Gang MA, Kehe WU,Tong ZHANG, Wei LI

North China Electric Power University

A Flexible Policy-Based Access Control Model for Workflow

***Abstract**. In recent years, more and more attentions have been paid on the security of Workflow Management Systems (WFMSs) for its importance both in research and commercial realms. Access control is crucial to security management in WFMSs. A novel dynamic policy-based access control model for WFMSs called PAWF is presented to satisfy the particular requirements of WFMSs. The proposed model supports fine-grained authorization based on authorization policies and a prototype system is developed to prove the effectiveness of the proposed access control model.*

***Streszczenie.** Zaprezentowano nowy model dostępu do sieci PWWF przy przepływie informacji. (**Elastyczny parametryzowany wytycznymi model dostępu do sieci przy przepływie informacji**)*

Keywords: Policy-based access control; WFMSs; Security constrains; Separation of duty; Authorization management.
Słowa kluczowe: przepływ informacji, dostęp do sieci.

I.Introduction

The workflow is an automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [1]. WFMSs make it possible that a workflow could be analyzed, simulated, designed, enact, controlled and monitored [2]. Meanwhile, more and more attentions have been paid on the security of workflows. In the security white paper published by WFMC, the security of the workflows are boiled down to authentication, authorization, access control, audit, data privacy, data integrity, non repudiation, security management and administration [3].

Access control is one of the important technologies which could ensure the safety of information systems. In the context of workflow systems, it may operate at the level of: (a) log-on to the workflow service, and (b) access to undertake particular activities or work items according to functional role and/or data sensitivity [3]. Many researchers have developed access control models, such as Role-based Access Control (RBAC) [4], Task-based Access Control (TBAC) [5] and Task-role-based Access Control (T-RBAC), in addition to Discretionary Access Control (DAC) and Mandatory Access Control (MAC).

In this paper, we summarize the requirements of access control in enterprise WFMSs in four aspects. Then, we propose our novel model called PAWF to satisfy such particular requirements in WFMSs. We adopt the advantages of RBAC and TBAC and introduce the dynamic authorization policies into it. We develop a prototype system of PBFM. The case study shows that PAWF meets the dynamic and flexible requirements, such as SoD, least privilege constraints, dynamic access control, and so on. PAWF surpasses RBAC, TBAC, PBAC and T-RBAC in evaluation.

The reminder of this paper is organized as follows. The related work is introduced in section II. Section III analyses the dynamic and flexible requirements of access control in the enterprise WFMSs and summarize the requirements in four aspects, which are the goals our model must achieve. The PAWF model is proposed in Section IV. Section V shows the implementation of PAWF model and gives promising results in case study. Conclusions and future work are presented in Section VI.

II. Related work

The security of information is a vital theme in workflows, while the access control is an important part of it. Researchers have made their every possible effort to built proper access control solutions for workflows. As a result, many access control models have been proposed. In this section, we briefly describe related work on access control. According to the authorization policy, access control could be divided into four categories, which are traditional access control, RBAC, TBAC and PBAC[6,7].

A.Traditional Access Control

Traditional access control contains DAC[8] and MAC[9]. However, DAC allows the object's owner assign the administration authority to others, which brings flexibility as well as security risk. And access control list (ACL) will be hard to maintain when the number of users is huge. In the MAC model, all subjects and objects are classified based on predefined sensitivity levels which is difficult in workflows and is not suitable for tasks execution. MAC is mainly used for the military systems with the high and multi-level security.

B. RBAC

RBAC is a passive and policy-neutral authorization model. RBAC gives access rights to roles instead of users, which can prevent users from accessing information objects at their discretion. Although the RBAC has many advantages, its disadvantages are easily found. RBAC could not cover efficiently in workflows[12]. For instance,

- RBAC is one of the passive access control models, and it does not support active access control.
- The role hierarchy concept where a higher role inherits all authorities of a lower role in the role hierarchy is a full inheritance of access rights, which cannot support the principle of least privilege in workflows.
- RBAC does not separate 'task' from 'role'[5].

C. TBAC

TBAC model is one of the active access control security model based on the tasks and dynamic authorization. It has the central idea of task-oriented in access control. Moreover, TBAC separates access right assignment for users and access right activation in the process of handling the tasks [6]. Although there are many advantages, the TBAC model is still has some limitations in the workflows.

- In the workflow, both passive and active access control model are needed, while the TBAC model only support the active one.
- The role hierarchy concept is not considered specially.

D. PBAC

PBAC is widely used in the enterprise information systems. It controls user access resources by pre-defined policies and rules. The flexibility and diversity of PBAC could adapt to the active requirements of workflows [7].

The model contains five parts: Rule, Policy, Resource, Association and Session Attributes. Rule is the core of the PBAC. In the enterprise workflow environment, the number of users and resources are huge, and it is hard to develop and manage rules. As such, the extensibility of PBAC is weak, which is the limitation of PBAC in workflows.

E. T-RBAC

T-RBAC is an improved access control model based on the RBAC model and contains the basic features of RBAC.

In this model, a user has a relationship with permission through role and task[5]. Although it has many advantages as shown above, there are also some limitations but important research in the workflows.

• It is hard to make classification of tasks. There are too many tasks to classify into suitable class as definition in enterprise environment.

• It could not deal with some business rules, such as delegation and closing account, which are common and important for enterprise environment to support efficient execution of business activities.

• It is not suitable for distributed enterprise environment. T-RBAC is centralized and it does not consider local and global access control integration and their communication in a distributed enterprise environment.

III.Requirements analysis

It is very important for us to know the requirements of access control in WFMSs, before we develop a proper access control model for workflows. Generally speaking, workflows mainly consist of users, operations and objects. By observation and analysis of the access control characteristics of the workflows in enterprise environment, we have found some special requirements of workflows:

• Keep authorization and task execution in synchronization. Workflow is a sequence of tasks. So, giving users permission should synchronize with the implementation of the workflow.

• Least Privilege Principle. User can only work with the minimum necessary set of permissions of a task when he is performing the task, and the set of permissions will be withdrawn as soon as the task is finished.

• Separation of Duty. Both static separation of duty relations (SSD) and dynamic separation of duty relations (DSD) need to be enforced to prevent information being misused and prevent fraudulent activities.

• Static and dynamic authorization. A workflow involving many participants, which are different in method of authorization. It needs static access control mechanisms to support passive authorization. Furthermore, the dynamic access control will help to support active authorization in the process of a workflow.

Many advanced access control models for workflows are presented, but they do not completely satisfy the dynamic and flexible requirements of workflows[15]. Our access control model proposed in this paper will deal with the four requirements, but not limited to. Although some of the requirements are general and well known, we give them new content and research methods.

IV.PAWF Model

PAWF is based on the RBAC and TBAC model, and therefore PAWF contains the basic features of them. It makes the policy separate from the task, and a task can be constrained by several policies. It supports both static constrains and dynamic constraints simultaneously.

A.Core PAWF

Fig. 1 shows a brief overview of PAWF. In this model, permissions are assigned to the task instances, and task instances are assigned to roles. Access rights are associated with roles and task instances. What's more, assignments of roles between task instances are restricted by authorization policies passively or actively. The definitions and properties below show the formal description of PAWF model. In our model, we introduce some concepts of basic elements in RBAC, TBAC and PBAC,such as *user*, *role* and so on.

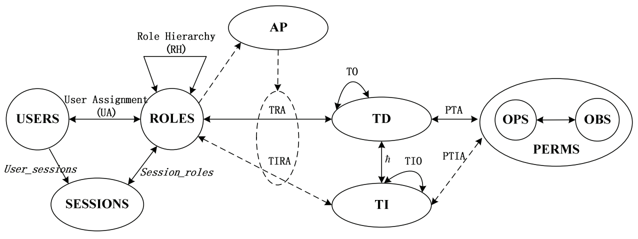


Fig.1. The PAWF model

Property 1 Task and task instance

✧ $\hbar$ (*td*: *TD*) → 2^{TI} , a one-to-many mapping task_definition-to-task_instances instance relation. $\exists\, td_i, td_j \in TD$, $\hbar(td_i) \cap \hbar(td_j) = \Phi \Leftrightarrow td_i \neq td_j$.

✧ *deft(ti*: *TI)* → $td \wedge td \in TD$, the one-to-one mapping of task_ instance-to-task_definition, which gives the task_difinition *t* of the task_instance *ti*. Formally: $deft(ti) = t \wedge ti \in \hbar(t)$.

Definition 1 Workflow_definition and workflow_instance

✧ *Workflow_definition*: a set of task definitions and the relations among them that could form a net. [9]

✧ *Workflow_instance*: we can get a *WI* when the *WD* executions once. Formally:

✧ $WD = \{wd_1, wd_2, \dots, wd_n\}$, the set of all workflow_definitions in system.

✧ $WI = \{wi_1, wi_2, \dots, wi_j \dots, wi_n\} \wedge \{\forall\, ti_j \in wi_j \mid deft(ti_j) \in wd_j\}$, the set of all *workflow_instances* in system.

Definition 2 Workflow_history(*WH*)

✧ *Workflow_history* is a set of information which is a part of the task_instance, written as *WH*. $\forall\, wh_i \in WH \Rightarrow wh_i \subseteq ti \wedge \exists\, ti \in TI$. Formlly: $wh_i = (wi, td, u, r)$ where:

• $wi \in WI \wedge ti \in wi$, *wi* gives the *workflow_instance* which contains *ti*.

• $td = deft(ti)$.

• *u* is the user who executes *ti*.

• *r* is the role assigned to *u* when *u* is executing *ti*.

Definition 3 Authorization policy (AP)

✧ *AP* is a set of authorization policies.

✧ A policy *p* assigned to task *t* is formally defined as a six-tuple *p* = (*wd*, *ob*, *index*, *rule*, *sign* , *actor*), where:

• *wd* is the *workflow_definition* that *t* belongs to.

• $ob = 2^{TD} \vee 2^{TID}$ is a *task_definition /task_instance which* need to be protected.

• *index* is number of the policy. A task can has one or more policies.

• *rule* is the regular expression of constraints, formally: rule: *p* → *q* , where *p* and *q* are both authority relationships.

• *sign* is a mark of positive or negative authorization. In our model, we use 0 for positive authorization while 1 for negative.

• $actor = 2^{USERS} \cup 2^{ROLES}$ is a set of users/roles or both which will be constrained by *p*.

Property 2 Authority relationship

✧ $positive_u(o, u) \Leftrightarrow r \in assigned\_roles(u) \wedge ((o, r) \in TRA \vee (o, r) \in TIRA)$, where $positive_u(o, u)$ only if user *u* can deal with object *o*.

✧ $positive_u(o) = 2^u \wedge positive_u(o, u)$, the set of *users* who deals with *o*.

✧ $positive_r(o, r) \Leftrightarrow (o, r) \in TRA \vee (o, r) \in TIRA$, where $positive_r(o, r)$ only if user who is assigned the role *r* can deal with object *o*.

✧ $positive_r(o) = 2^r \wedge positive_r(o, r)$, the set of roles who deals with *o*.

✧ $negative_u(o, u) \Leftrightarrow r \in assigned\_roles(u) \wedge ((o, r) \notin TRA \wedge (o, r) \notin TIRA)$, where $negative_u(o, u)$ only if user u can not deal with object o.

✧ $negative_r(o, r) \Leftrightarrow (o, r) \notin TRA \wedge (o, r) \notin TIRA$, where $negative_r(o, r)$ only if role r do not has the rights to deal with object o.

✧ $PTR \subseteq \bigcup_{\alpha \in 2^{TD} \times 2^{ROLES}} \{ \alpha \mid positive_r(\alpha) \}$

✧ $PTIR \subseteq \bigcup_{\beta \in 2^{TID} \times 2^{ROLES}} \{ \beta \mid positive_r(\beta) \}$

✧ $NTR \subseteq \bigcup_{\alpha \in 2^{TD} \times 2^{ROLES}} \{ \alpha \mid negative_r(\alpha) \}$

✧ $NTIR \subseteq \bigcup_{\beta \in 2^{TID} \times 2^{ROLES}} \{ \beta \mid negative_r(\beta) \}$

Property 3 Task_definition-role assignment (TRA)

✧ $TRA \subseteq TD \times ROLES - NTR + PTR$ is a many-to-many mapping which gives the relations of task_definition-to-role.

Property 4 Permission-task_definition assignment (PTA)

✧ $PTA \subseteq PERMS \times TD$ is a many-to-many mapping permission-to-task_definition assignment relation.

Property 5 Task_instance-role assignment (TIRA)

✧ $TIRA \subseteq TI \times ROLES - NITR + PTIR$ is a many-to-many mapping which gives the relations of task_instance-to-role.

Property 6 Permission-task_instance assignment (PTIA)

✧ $PTIA \subseteq \bigcup_{p \in PERMS,\ ti \in TI} \{ (p, ti) \mid (p, deft(ti)) \in PTA \}$ is a many-to-many mapping permission-to-task_istance assignment relation.

Property 7 Task_definiton order (TO) and task_istance order (TIO)

✧ TO is the order of the tasks in a work_definition when they are executing, written as $\mapsto$. If t_1, t_2, …, t_n are prior tasks of t_i, written as $\{ t_1, t_2, \ldots, t_n \} \mapsto t_i$, where t_i is activated now only if t_1, t_2, …, t_n are completed.

✧ *TIO* is the order of the task_instances in a work_instances. Generally, *TIO* is a copy of *TO*.

Property 8 Task_instance-role assignment (TIRA)

✧ $TIRA \subseteq TI \times ROLES - NITR + PTIR$ is a many-to-many mapping which gives the relations of task_instance-to-role.

.B. Authorization policies

In the PAWF model, the authorization policy is an important part. According to our previous definition, a policy p = (*wd*, *object*, *index*, *rule*, *sign*, *actor*). The *rule* is the critical factor. In order to give the formal description of *rule*, we give some definitions firstly.

Definition 4 Authorization policy (*AP*)

We give a task_definition $td \in TD$, and

✧ $pr(td\colon TD) \rightarrow \{ r \mid \exists (td, r') \in TRA \wedge r \succ r',\ r, r' \in ROLES \}$ gives the set of roles which are the parent of the *td*'s initial roles.

✧ $ur(td\colon TD) \rightarrow \{ u \mid \exists (u, r) \in UA \wedge r \in pr(t),\ u \in USERS,\ r \in ROLES \}$ gives the set of users which are assigned the roles in $pr(td)$.

We use notation $\triangleright$ to describe a rule. $rule = p \triangleright q$, where p and q are both authority relationships and authority of q depends on the elements in p. In addition, rule is a static constraint when p is *null* otherwise, it is dynamic.

We give an example of rules in this paper: $rule = positive_r(t_1, r_1) \triangleright positive_r(t_2, r) \wedge r \in pr(t_1)$, $t_1 \mapsto t_2$, which means that roles r who execute t_2 should be parent of r_1.

Authorization policy can be generated in *TRA* or *TIRA* when the workflow is defined. In order to keep authorization and task execution in synchronization, the policies of posterior task_nstance could be generated by the prior while the workflow_definition is executing if the prior is authorized to do that. It makes PAWF flexible and scalable.

C.Separation of duty relations

Separation of duty relations is a common approach to enforce conflict of interest policies in a workflow that may cause information misuse and fraudulent activities. Conflict of interest in PAWF model may arise because of a user gained an authorization for permissions associated with conflicting roles. As a security principle, SoD has long been recognized for its wide application in business, industry, and government [10,11].

PAWF allows for both SSD and DSD. SSD implements mainly in the definition phase of workflows. It is mainly used in *UAI*, *RH* and *TRA*. DSD works while the workflow_definition is executing. It is a necessary complement to SoD. In PAWF, we pay main attentions to DSD. In this paper, we use the concept of Conflicting Entities to express SoD principles.

Definition 5 Conflicting Entities

✧ $CP \subseteq PERMS \times PREMS$ is a set of conflicting permissions. Where $p_i, p_j \in PERMS$, $(p_i, p_j) \in CP$, if p_i and p_j both are assigned to user u and u will gain enough authorities to do fraudulent operations.

✧ $CU \subseteq USERS \times USERS$ is a set of conflicting users. Where $u_i, u_j \in USERS$, only if u_i and u_j have the possibility to finagle together in a workflow, then $(u_i, u_j) \in CU$.

✧ $CT \subseteq TD \times TD$ is a set of conflicting tasks_definitions, if the permissions to deal with td_i and td_j are conflicting, $(td_i, td_j) \in CT$.

✧ $CR \subseteq ROLES \times ROLES$ is a set of conflicting roles, if r_i and r_j have conflicting tasks_definitions, then $(r_i, r_j) \in CR$.

Conflicting Entities take new impacts to *RH*, so we should give new properties of it to describe the improvements, as shown in Fig. 2 [13].

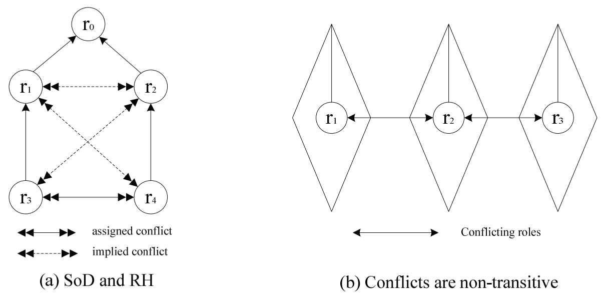


Fig.2. Conflicting entities and *RH*

Property 9 Conflicting entities and *RH*

✧ r_i and r_j can not have the same parent role if r_i and r_j are conflicting roles, which $(r_i, r_j) \in CR \Leftrightarrow \neg\exists\, r_x \in ROLES \wedge (r_x \succ r_i) \wedge (r_x \succ r_j)$.

As shown in Fig. 2(a). r_3 and r_4 are specified conflicting, so implied conflict exists between r_1 and r_4, r_1 and r_2, and r_2 and r_3. What's more, r_0 inherit the permissions associated with r_3 and r_4, which clearly gives unwanted power to r_0.

✧ Conflicts are non-transitive

Conflicts are not transitive. They are only implied by the relationships in the role hierarchies. As shown in Fig. 2(b), r_1, r_2 and r_3 belong to three different and unrelated role hierarchies. Though r1 and r_2, r_2 and r_3 are conflicting, r_1 and r_3 are not conflicting.

The role assigned to current tasks depends on the history of roles assigned in a workflow task instance. With the concept of conflicting entities we could describe *SoD* conveniently in PAWF. By adding *SoD* features, we give following three authorization principles.

Principle 1 Conflicting roles should not be assigned to one user in a workflow_instance,which $(r_i, r_j) \in CR \wedge (wi, td_i, u_i, r_i) \in WH \wedge (wi, td_j, u_j, r_j) \in WH \Rightarrow u_i \neq u_j \vee u_i \notin \{u' \mid (u', u_j) \in CU\}$.

Principle 2 Conflicting permissions should not be assigned to one task_instance in a workflow_instance, which $(p_i, p_j) \in CP \wedge (wi, td_i, u_i, r_i) \in WH \wedge (p_i, td_i) \in PTA \wedge (wi, td_j, u_j, r_j) \in WH \wedge (p_j, td_j) \in PTA \Rightarrow td_i \neq td_j \vee td_i \notin \{td' \mid (td', td_j) \in CT\}$.

Principle 3 Conflicting tasks_definitions should not be assigned to one role in a workflow_instance, which $(td_i, td_j) \in CT \wedge (wi, td_i, u_i, r_i) \in WH \wedge (wi, td_j, u_j, r_j) \in WH \Rightarrow r_i \neq r_j \vee r_i \notin \{r' \mid (r', r_j) \in CR\}$.

We can use table 1 to show summary of the assignment relations between conflicting entities proposed by the previous three principles.

Table 1 assignment relations between conflicting entities (co: conflicting, nc: non-conflicting)

Can be assigned?	*USERS*		*ROLES*		*TD*		*PERMS*	
	co	nc	co	nc	co	nc	co	nc
CU	-	-	N	Y	$\underline{N}$	$\underline{Y}$	$\underline{N}$	$\underline{Y}$
CR	N	Y	-	-	N	Y	$\underline{Y}$	$\underline{N}$
CT	$\underline{N}$	$\underline{Y}$	Y	N	-	-	Y	N
CP	$\underline{N}$	$\underline{Y}$	$\underline{Y}$	$\underline{N}$	Y	N	-	-

As shown in table 1, the implied assignment relations between conflicting entities are expressed by the conclusion with underline, which could be easily deduced from the previous three principles.

Definition 6 Proxy

✧ *proxy(u1, u2)* is an authorization relationship which means user *u1* gives all his permissions to user *u2*.

proxy(u1, u2) means that use u2 have all the permissions which are assigned to *u1* and can do anything *u1* can do. Practically, it is constrained by authorization policies.

For the sake of brevity, we simplify *SSD* and bring the main features of RBAC model into PAWF model. Combined with the actual requires of PAWF, we pay main attentions to *DSD*. With the benefit of concept of *Conflicting Entities*, we propose three authorization principles to support *DSD*.

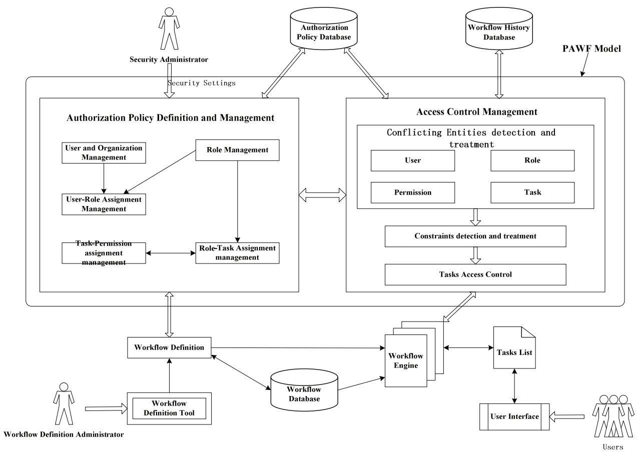

Security Administrator
Authorization Policy Database
Workflow History Database
PAWF Model
Security Settings
Authorization Policy Definition and Management
User and Organization Management
Role Management
User-Role Assignment Management
Task-Permission assignment management
Role-Task Assignment management
Access Control Management
Conflicting Entities detection and treatment
User
Role
Permission
Task
Constraints detection and treatment
Tasks Access Control
Workflow Definition
Workflow Engine
Tasks List
Workflow Database
Workflow Definition Tool
User Interface
Workflow Definition Administrator
Users

Fig.3. The prototype system based on PAWF model

V. Evaluation

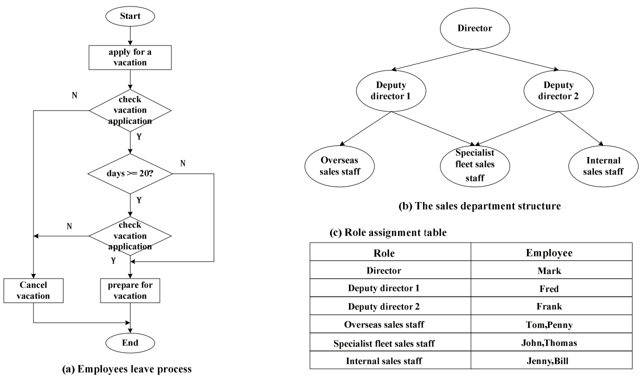
A.Implementation of PAWF Model

We have developed a prototype system based on PAWF model with Java language. This prototype system runs in workflow environment and provides access control services for WFMSs to ensure that the legal and authorized use can access WFMSs and do proper operations. It is a security interface between users and WFMSs.

As shown in Fig. 3, this prototype system consists of Authorization Policy Definition and Management and Access Control Management. Access control services provided by the prototype system contain two phases: workflow design phase and workflow execution phase. In workflow design phase, the prototype system manages Users, Roles, Tasks and relationships among them which are the input data of the workflow execution phase. In workflow execution phase, the prototype system handles the constraint relations between conflict entities and does authorization.

B.Case Study

In this section, we illustrate the application of the model with an enterprise employee leave process named *w*, as shown in Fig. 4 (a) contains five tasks.


Start
apply for a vacation
N
check vacation application
Y
days >= 20?
N
Y
N
check vacation application
Y
Cancel vacation
prepare for vacation
End
(a) Employees leave process
Director
Deputy director 1
Deputy director 2
Overseas sales staff
Specialist fleet sales staff
Internal sales staff
(b) The sales department structure
(c) Role assignment table

Role	Employee
Director	Mark
Deputy director 1	Fred
Deputy director 2	Frank
Overseas sales staff	Tom,Penny
Specialist fleet sales staff	John,Thomas
Internal sales staff	Jenny,Bill

Fig.4. An example of enterprise environment

- T1: Staffs apply for a vacation.
- T2: The leader who in charge of applicant (as shown in Fig. 4. (b)) checks the vacation application.
- T3: The Director checks the application.
- T4: The applicant prepares for have a vacation.
- T5: The applicant cancels the vacation.

We illustrate properties of the roles, the role hierarchy, users and the assignment of the process in Fig. 4 (b) (c). As shown, Fred is assigned to deputy director 1 and in charge of overseas sales staff and specialist sleet sales staff. His permissions are the union of the permission sets assigned to them. The other users, roles and the assignment are also shown in the role graph.

According to the enterprise business security requirements, the processes should meet the following constraints:

- C1: T1 can not be handled by the director.
- C2: T2 should be handled by the leader who is in charge of applicant (as shown in Fig. 4. (b)) .
- C3: If a deputy director makes an application, T2 should be handled by the other deputy director.
- C4: T3 can only be executed by the director or his proxy.
- C5: T1, T2, T3 should not be handled by a same person.
- C6: T1, T4 and T5 should handled by a same applicant.

According to the categories of security constrain, we can clearly see that C1, C4 and C6 are static constrains, C3, C5 is dynamic constrains, C2 is a mixed constrain. In PAWF model, we make authorization policies for each tasks in a workflow to meet security constrains. In the enterprise employee leave process, we define authorization policies for each task as follow.

Table 2 Authorization policies of each task

Tasks	Authorization policies
T1	P1:(*w*, T1, 1, rule, 0, *r*)
T2	P2:(*w*, T1, 2, rule, 0, *r*), P3:(*w*, T2, 1, rule, 0, *u*), P4:(*w*, T2, 2, rule, 1, *u*)
T3	P5:(*w*, T3, 1, rule, 0, *r*), P6:(*w*, T2, 2, rule, 1, *u*)
T4	P7:(*w*, T4, 1, rule, 0, *r*)
T5	P8:(*w*, T5, 1, rule, 0, *r*)

In order to meet the needs of a flexible authorization, we generate authorization policies of each task as shown in table 2 according the constraints which the process should follow. As defined in **Definition 3**, it is a one-to-many mapping between tasks and authorization policies and the rules are the core of policies. Rules of each policy are described as follows.

- P1.rule = $null \triangleright negative_u$(T1,(Director)).

Description: The Director can not apply for a vacation in T1. There is no authorization policy P1 should depend on and P1 is a static constrain.

- P2.rule = $positive_r$(T1, r) $\triangleright$ $positive_r$(T2, r1) $\wedge$ r1 $\in$ *pr*(T1) .

Description: The user who executes T2 should be leadership of the applicant in T1.

- P3.rule = $positive_r$(T1, r) $\wedge$ r $\in$ {deputy director1, deputy director2} $\triangleright$ $positive_r$(T2, r1) $\wedge$ r1 $\in$ {deputy director1, deputy director2} - r .

Description: If the role of the applicant in T1 is deputy director, then the other deputy director should execute T2.

- P4.rule = u $\in$ $positive_u$(T1) $\triangleright$ $negative_u$ (*T2*, *u*)

Description: This is a dynamic constrain which prevents one user dealing with T1, T2 at the same time.

- P5.rule = *null* $\triangleright$ $positive_r$(T3, Director) $\vee$ ($positive_u$(T3, u) $\wedge$ u $\in$ *proxy*(Director, u)).

Description: Only the Director can execute T3.

- P6.rule = $positive_r$(T1,(Production manager, Material manager)) $\triangleright$ $positive_r$(T3, r) $\wedge$ r $\in$ pr(T1).

Description: The role which deals with T3 should be parent of the role that deals with T1.

- P7.rule=u1 $\in$ $positive_u$(T1) $\wedge$ u1 $\in$ *USERS* $\triangleright$ $positive_u$(T4, u1) $\wedge$ ($negative_u$(T4, u) $\wedge$ u $\in$ *USERS*-{u1}).

Description: Only the applicant in T1 can deal with task T4 in a same workflow instance.

- P8.rule=u1 $\in$ $positive_u$(T1) $\wedge$ u1 $\in$ *USERS* $\triangleright$ $positive_u$(T5, u1) $\wedge$ ($negative_u$(T5, u) $\wedge$ u $\in$ *USERS*-{u1}).

Description: Only the applicant in T1 can deal with task T5 in a same workflow instance.

We give a scenario to simulate *w* according to above authorization policies and compare there models: Tom wants to apply a 25 days vacation, and Fred delegated his work to Frank. Tom makes an application in T1, and then the application will be send to deputy Fred. But Fred delegated his work to Frank, so Frank will check the application in T2. If Frank says no to the application, Tom should cancel the vacation in T5. Then the process ends. If Frank agrees and the vacation days are more than 19 days, the application will be sent to the Mark to approval. If Mark agrees the vacation in T3, Tom then can prepare for have a vacation in T4. Otherwise, he should cancel the vacation in T5. From the process, we can clearly see that our authorization policy can express the three categories of security constrains easily and can strongly enforce static and dynamic authorizations. The comparison result of supporting the case between access control models is shown in Table 3.

Table 3 Comparison of supporting the case

No.	Constraints	RBAC	TBAC	PBAC	T-RBAC	PAWF
1	C1	√	√	√	√	√
2	C2	√	√	×	√	√
3	C3	×	×	×	×	√
4	C4	×	×	×	×	√
5	C5	√	√	√	√	√
6	C6	√	√	√	√	√

VI.Conclusions and Future Work

In this paper, we propose an improved model named PAWF to satisfy the particular requirements of workflows in enterprise environment. It is based on the characteristics of tasks and policies. This access control model meets both 'flexibility' and 'scalability'. PAWF model achieves dynamic authorizations with task instances and with the help of conflicting entities to express SoD principles.

Although PAWF has many advantages, there remain some important research topics. We need further research on time constraints for authorization policies, cooperating with other security control mechanisms, optimizing policy descriptions.

This work is partially supported by Beijing Engineering Research Center of Electric Information Technology. We would like to thank Ph.D. Hui He and anonymous reviewers for their constructive comments that greatly improved the quality of this paper.

REFERENCES

[1] WFMC. Workflow Management Coalition: Terminology & Glossary. WFMC-TC-1011,Issue 3.0,1999

[2] WFMC. Workflow management coalition: workflow reference model. WFMC-TC00-1003,Issue 1.1,1995

[3] WFMC. Workflow management coalition: Workflow Security Considerations-White Paper. WFMC-TC-1019,Issue 1.0,2001

[4] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli. Proposed NIST Standard for Role-Based Access Control, *ACM Trans. Information and System Security*, 4(2001), No. 3, 224-274

[5] Sejong Oh, Seog, Park. Task-role-based Access Control Model, Information System, 28(2003): 533 -562

[6] Xie Dongwen, Liu Min,Wu Cheng. Policy-based access control in enterprise information system, *In Computer Integrated Manufacturing Systems,*11(2005) ,No 4,561-564

[7] Lin Zhi, Wang Jing, Chen Xiao-su, Jia Lian-xing. Research on Policy-based Access Control Model, *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2009,164-167,

[8] N. Li, M. V. Tripunitara. On safety in discretionary access control, *IEEE Security and Privacy*,2005.

[9] Claude Girault, Rudiger Valk. Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications, *Springer-Verlag New York*, 2001

[10]M Nyanchama, S Osborn. The role-graph model and conflict of interest, *ACM Transactions on Information and System Security*, 2(1999), No.1,3-33

[11]Reinhardt A. Botha, Jan H. P. Eloff. Separation of duties for access control enforcement in workflow environments, *IBM Systems Journal*, 40(2001), No.3, 666--682

[12]I. Ray and M. Toahchoodee. A spatio-temporal role-based access control model, *Proceedings of the 21th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2007,211--226

[13] Reinhardt A. Botha. CoSAWoE - A Model for Context-sensitive Access Control in Workflow Environments [D]:[PhD thesis]. South Africa: Johannesburg, 2001

Authors

Ph.D. Gang Ma, School of Electrical & Electronic Engineering, North China Electric Power University, E-mail: epumagang@ncepu.edu.cn; Prof. Kehe Wu, Deputy Dean of Control and Computing Engineering School, North China Electric Power University, E-mail:epuwkh@126.com; Ph.D. Tong Zhang, Control & Computing Engineering School of North China Electric Power University, E-mail:zhtzhangtong@163.com; Associate Prof. Wei Li, Control and Computing Engineering School of North China Electric Power University, E-mail:liwei@ncepu.edu.cn.