaptive WfS. However, access control to address changes at organizational and service level is not considered, though changes might also pose security threats to the WfS.

Hence in this paper, we propose an extended RBAC model

for adaptive WfS (AW-RBAC). This model enables WfS to enforce access control under all different kinds of workflow changes. Specifically, the AW-RBAC model enables the definition of permissions in order to perform authorized control and data flow changes, administrative modifications and adaptations at the service level of a WfS. In addition, access rights based on AW-RBAC can be specified in fine granularities based on the definition of constraints. The application of the AW-RBAC model is evaluated based on several use cases and a proof-of-concept prototype.

We introduce the architecture of WfS in Sect. II. Sect. III provides the access control requirements for adaptive WfS. In Sect. IV, we present the AW-RBAC model. A case based evaluation is shown in Sect. V, followed by a demonstration of the prototype implementation in Sect. VI. We discuss related work in Sect. VII and conclude in Sect. VIII.

II. Adaptive Workflow Systems

The general architecture of a WfS is shown in Fig. 1. At design time, process models are designed and stored within a process repository. Existing systems such as ADEPT [8], Staffware, or WebSphere [9] typically control access to process activities by assigning access rules to activities that are resolved over the associated organizational structures (e.g., represented by RBAC models) at runtime.

At runtime, the workflow engine manages the process instances as well as data and services which both are used in the process instances. While an activity is an abstract functionality with input and output data elements, at runtime for an activity a concrete service is invoked and executed. Services in WfS typically include applications, web services or calls to databases. It is important to note, that for a given activity, several services that can return valid results may exist in a service repository. A worklist is generated for each active and authorized user displaying open tasks. Monitoring components provide means to, for example check the current state of a certain workflow instance or to see the results of applying a change operation.

Basically, the following four main workflow *change categories* can be identified: administrative, control flow, data flow, and service changes (Figure 1 indicates at which part of the architecture the change operations are handled): When organizations change (e.g., by outsourcing departments) the corresponding organizational model, e.g., represented by a RBAC model, needs to be adapted accordingly. In this paper, we use the term *"Administrative Changes"* for organizational changes and for functional adjustments of job functions, permissions, and constraints (i.e., granting permissions to a role, adding time constraints to a role-permission assignment). In [10], we provide a formal framework for administrative changes. Control flow adjustments are alterations on the structure of a process model (schema evolution) or of certain process instances (ad hoc changes). An example is inserting a new *activity A* into a given process.

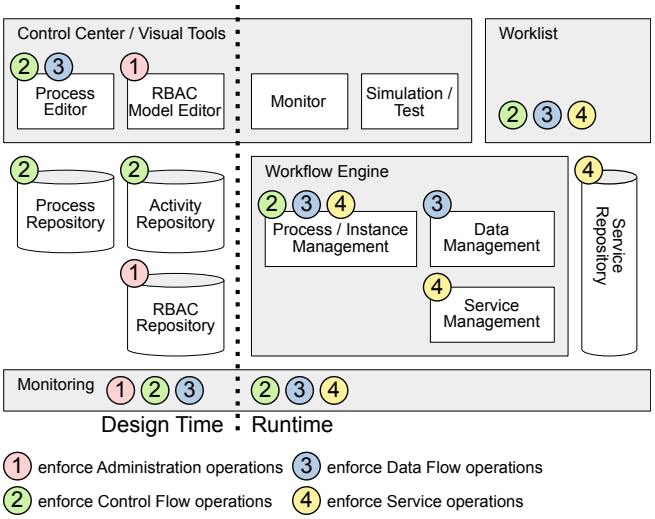

Figure 1. Typical WfS Architecture

Data flow changes insert or delete new data elements or connect data elements with process activities. Often data flow adaptations become necessary in the context of control flow changes. For an overview on control and data flow change patterns see [4]. Service changes are adjustments on functionalities (i.e., applications, web services) connected and communicating with the WfS to support the completion of activities (i.e., when *service D* is unavailable and therefore is replaced with *service E*).

Currently, only few adaptive WfS exist [4] and they – if at all – only supervise control flow changes. Other adjustments are performed but are not authorized. This might lead to unintended errors, security violations, or even workflow blockage. Imagine the following scenario: A WfS of a bank in country F exists where a user X is able to change its permissions to access private client data. The person X copies the data and offers it to another nations department of the treasury. To avoid such security risks, an access control system for all kinds of workflow changes is indispensable.

III. Access Control Requirements for Adaptive Workflow Systems

Adaptive workflow systems need specific access control requirements in order to support all functional aspects. More precisely, it must be possible to restrict access to workflow activity execution as well as the authorization to perform workflow changes. The latter is important to avoid misuse or mistakes when performing workflow changes possibly leading to security violations such as injecting undesired behavior into the workflow. So far, access control in adaptive workflow systems disregarded administrative and advanced operative changes, e.g., modifications at the service level [11], [7]. Hence, as a first requirement on access control in adaptive WfS we claim that all major change categories as set out in Sect. II must be supported within one access

control model. This requirement also implies that the control of different change categories must not be outsourced into different access control models, as for combining the RBAC model for administrative changes proposed in [6] and the extended RBAC model for controlling control flow change presented in [7]. Reason is that the distribution of access rights over several access control models might hamper the maintenance of these models. Even worse consistency checks of access rights might be prohibited at all.

Requirement 1 (Completeness): Workflow functionality including change is covered by one access control model. This comprises access to activity execution as well as the *supervision* of the following four change categories:

(A) Administrative changes
(B) Control flow changes
(C) Data flow changes
(D) Service changes

Ignoring the supervision of change operations might lead to security violations such as unauthorized access (e.g., to confidential data), accidental errors (e.g., deleting a user), criminal actions (e.g., spying on business secrets), workflow exceptions and blockage (e.g., due to the deletion of a data element that provides information for the activity), or security holes (e.g., due to adding a malicious service).

Requirement 2: Support of user and context dependent access rights

Including context information such as location or time within access rights is an important claim in general. User dependent access rights enforce that only authorized users are able to enter the system and its resources (e.g., only *Senior Accountants* are allowed to give *Credits* to customers). Furthermore, the business process context matters in workflow systems. Only context matching changes should be allowed (e.g., activity *transferSalary* should not be inserted into the process *claimTravellingExpenses*).

In addition, we include the following two requirements as claimed by [7]) into our discussion:

Requirement 3: Maintainability of access rights

In access control systems with a large number of roles and users, it is essential that access rights are easy to add, modify, and delete (i.e., update a permission). Additionally, context specific characteristics (i.e., process instance, process schema) are omitted to keep the RBAC model (and further its maintainability) minimal.

Requirement 4: Fine-grained definition of access rights

A fine-grained definition of access rights allows the definition of individual change operations according to context dependencies. For example, an *Analyst* should be authorized to perform control flow changes (e.g., deleting an activity) but should be limited to specific processes (e.g., *clientRequest*), scenarios (e.g., if no investigation is necessary), or activities (e.g., *analyzeRequest*). The access control model must ensure the separation of data, process, and security related aspects.

In Sect. VII we provide a comparison of selected RBAC models along the four requirements stated above.

IV. Adaptive Workflow RBAC Model

Th design of the AW-RBAC model depicted in Fig. 2 follows the requirements set out in Sect. III. In addition, it follows the following basic design decisions: (a) reflecting the NIST RBAC standard as far as possible and (b) separate all context specific information from the definition of access rights (i.e., specify context information as constraints).

The term *User* refers to humans as well as machines, in that access should also be restricted for external systems such as monitors or repair engines.

The concept of a *Role* allows to structure users according to their capabilities as well as to structure the *Roles* themselves. This is useful for describing organizational hierarchies (or graphs) with nested organizational units.

We use *Object* and *Operation* together as basis for defining *Permissions*. While *Object* refers to critical concepts found in WfS, *Operation* describes how these *Objects* may be used. For example, to adapt the AW-RBAC model (Object.Category: Administration) a user (Object.Type: User) may be added (Operation: add).

The purpose of *Permission* is to describe which combinations of *Operation* and *Object* may exist as well as constrain their actual use. This restriction may be according to **data** (e.g., a user may only grant a credit - execute a process activity - if the credit sum is smaller than € 10000), **time** (e.g., a process activity may be executed only during a certain time period), or to concrete instances of *Object* (a set of processes, process instances, or activities).

Finally the semantic of *Session* is highly dependent on the *Object*. To unify relation of *Session* to *User* and *Role* we decided to narrow the meaning of *Session* in comparison to the definition in the original NIST RBAC model, where a session is defined as "mapping of one user to possibly many roles", e.g., the duration a user is logged into his worklist.

- For *Administration* it refers to one operation, e.g. change of the organization structure or other administrative objects.
- For *Control Flow* it refers to one adaption of a particular process model or process instance (applied by process designers as well as repair engines). For the *monitor* operation it restricts the view of a system or user on the progress of a process instance (which activities have already been executed). Thus for *monitor*, a *Session* exists for the lifetime of a process instance.
- For *Data Flow* it refers to one data flow adaption in a process model or process instance. This includes changing data element values at runtime during repair. For the *monitor* operation the semantic of a *Session* is identical to *Control Flow*.
- For *Service* a *Session* exists for the period of time during which an activity is executed. As defined above,

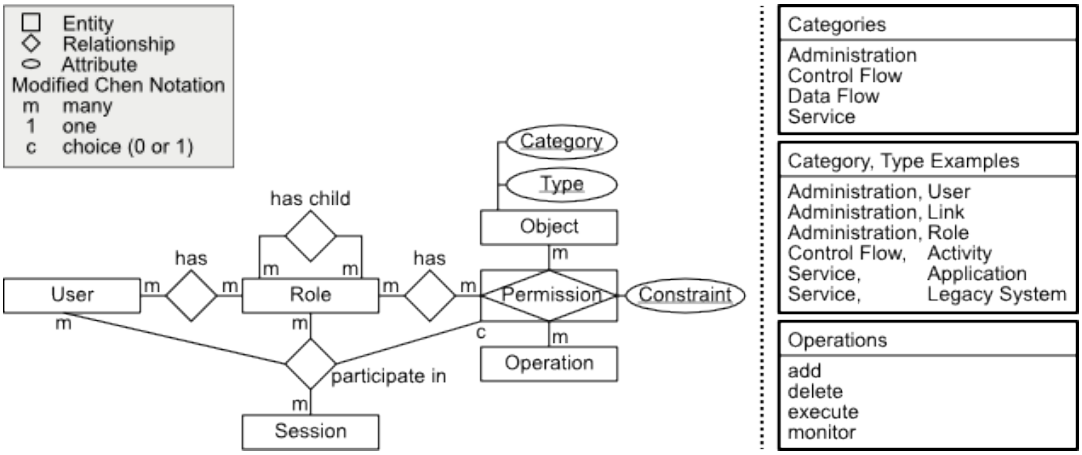

Figure 2. An Access Control Model for Adaptive Workflow Systems (AW-RBAC)

there may be a set of service implementations from which to choose at runtime.

As shown in Fig. 2 a session can also be confined by a permission. This means that a session may be restricted independent from the permissions that a user inherits from a certain role. For example, when several users have the right to execute a particular activity, an administrator may exclude some users for this session (e.g. the users currently on leave).

In the following we will explain the semantics of the relations shown in Fig. 2, with a special focus on the original NIST RBAC model. The following pieces are similar to the original model:

User:Role (m:m) A user has several roles.

Role:Role (m:m) In a role hierarchy, a role can have one or more parents and one or more children. A role being parent or child of itself is not allowed, loops are not allowed.

Role:Permission (m:m) A role can have several permissions attached.

The **User:Role:Session (m:m:m)** relation depends on our definition of session.

(User,Session):Role. A user can have several roles regarding a certain session. For example, a user can execute an activity as an *Accountant* (with specific permissions) or as a *Supervisor* (with different permissions) with the rights to adjust values not changeable by the *Accountant*.

(Role,Session):User. For a particular role connected to a particular session, several users may be available. An activity may be available to several users for concurrent work. An important scenario for WfS is the following: several users are allowed to execute an activity. As soon as the first user activates an activity, the other users are no longer allowed to execute the activity. As mentioned above this can be solved easily by adding a restriction (*Permission*) to all Role:Session:User combinations that are no longer allowed to execute the activity.

(Role,User):Session. For a particular user with a particular role, several sessions may exist (e.g. execution of activities in different instances).

Consequently, entries in a user worklist, for example, have particular roles attached and represent multiple sessions, with sessions being process instances.

(User,Session,Role):Permission represents the optional confinement as described above.

The relation between *Operation* and *Object* is assigned a *Permission*:

Subject of the Permission: the combination of *Object* and *Operation*.

Characterization of the Permission: the attribute *Constraint* connected to *Permission*.

The main ingredient of a permission is the *Constraint*. The structure and semantic of a *Constraint* strictly depends on the *Object* and *Operation* it connects. As shown in Figure 1 by colored circles, the above defined four object categories (1) **Administration**, (2) **Control Flow**, (3) **Data Flow**, and (4) **Service** have to be enforced in various parts of a WfS. Enforcing means that each of the marked components has to actively ensure that each permission connected to a particular *User*, *Role*, and *Session* is satisfied. The purpose of *Constraint* is to help the component identify the permissions that apply in a particular context. Thus the generic form of *Permission* is:

(*Operation*, *Object.Category*, *Object.Type*, *Constraint*)

As mentioned above, the semantic of the *Constraint* is tied to *Object*:

Administration: As Administration deals with changing the AW-RBAC model itself, the constraints describe where a particular *User* is allowed to insert / delete roles and users. Example: (*delete*, *Administration*, *Role*, *roles that are children of "Accountant"*).

Control Flow: For this category a constraint may refer to certain processes, instances, or activities that may be changed, monitored or executed. Example: (*add*, *Control Flow*, *Activity*, *arbitrary new activities in process "Invoice"*).

Data Flow: For this category a constraint may refer to certain processes, instances, or activities where data elements may be changed or monitored. Example: (*monitor*, *Data Flow*, *Data Element*, *changes of data element "Costs" in all instances a process "Credit Grant"*).

Service: For this category a constraint may refer to an activity (in a particular process / instance) and a set of services. Example: (*delete*, *Service*, *Web Service*, *for activity "creditworthiness" in process "Credit Grant" select one of: http://a, http://b*).

This interpretation of the RBAC model in combination *Object* / *Operation* extension provides a simple yet fine-grained holistic access control model. To the best of our knowledge not yet found in other RBAC extensions.

V. Use Case based Evaluation

In the following, we illustrate the AW-RBAC model by means of three use cases reflecting administrative, data flow, and service changes. Examples for securing control flow changes can be found in [7].

A. Administrative Change

An organizational model is depicted in (cf. Fig. 3) and formalized in Table I. In this setting, obviously, `Smith` is allowed to assign an additional permission `p3` to its role `Accountant` and, consequently, would be allowed to perform activity `evaluateRequest` afterwards.

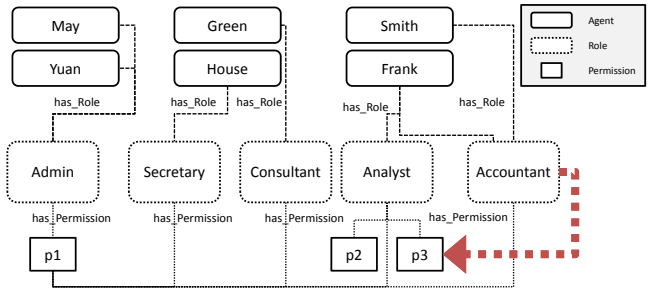

Figure 3. Granting an additional permission to a role

Current adaptive workflow systems would have allowed this new assignment because they do not support the restriction of change rights to users. Hence, any user is able to increase its authority like `Smith`. This risky setting can lead security violations and errors (i.e., a user accidentally deletes a role) as well as aids criminal actions (i.e., fraudulent actions, spying of business secrets, accessing private employee data). By constrast, using the AW-RBAC model, all permissions can be restricted to authorized users such as the `Admin` as done in Table II).

Table I
Selection of general assignments

General Assignments	
User =	{May, Yuan, Green, House, Smith, Frank}
Role =	{Admin, Secretary, Consultant, Analyst, Accountant}
UserHasRole =	{(May,Admin),(Yuan,Admin),(Green,Consultant), (House,Secretary),(Smith,Accountant),(Frank,Analyst), (Frank,Accountant)}
RoleHasPerm =	{(Admin,p_1),(Secretary,p_1),(Consultant,p_1), (Analyst,p_1),(Accountant,p_1), (Analyst,p_2),(Analyst,p_3))}
P =	$\{p_1\} \cup \{p_2\} \cup \{p_3\}$
p_1 =	(execute,ControlFlow,Activity,c_1)
p_2 =	(execute,ControlFlow,Activity,c_2)
p_3 =	(execute,ControlFlow,Activity,c_3)
c_1 =	(processName=clientRequest, activityName=formalCheck)
c_2 =	(processName=clientRequest, activityName=analyzeRequest)
c_3 =	(processName=clientRequest, activityName=evaluateRequest)

Table II
Selection of an extended permission set for administrative change

Role-Permission Assignments	
$RoleHasPerm_1$ =	RoleHasPerm $\cup$ {(Admin,p_4),(Admin,p_5), (Admin,p_6)}
P_1 =	$P \cup \{p_4\} \cup \{p_5\} \cup \{p_6\}$
p_4 =	(add,Administration,User,c_4)
p_5 =	(add,Administration,Link,c_5)
p_6 =	(add,Administration,Role,c_6)
c_4 =	(setName=true)
c_5 =	(role=Accountant)
c_6 =	(user=May)

B. Data Flow Change

Data flow changes occur regularly in workflow systems. On the one side data flow changes are often accompanying control flow changes, but also become necessary, in the context of dealing with exceptional situations. Imagine, for example, that a mandatory data element cannot be provided by the planned resource and thus has to be provided by another data source. Particularly here, data flow change operations can also be source for misuse and consequently, must be reviewed and secured.

The example in Fig. 4 displays agent `Frank` deleting a data edge `(Formal Check, Result, write)` writing data element `Results` that is required for consecutive workflow activity `Analyze Request`. What are the consequences of this change? As soon as activity `Analyze Request` is started, there will be an exception within the workflow execution. Then it depends on the specified recovery mechanisms if the workflow execution can be restarted or the workflow is blocked.

Based on the AW-RBAC model, applying data flow changes can be restricted to authorized users. In this case,

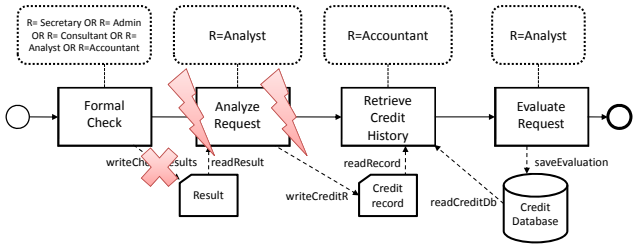

Figure 4. Deletion of a data edge in process clientRequest

only agents having role `Admin` are permitted to carry out the adjustment (cf. Table III).

Table III
Selection of an extended permission set for data flow change

Role-Permission Assignments	
RoleHasPerm_3 =	$\text{RoleHasPerm}_2 \cup \{(\text{Admin}, p_7), (\text{Analyst}, p_8)\}$
p_7 =	(delete,DataFlow,DataEdge,c_7)
p_8 =	(add,DataFlow,DataEdge,c_8)
c_7 =	(processName=clientRequest, dataEdgeName=writeCheckResults)
c_8 =	(processName=clientRequest, dataEdgeName=readECBdb)

C. Service Change

Generally, a service is a functionality (e.g., application, legacy system, web service) that can communicate and operate with the WfS. Service changes are becoming more and more custom. Particularly in agile environments, the ad hoc service composition of workflows has gained great interest lately. At design time workflow activities are linked to services and the services are invoked during runtime. The data mapping between the workflow activity and service has to be secured by matching the input/output parameters syntactically. Semantical checks on data are not included this paper but subject to future work.

Security between the WfS and an external service is defined on basis of service level contracts. The contracts include service level attributes as well as security relevant items (e.g. encryption level, data privacy policy, authorized agents, availability). It must be ensured that the service contract with the new service has at least the same security level or higher as the previous service. Exchanging data with less secured services might lead to security threats (e.g. data leaks, unauthorized data access, exposition to attackers). Hence, service changes should be monitored and supervised by an agent or a service (e.g., monitoring service).

Consider, for example, that `Booking Service` is replaced by `New Booking Service` (see Fig. 5). The new service must provide at least the same service level security policies as the former service. Additionally, the input/output parameters must match between the workflow activity and the new service. Until now, service changes have not been supervised nor monitored in workflow systems. An unexperienced user, for example, agent `House` is only used to execute the secure service `Booking Service`. Due to the unavailability of `Booking Service`, the user replaces the secure transaction service with a unreliable one which might lead to privacy issues (e.g., leaking of client data). Therefore, we must enable a supervised service change.

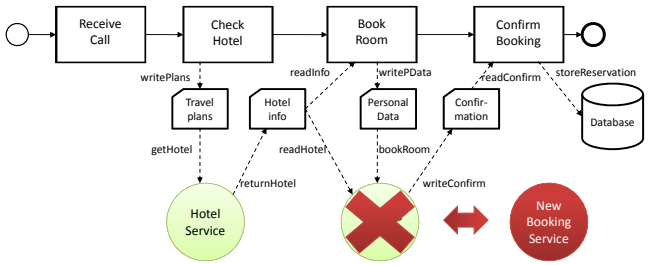

Figure 5. Service exchange in process bookHotel

In AW-RBAC model, we can specify permissions restricting the access of services (cf. Table IV). In this case, only agents having role `Admin`, namely `May` and `Yuan`, are authorized to perform the service exchange. Thus the security breach described before is prevented. By monitoring the service exchange, we enable a secure transition and reduce security violations.

Table IV
Selection of an extended permission set for service change

Role-Permission Assignments	
RoleHasPerm_4 =	$\text{RoleHasPerm}_3 \cup \{(\text{Admin}, p_9), (\text{Admin}, p_{10}), (\text{Admin}, p_{11}), (\text{Admin}, p_{12})\}$
p_9 =	(delete,Service,BookingService,c_9)
p_{10} =	(delete,Service,LinkGroup(readHotel$_1$, bookRoom$_1$,writeConfirm$_1$),c_9)
p_{11} =	(add,Service,NewBookingService,c_9)
p_{12} =	(add,Service,LinkGroup(readHotel$_2$,bookRoom$_2$, writeConfirm$_2$),c_9)
c_9 =	(processName=bookHotel, activityName=bookRoom)

In summary, security incidents due to workflow changes can be prevented by using access control mechanisms. All use cases demonstrated that the AW-RBAC model enables supervision of all workflow changes and therefore increases security at change time.

VI. Implementation

In this section we will describe the integration of the AW-RBAC model and a selected adaptive WfS. This implementation section presents a work-in-progress, as we currently have integrated the AW-RBAC only in a subset of WfS components.

As shown in Fig. 1 by the colored circles, the security restrictions imposed by the AW-RBAC model have to be

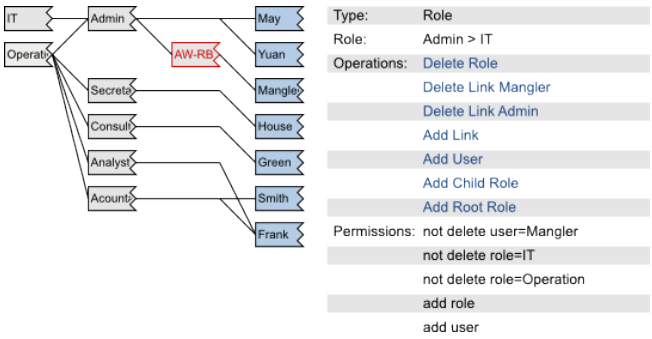

Figure 6. Deleting the Link to Self is NOT Allowed

enforced in a multitude of components throughout the WfS. The *Process Editor* has to ensure that only certain users are allowed to design or change the control or data flow for certain processes. The *Process Repository* can check if certain changes are applied by certain users, and thus refuse to save process models. The *Activity Repository*, which holds all activities that can be used to design processes, can refuse the usage of certain activities for certain users during the design process. Our HTML5 based *Process Editor* is well integrated with the AW-RBAC model, and allows for fine-grained restrictions regarding all modeling operations.

The *RBAC Model Editor* is special because it is used to add / delete roles, users and permissions. As not every user is allowed to do so, it has to enforce its own rules. In Fig. 6 our HTML5 based RBAC model editor is shown. In the screenshot, the model described in Sect. V is loaded. The currently logged in user (*Mangler*) has selected the *AW-RBAC* Admin role and tries to delete the link (and thus the user) *Mangler*. The editor refuses to carry out this operation, as the user would lock himself / herself out of the editor. The editor has the following properties: (a) It can load and save multiple model instances. As we currently integrate the AW-RBAC model with several independent installations of WfS's, we implemented a repository that can handle organizational models. When loading or saving from the repository the editor always includes the current user / role, so that the repository can check for security breaches. (b) It visualizes the users and roles, and their links. (c) It provides a set of operations for each entity. It is possible to add / delete links, users and roles. (d) The editor enforces the set of permissions connected with user / role that modifies the model. The permissions themselves cannot yet be edited in the UI (but have to be inserted by hand in the storage backend).

In general, a *monitoring* interface is very important. As it handles and restricts access to historic and real-time monitoring information (including progress of instances in respect to control and data flow) it can be used by independent monitors (which themselves do not have to enforce the AW-RBAC model) for various process related data mining tasks, without compromising security.

The **runtime** aspects (right hand side of Fig. 1) require even deeper integration with the AW-RBAC (not yet finished in our prototype implementation). The *Workflow Engine* holds a set of components (described in [12]) including the ability to select or replace concrete services for activities at runtime. The *Worklist* is a special service (represented by an activity in the process), that has to enforce the AW-RBAC model itself (i.e. without help of the *Workflow Engine*). It has to extract a list of potentially entitled users from RBAC repository in order to enforce access control.

VII. Related Work

One of the most well-established models to express security rules is the RBAC model which is commonly adopted to restrict authorization in workflow systems e.g. NIST RBAC model [13]. Many RBAC models and extensions have been proposed. For example, the ARBAC97 model provides a subset of administrative roles (disjoint from the other roles) which can perform administrative operations [5]. The SARBAC model family uses the concept of administrative scope for supervising the role hierarchy [14]. The UARBAC model presented in [6] introduces object classes and access modes in the RBAC schema to further define object and class permissions. All administrative RBAC approaches focus on the administration of user-role, role-permission, role-role, and object assignments. But none of them include the supervision of control flow, data or service changes of WfS.

Workflow related RBAC models have been developed in [15], [16], [7]. In the T-RBAC model, roles are associated with tasks and a task consists of a set of permissions [15]. W-RBAC enhances the RBAC model with cases (workflow instances) and a relation "Doer" between users and privileges (permissions) [16]. Furthermore, the W-RBAC model includes an entity "organizational unit". Although the T-RBAC and W-RBAC model include process related entities, they miss the administration of any workflow changes.

RBAC models for adaptive WfS are proposed by [7], [11]: they supervise authorized control flow changes. However, other workflow changes such as administrative, data flow, or service changes are not incorporated in these models.

Further approaches focus on permission-based security in UML models. For example, in [17], consistency checks between workflows and the design of the security permissions are supported to support early stages of system development.

Table V presents the results from evaluating existing RBAC models and the AW-RBAC model along the requirements for securing workflow change as set out in Sect. III). In summary, the AW-RBAC model is the only access control model that supports all workflow changes and requirements for adaptive WfS.

VIII. Conclusion and Future Work

In this paper, an extended RBAC model for adaptive Workflow Systems (AW-RBAC) has been presented. Adap-

Table V
Evaluation of existing RBAC models and AW-RBAC

RBAC Model	R1				R2	R3	R4
	A	B	C	D			
ARBAC97 [5]	+				+	+	+
SARBAC [14]	+				+	+	+
UARBAC [6]	+				+	+	+
NIST RBAC [13]					+	+	+
W-RBAC [16]					+	$\sim$	+
Domingos et al. [11]		+			+		+
Weber et al. [7]		+			$\sim$		+
AW-RBAC	+	+	+	+	+	+	+

tive Workflow Systems constitute an exciting new technology that meets enterprises' needs for flexibility. However, allowing arbitrary users of Workflow Systems any change is dangerous as we have shown based on different use cases. Hence the proposed AW-RBAC model is designed to enable the controlled application of changes in Workflow Systems. A particular challenge was to integrate all different kinds of workflow changes within one model. Specifically, changes of control and data flow, administrative changes of organizational structures, and changes at the service level of a workflow can be controlled based on the AW-RBAC model. Further on, the AW-RBAC model enables the fine-grained definition of access rights based on the separation of RBAC elements and constraints. The AW-RBAC model has been realized as security service that can be implemented by any Workflow System.

In future work we aim evaluating the AW-RBAC model based on further case studies. In addition, we will extend our considerations to advanced workflow scenarios such as cross-organizational settings and mobile applications. Overall, the AW-RBAC model facilitates the application of adaptive Workflow Systems in practice by providing means for secure change application in such systems.

ACKNOWLEDGEMENTS

The research was partially funded by COMET K1, FFG - Austrian Research Promotion Agency.

REFERENCES

[1] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[2] V. Atluri and W. Huang, "An authorization model for workflows," in *Computer Security - ESORICS 96*, 1996, pp. 44–64.

[3] P. C. K. Hung and K. Karlapalem, "A secure workflow model," in *Australasian information security workshop conference on ACSW frontiers 2003 - 21*, 2003, pp. 33–41.

[4] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features – enhancing flexibility in process-aware information systems." *Data and Knowledge Engineering*, vol. 66, no. 3, pp. 438–466, 2008.

[5] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 model for role-based administration of roles," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 1, pp. 105–135, 1999.

[6] N. Li and Z. Mao, "Administration in role-based access control," in *2nd ACM symposium on Information, computer and communications security*, 2007, pp. 127–138.

[7] B. Weber, M. Reichert, W. Wild, and S. Rinderle, "Balancing flexibility and security in adaptive process management systems," *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pp. 59–76, 2005.

[8] P. Dadam and M. Reichert, "The ADEPT project: a decade of research and development for robust and flexible process support," *Comp. Science - Research and Development*, 2009.

[9] N. Russell, W. M. van der Aalst, A. H. ter Hofstede, and D. Edmond, "Workflow resource patterns: Identification, representation and tool support," in *Proc. Int'l Conf. Advanced Information Systems Engineering*, 2005, pp. 216–232.

[10] S. Rinderle-Ma and M. Reichert, "Comprehensive life cycle support for access rules in information systems: The CEOSIS project," *Enterprise Inf. Syst.*, vol. 3, no. 3, pp. 219–251, 2009.

[11] D. Domingos, A. Rito-Silva, and P. Veiga, "Authorization and access control in adaptive workflows," in *Computer Security - ESORICS 2003*, 2003, pp. 23–38.

[12] G. Stuermer, J. Mangler, and E. Schikuta, "Building a modular service oriented workflow engine," in *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, 2010, p. 1–4.

[13] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *Proc. of the fifth ACM workshop on Role-based access control*, 2000, pp. 47–63.

[14] J. Crampton, "Understanding and developing role-based administrative models," in *Proc. of the 12th ACM Conference on Computer and communications security*, 2005, pp. 158–167.

[15] S. Oh and S. Park, "Task-role-based access control model," *Information Systems*, vol. 28, no. 6, pp. 533–562, 2003.

[16] J. Wainer, P. Barthelmess, and A. Kumar, "W-RBAC - a workflow security model incorporating controlled overriding of constraints," *International Journal of Cooperative Information Systems*, vol. 12, no. 4, pp. 455–485, 2003.

[17] J. Jürjens, M. Lehrhuber, and G. Wimmel, "Model-based design and analysis of permission-based security," in *Engineering of Complex Computer Systems, 2005. ICECCS 2005. Proc. 10th IEEE Int'l Conf. on*, 2005, pp. 224–233.