Dynamic Access Control through Petri Net Workflows

Konstantin Knorr
Department of Information Technology
University of Zurich
knorr@ifi.unizh.ch

Abstract

Access control is an important protection mechanism for information systems. An access control matrix grants subjects privileges to objects. Today, access control matrices are static, they rarely change during time. This paper shows how to make access control matrices dynamic by means of workflows. Access rights are granted according to the state of the workflow. By this practice the risk of data misuse is decreased which is proven through an equation given in the paper. The concept of workflow is defined by Petri nets which offer a solid mathematical foundation and are well suited to represent discrete models like workflows.
Keywords*: access control, workflow, Petri net*

1. Introduction

In information systems access control is a very important task. Legitimate users should be allowed to access data items, illegitimate users should be detained from data access. Access control matrices are a means to enforce these restrictions. An access control matrix grants subjects access rights to objects. This approach can be expanded by workflows. If the current state of a workflow is taken into account in the granting of access rights, the 'classical' access control matrix is extended from two (subject, object) to three dimensions (subject, object, state of the workflow). Access rights are now dependent on the context of the workflow. This concept is called dynamic access control.

By doing this, the execution of the workflow gets more secure, because the possibilities for unauthorized data access and data misuse, e.g. the unauthorized reading of a file, is reduced. Regarding the security services

- identification/authentication
- authorization
- confidentiality
- integrity
- non-denial/non-repudiation

(as defined in the ISO standard 7498-2) dynamic access control is a special mechanism providing for authorization. The focus of the proposed mechanism is on confidentiality, but integrity of the data items and and non-repudiation is also supported because only authorized subjects are allowed to operate with the data in the system. Furthermore, the access control mechanism realizes the need-to-know-paradigm because subjects will only be assigned with privileges if these privileges are needed for the execution of an activity in the workflow.

A specification of a workflow can be done by a net. Petri nets have been studied intensively in computer science. The description of workflows by Petri nets offers the following advantages:

- Petri nets are well suited to represent discrete dynamic models. A workflow combined with its execution is such a model because the points in time of the execution of the activities form a discrete set in comparison to the time axis which is continuous.
- A Petri net has a solid mathematical definition, its syntax and semantics are precisely defined.
- If a workflow is mapped onto a Petri net, certain properties such as the reachability of an end marking is mathematically provable. There is a large number of analysis methods which deal with the verification of Petri nets.

For these reasons this paper uses Petri nets for the specification of workflows.

The remainder of this paper is organized as follows. Section 2 discusses work related to the topic of this paper. Section 3 gives an introduction to workflow management and discusses the content of a workflow specification. The basic definitions of Petri nets are given in Section 4. Section 5 deals with the specification of workflows by Petri nets. The major contribution of this paper follows in Section 6. The relation between two and three dimensional access control

matrices is given. A sample workflow from an insurance company illustrates the practical implications of the proposed mechanism in Section 7. Section 8 discusses the findings and mentions future research issues.

2. Related Work

Work related to the specification of workflows through Petri nets can be found in [1, 17, 24, 25]. The basic idea is to associate transitions in Petri nets with activities in workflows.

The concept of the dynamic access control goes back to [21, 22, 23]. The authors define *task based access control* as an extension of *role based access control*.

Holbein et al. [10, 11] suggest to grant access rights according to the context of a business process, an approach which they call context-dependent access control. This is done through the use of a commercial workflow management system.

A publication about security aspects of workflows specified by Petri nets is [2]. An authorization model for workflows is introduced. Focus is laid on the synchronization of the workflow and the authorization flow. This is done by assigning time slots to the tasks in the workflow using a combination of colored and timed Petri nets which makes the model highly complex. The privileges to objects have to be defined as an essential part of the workflow specification.

Bertino et al. [3, 4] focus on the specification of authorization constraints in workflow management systems like separation of duties. The authors identify various types of constraints, a language to express these constraints, and an algorithm checking the consistency of the model.

Cholewka et al. [6] introduce a context-sensitive access control model and shows its feasibility with a prototype. Focus is laid on access control requirements like the sequence of activities, strict least privilege, and separation of duties, but not on access control on the data layer.

Harn and Lin [9] and Yen and Laih [27] also use the expression dynamic access control but in a different context. In most computer systems the user authentication and the actual access control is separated. Obviously, the more frequently authentication is performed, the better is the system's security. By enforcing user authentication every time a resource is accessed, a dynamic access control scheme is developed which is based on public key cryptography.

3. Workflows

Workflow management is an essential research area in the field of applied computer science. A *workflow management system* (WfMS) is a software system which supports the administration, modeling, and execution of workflows. A workflow is an executable business process. Before a workflow can be executed it has to be described in a way, the workflow management system is able to understand. This description is called *workflow specification*. The specification has to be done before a workflow can be executed. The most important part of a WfMS is the *workflow engine* which is responsible for the execution of the workflow during run time of the system when many instances of a workflow are created according to the workflow specification [8, 12, 26].

The main elements of a workflow specification are:

- activities
- control flow
- subjects
- data items
- data flow

The basic building blocks of a workflow are the activities whose temporal and logical order is given by the control flow. To describe an activity, it has to be specified which subjects are allowed to execute an activity and which data items are needed for and created during the execution.

Subjects can be associated with persons, but also groups of persons, roles, machines, and computer programs are possible subjects. In practice, the concept of *role* is very popular [14]. The execution of an activity is bound to a specific role and an employee of a company can activate one or more roles. For the reminder of the paper we associate subjects with persons or the workflow engine. In the case study in Section 7 there will be a special role for each activity; subjects are associated with roles.

Examples of data items are database entries and files. A subject executes an activity by creating new and/or using already existing data items. The data flow states how the data items move between the different activities [15].

In the remainder of this paper T will denote the set of the activities, S the set of the subjects, and D the set of data items in a workflow. The control flow will be given through a Petri net, the data flow and the assignment of subjects to activities through attributes of activities (cf. Section 5).

WfMSs are especially useful for *electronic* workflows. An electronic workflow is a workflow whose data items are stored in an electronic form. In this case the WfMS can forward the data items to the subjects and the (dynamic) access control can be enforced by a special part of the workflow engine.

4. Petri Nets

Petri nets — a vast research area with many publications — originated with Carl Adam Petri [18]. This section gives

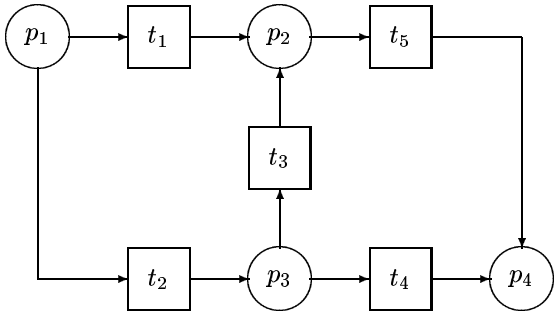

Figure 1. Example of a Petri net

the basic definitions [19].

Definition 1 (Petri Net) *A* Petri net N *is a triple* $N = (P, T, F)$. P *is the finite set of the* places, T *the finite set of the* transitions *with* $P \cap T = \emptyset$. *The* flow relation F *is defined by*

$$F \subseteq (P \times T) \cup (T \times P).$$

Let $y \in P \cup T$. $\bullet y$ *is called the* preset *of* y *and is defined by*

$$\bullet y := \{x \in P \cup T | (x, y) \in F\}.$$

$y\bullet$ *is called the* postset *of* y *and is defined by*

$$y\bullet := \{x \in P \cup T | (y, x) \in F\}.$$

Figure 1 shows an example of a Petri net. The net consists of the places $p_1, \ldots, p_4$ and the transitions $t_1, \ldots, t_5$. The sets of P, T, and F are defined as follows:

$$\begin{aligned} P &= \{p_1, \ldots, p_4\}, \\ T &= \{t_1, \ldots, t_5\}, \\ F &= \{(p_1, t_1), (p_1, t_2), (t_1, p_2), (t_2, p_3), (p_3, t_3), \\ &\quad (t_3, p_2), (p_2, t_5), (t_5, p_4), (p_3, t_4), (t_4, p_4)\} \end{aligned}$$

The graphical interpretation of a Petri net is a bipartite graph. Places can only be connected to transitions, transitions can only be connected to places. Places are represented graphically as circles, transitions as rectangles. The graphical interpretation of an element $(x, y) \in F$ is an arrow from x to y. In the example $(t_1, p_2) \in F$. Therefore an arrow is connecting transition t_1 with place p_2.

The preset and postset of a transition is a set of places. This set can be empty. The preset and postset of a place is a set of transitions. This set can be empty, too. Presets and postsets from the example: $\bullet t_1 = \{p_1\}, t_1\bullet = \{p_2\}, p_3\bullet = \{t_3, t_4\}$, and $\bullet p_1 = \emptyset$.

Definition 2 (Behavior of Petri Nets) *A non empty set* M *with* $M \subseteq P$ *is called a* marking *of a Petri net. A transition* t *is called* activated *under marking* M*, if*

1. $\bullet t \subseteq M$ *and*
2. $M \cap t\bullet = \emptyset$.

An activated transition can fire. *If a transition* t *fires,* M *changes:* $M \xrightarrow{t} M'$. *The new marking* M' *is defined by*

$$M' := (M \setminus \bullet t) \cup t\bullet .$$

The first marking of a Petri net is called start marking.

Graphically a marking is represented by filled circles in all its places. These filled circles are called *tokens*.

The behavior of Petri nets shall be illustrated following the example in Figure 1. Let the start marking be $\{p_1\}$. A token is put on place p_1. The transitions t_1 and t_2 are activated. If t_2 fires, the token moves from p_1 to p_3. The two transitions t_3 and t_4 are activated now. If t_4 fires the token moves to p_4. Now there are no more activated transitions. Formalized:

$$\{p_1\} \xrightarrow{t_2} \{p_3\} \xrightarrow{t_4} \{p_4\}.$$

With start marking $\{p_1\}$ there are two other possibilities for the Petri net to execute: $\{p_1\} \xrightarrow{t_2} \{p_3\} \xrightarrow{t_3} \{p_2\} \xrightarrow{t_5} \{p_4\}$ and $\{p_1\} \xrightarrow{t_1} \{p_2\} \xrightarrow{t_5} \{p_4\}$.

For the specification of workflows it is important to express the concepts of

- sequence,
- parallelism, and
- conditionality.

A sequence can be modeled by Petri nets through transitions and places with only one input and output place or transition. Parallelism bases on transitions with multiple output places. The third concept — conditionality — is based on places with multiple output transitions (XOR split). Places p_1 and p_3 in Figure 1 are examples. The decision which of the transitions may fire in the conditional case is left to the workflow engine and is typically based on the value of a data item.

5. Workflow Specification with Petri Nets

This section describes the connection between workflows according to Section 3 and Petri nets according to Definition 1 and 2. A workflow specified by a Petri net will be called *Petri net workflow*, the underlying net will be called *workflow net*.

A Petri net workflow has the following characteristics:

- Activities in workflows correspond to transitions in Petri nets. Executing an activity corresponds to the firing of a transition.

- The marking of a Petri net represents the current state of a workflow. The tokens are also called *control tokens*. The control tokens represent the state of the workflow, i.e. which activities are activated. The flow relation says how the tokens can move in the net. The control flow of a workflow is determined by the flow relation and the start marking.

So far it has not been defined which subjects and data items are associated with each activity. To be more specific, we have to define which subjects may execute an activity and which data is needed for and created through the execution of the activity. This is done by the three functions Sbj, D_{in}, and D_{out}.

- Function Sbj maps transitions to a set of subjects. These subjects may execute the activity associated with the transition. Formalized:

$$Sbj : T \to \mathcal{P}(S) \setminus \emptyset$$

  $\mathcal{P}$ is the symbol for the power set (the set of all subsets) and S is the set of subjects. Note, that the workflow engine itself is a valid subject.

- On the one hand, the execution of an activity creates data. On the other hand, to perform an activity a subject uses data items that already exist. The data input and output set of a transition is defined through the two functions D_{in} and D_{out}:

$$D_{\text{in}}, D_{\text{out}} : T \longrightarrow \mathcal{P}(D)$$

  D is the set of all data items (cf. Section 3). The empty set is a valid function value. In this case no data are needed for the execution or no data are created through the execution of the activity.

Figure 2 shows Figure 1 extended with the attributes of the transitions defined by the functions $D_{\text{in}}, D_{\text{out}}$ and Sbj. In Figure 2 the two sets S, D are $S = \{s_1, s_2, s_3\}, D = \{d_1, \ldots, d_5\}$. Transition t_1 may only be executed by subject s_1. Several subjects can be assigned a transition, e.g. transition t_4 can be executed by the three subjects s_1, s_2, and s_3. Subject s_2 needs data item d_2 to execute transition t_3. By executing t_3 subject s_2 creates d_3. The values of D_{in} and D_{out} can be empty, e.g. $D_{\text{in}}(t_2) = \emptyset$. In this case no data is needed to execute the activity associated with the transition.

The execution of workflows is supervised by the workflow engine which creates and manages numerous instances of a workflow. The workflow specification provides a template which is the basis for instances of this workflow. Different data items are created for every workflow instance. Access control is enforced upon the data items of the workflow's instances. To implement the access control mechanism — which will be discussed in the following section — the workflow engine manages a Petri net for every instance. Every time a transition fires in an 'instance net', the marking has to be changed. It is therefore sufficient to examine only a single instance.

A problem characteristic for the workflow specification in this paper is, if the 'data flow' is correct, i.e., if the data items are existing when they are needed. For the following sections it will be assumed that the Petri net workflow — including the data flow — is correct.

To determine if a workflow net is completely executable, i.e., if the end marking is reachable from the start marking, is a non trivial problem. This problem is related to the reachability problem in Petri nets [16]. The question, if a workflow specification is syntactically and semantically correct, is very important. To answer this question concepts like *liveness*, *invariants*, *deadlocks*, and *soundness* have been introduced [19]. Van der Aalst [24] focuses on the verification of workflow nets.

6. Dynamic Access Control

The purpose of access control is to grant access to data items only to legitimate subjects. This is done by access control matrices [7]. An access control matrix Z grants subjects rights to data items. If S is the set of the subjects, D the set of the data items, and R is the set of possible access rights, Z can be defined as the function:

$$Z : S \times D \to \mathcal{P}(R)$$

According to this definition an access control matrix is two dimensional. The most commonly used access rights are *read* (=r) and *write* (=w). In addition to reading and writing there are other access rights, e.g. data items can be executed, updated, or appended to other data items.

If access rights are granted according to the current marking of the workflow net, then the access rights change with the marking of the Petri net. If the marking of the workflow net is taken into account in the access control matrix, Z must be extended by one dimension. The two dimensional access control matrix changes into a three dimensional one. The new access control matrix is called Z_{dyn}. Again, Z_{dyn} can be defined as a function:

$$Z_{\text{dyn}} : S \times D \times T \to \mathcal{P}(R)$$

Interpretation: Let us assume a subject s requests a specific privilege r for data item d. In order to use Z_{dyn} the access control mechanism checks the current marking of the workflow net for all activated transitions. Let t be an activated transition and $s \in Sbj(t)$:

1. If $r \in Z_{\text{dyn}}(s, d, t)$, privilege r is granted. If t is not activated any more, then r is revoked again.

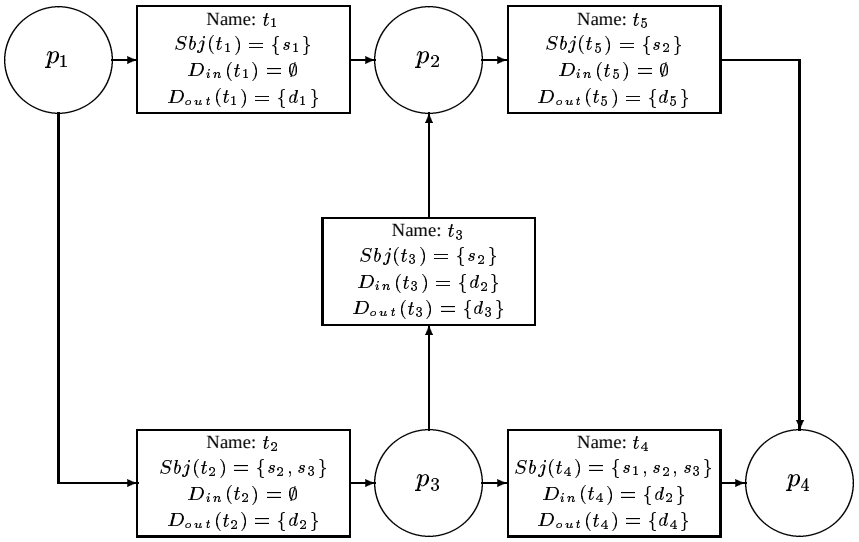

Figure 2. Example of a workflow net

2. If $r \notin Z_{\text{dyn}}(s,d,t)$, privilege r is not granted.

The extension from two to three dimensions increases the number of the entries. $|Z_{\text{dyn}}| = |S| * |D| * |T|$, while $|Z| = |S| * |D|$. The expression $|\ldots|$ denotes the number of elements in a set. In the example of Figure 2 the 'dynamic' access control matrix Z_{dyn} is a $3 \times 5 \times 5$ matrix. Altogether, there are 75 entries, most of which are empty, compared to Z with 15 entries.

A comparison between the 'traditional' access matrix Z and the 'dynamic' access control matrix Z_{dyn} leads to the following equation:

$$Z(s,d) = \bigcup_{t \in T} Z_{\text{dyn}}(s,d,t) \tag{1}$$

(1) must hold because all access rights from Z should be included in Z_{dyn} as well. The access rights are only distributed in a larger matrix. (1) shows that Z_{dyn} grants access rights more restrictively than Z. Nevertheless, all subjects have all the permissions they need to execute the activities they are assigned. The risk of data misuse in Z_{dyn} is lower than in Z since

$$Z_{\text{dyn}}(s,d,t) \subseteq Z(s,d)$$

which is an immediate consequence of (1). In most cases $Z_{\text{dyn}}(s,d,t)$ will be a real subset of $Z(s,d)$. From this point of view a workflow using dynamic access rights is more secure than a workflow without.

If we apply a special security policy, the access control matrix can be derived directly from the workflow specification.

Definition 3 (RW Security Policy) *The rw security policy holds, if subjects which are legitimate to execute an activity are granted access rights to the data input items and write rights to the data output items.*

If we apply the rw security policy, the access control rules can be expressed as

$$Z_{\text{dyn}}(s,d,t) = \begin{cases} \{r\}, & \text{if } s \in Sbj(t) \text{ and } d \in D_{\text{in}}(t) \\ \{w\}, & \text{if } s \in Sbj(t) \text{ and } d \in D_{\text{out}}(t) \\ \emptyset, & \text{otherwise} \end{cases} \tag{2}$$

where $s \in S, d \in D$, and $t \in T$. $Z_{\text{dyn}}(s,d,t) = \{r,w\}$ is not possible, because a data item has to be written before it can be read. Rewriting or appending information to a data item should produce a new data item.

To illustrate the differences between Z and Z_{dyn} we will apply the rw security policy to the example in Figure 2. Matrix (3) shows how the access control matrix Z of the example would look like if the granting of access rights were static, i.e. not based on the state of the workflow.

	d_1	d_2	d_3	d_4	d_5
s_1	$\{w\}$	$\{r\}$	$\emptyset$	$\{w\}$	$\emptyset$
s_2	$\emptyset$	$\{r,w\}$	$\{w\}$	$\{w\}$	$\{w\}$
s_3	$\emptyset$	$\{r,w\}$	$\emptyset$	$\{w\}$	$\emptyset$

(3)

Explanation through an example:

- Static case: Subject s_3 needs write permission on d_2 because it could create d_2 by executing t_2. Also, subject s_3 needs read permission on d_2 when executing t_4. Therefore $Z(s_3, d_2) = \{r, w\}$.

- Dynamic case: $Z_{\text{dyn}}(s_1, d_4, t_4) = \{w\}$, since subject s_1 may need write permission on d_3 in t_4. $Z_{\text{dyn}}(s_2, d_2, t_3) = \{r\}$, since s_2 needs read access to d_2 to perform t_3.

On the one hand, the rw security policy is based on $R = \{r, w\}$ and neglects all other access rights. On the other hand, since Z_{dyn} can be directly derived from the workflow net, increased security can be deduced without an 'security extension' of the workflow specification. This is of high practical relevance: in a WfMS the workflow engine can be extended with the proposed access control mechanism to enforce the rw security policy based on the 'traditional' workflow specification.

7. Case Study

This section illustrates the formal aspects of the previous sections through an example. The workflow chosen deals with the handling of an insurance claim. Further information about the process can be found in [6]. Figure 3 shows a semi-formal description of the workflow. The rw security will be applied.

A clerk checks if police reports, witness reports, and quotations are available, initializes the workflow, and creates a claim form which contains information about the type and value of the claim. An assessor has to pass an expert opinion on the physical evidence of the claim (e.g. a wrecked car) in (5). Depending on the value of the claim, (6) or (3) follow. Based on the type of the claim ('vehicle' or 'household') there is a XOR split. (7)/(8) or (9)/(10) can be done in parallel, respectively. Finally, a claim manager approves the claim based on the results of the previous activities (14).

Control flow Note that Figure 3 is not sufficient to define the exact control flow of the workflow. The Petri net specification of the workflow in Figure 4 is more precise but requires more elements. t_1 involves the initialization of the process by generating a new document called 'claim schedule'. After t_1 an AND split to p_2 and p_4 is done. In p_2 depending on the value of the claim the workflow engine performs a XOR split to t_2 or t_6. In p_3 depending on the type of the claim a XOR split is performed, either to t_3 or to t_4. In the t_3-branch t_7 and t_8 can be done parallel, the same is true for t_9 and t_{10} in the t_4-branch. t_{11} merges the t_3-branch, t_{12} the t_4-branch. Similarly, p_{15} corresponds to p_3, p_{14} to p_2, and t_{14} to t_1. The final activity is t_{14}.

Note, that there are more transitions than activities because special transitions like t_{11}, t_{12}, and t_{13} are needed to join different branches of the net which have been split before.

Subjects The role of the subjects allowed to perform an activity are written in bold face below the description of the activity in Figure 3. The workflow engine is responsible for transitions $t_2, t_3, t_4, t_{11}, t_{12}$, and t_{13} in Figure 4.

Data items and data flow Altogether, there are eleven data items $(d_0, \ldots, d_{10})$ included in the example as shown in Table 1. A police report d_0, witness reports d_1, and quotations d_2 are the data input items for t_1. Data output items of t_1 are the claim form d_3, the claim type d_5, and claim value d_6. If p_2 is marked in the net, the workflow engine decides based on the value of d_6 if t_2 or t_6 is activated. Similarly, the value of d_5 is decisive for the split following p_3. The assessor in t_5 needs d_3 to produce d_4. The clerk in t_6 needs d_3 to produce d_7. The following table gives the data input and output items for the other transitions.

Transition	D_{in}	D_{out}
t_1	d_0, d_1, d_2	d_3, d_5, d_6
t_5	d_3	d_4
t_6	d_3	d_7
t_7	d_2	d_8
t_8	d_0, d_1	d_9
t_9	d_2	d_8
t_{10}	d_0, d_1	d_9
t_{14}	d_3, d_4, d_7, d_8, d_9	d_{10}

The number of data items is so high, because the model reduces the access rights to read and write. The number of data items could be reduced if other privileges were introduced or through the use of an object oriented model where parts of an data item can be accessed.

The workflow specification represents a template which is used to generate instances of the workflow. In the example, the WfMS generates an instance for every claim. Note, that the execution order of activities can be different in different instances. For example $\{p_1\} \xrightarrow{t_1} \{p_2, p_4\} \xrightarrow{t_5, t_6} \{p_{13}, p_{14}\} \xrightarrow{t_{14}} \{p_{16}\}$ for a first claim and $\{p_1\} \xrightarrow{t_1} \{p_2, p_4\} \xrightarrow{t_2} \{p_3, p_4\} \xrightarrow{t_3} \{p_4, p_5, p_6\} \rightarrow \ldots \xrightarrow{t_{14}} \{p_{16}\}$ for a second claim.

The following three examples illustrate the access control mechanism introduced in Section 6:

- Let the marking of a workflow instance be $M = \{p_3, p_4\}$. In this case transitions t_3, t_4 and t_5 are activated. The assessor role gets the necessary privileges for d_3 and d_4, read access to d_3 and write access to d_4

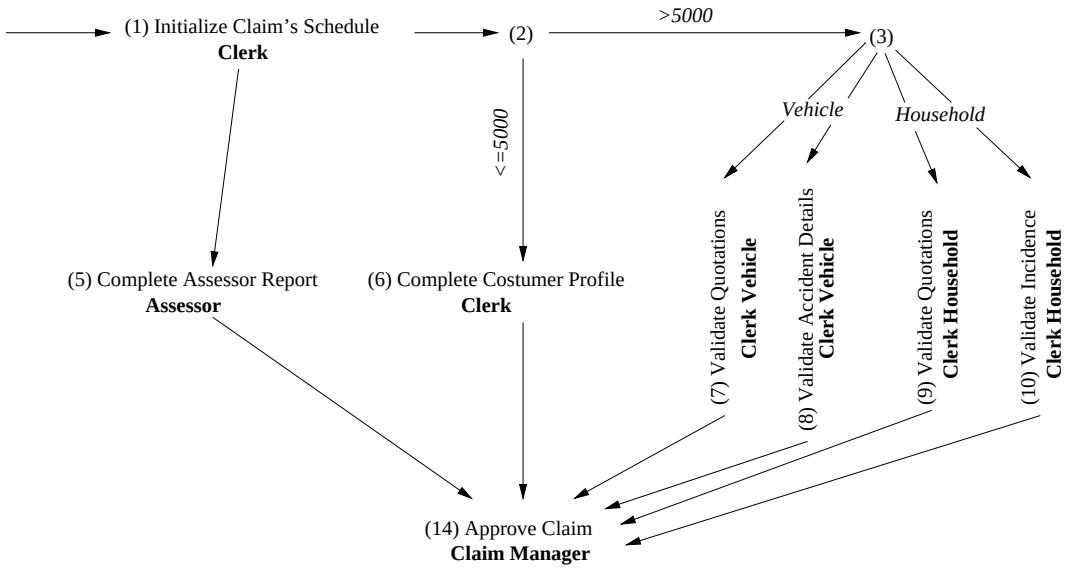

Figure 3. 'Insurance claim' workflow — textual form

if the rw policy is applied. For t_3, t_4 the workflow engine has to decide which branch to take based on the type of the claim. The workflow engine has all possible privileges because it is responsible for the correct propagation and storage of the data items. No other role has privileges of any kind.

- $M = \{p_5, p_6, p_{13}\}$: Transitions t_7 and t_8 are activated. Because t_7, t_8 may be executed by the same role, the necessary privileges for t_7 and t_8 are granted to the 'Clerk Vehicle' role. In case of the rw policy read access to d_0, d_1, d_2 and write access to d_8, d_9 is granted.

- $M = \{p_3, p_{13}\}$: No role gets any privileges. The workflow engine has to decide first, if the t_3 or t_4 branch is to be executed.

These three examples show the access control mechanism for roles. In practice, access rights are granted to the end user and not to roles. There are various publications on role based access control [20] and on the resolution of role assignments [3, 5, 6]. The objective of role based access control is to simplify the privilege management of the users by introducing a middle layer between subjects and objects — so called *roles*. A subject may be a member of different roles and many subjects may be able to activate the same role. The approach gets more complicated if role hierarchies are introduced.

The assignment of activities to the actual user is done via a *work list*. A user's work list contains all pending activities in the different instances of the workflow (and other workflows if the WfMS manages various workflows). In the example the marking $\{p_2, p_4\}$ would result in an entry 'Complete Customer Profile' in the work list of a clerk and 'Complete Assessor Report' in the work list of an assessor.

If a user decides to execute a transition, his access rights for the data items of the chosen activities are granted according to the companies security policy. After the transition has fired, the access rights are revoked again. Note, that if a user has decided to perform an activity, the workflow engine has to lock this activity of the workflow instance for all other users of the system. Also, there should be a 'time out' after which the user's rights are revoked automatically and the activity is unlocked again. If the user did not perform the activity, this should be noted in a special log file.

A prominent requirement for an access control mechanisms is *Separation of Duties* (SoD). E.g. in the insurance claim workflow a person who is capable to activate the clerk role (t_1) and the claim manager (t_{14}) role should not be able to perform these two activities in the same workflow instance. SoD is not enforced in the proposed model. Nevertheless, the workflow engine can use its log files to enforce SoD. Every time a transition fires, the workflow engine adds an entry to its log file, e.g. claim number, user, activated user role, activity, and time stamp. Based on the history of the instance captured in the log file, SoD can be enforced.

8 Discussion and Outlook

This paper has shown how access rights can be derived from a Petri net workflow dynamically. A subject has only the access rights which are needed for 'activated' activities.

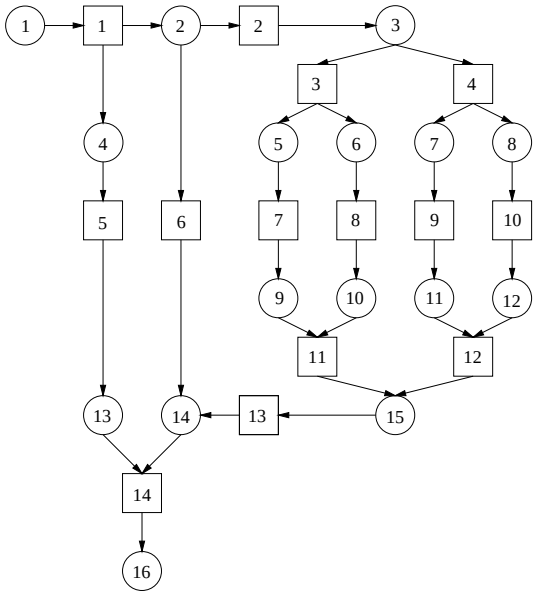

Figure 4. 'Insurance claim' workflow — Petri net

By doing this, unnecessary access is eliminated. Through the application of the rw security policy the access rights could be directly derived from the workflow net. The 'traditional' two dimensional is extended to a three dimensional matrix through this practice. Furthermore, it was shown how a workflow engine can enforce the proposed mechanism.

As a prerequisite of the proposed security mechanism there has to be a WfMS. Today's WfMSs have the following drawbacks:

- Many business processes have no clear structure and cannot be formalized to an extent necessary for a WfMS. They cannot be put into a 'computer-understandable' format like a workflow specification. Next to the missing structure there are exceptions during run time and even a change to the workflow specification may be necessary. 'Exception handling' is a major research area in the field of workflow management.

- Many existing WfMS struggle with technical problems such as connecting to legacy systems, workload management, and availability of the system.

- Finally, social aspects are of interest, too. A WfMS monitors all the activities of its users. It is very easy to investigate the working habits of individuals. Furthermore, Taylorism is an issue. Users of a WfMS are forced to do activities over and over again.

d_0	police report
d_1	witness reports
d_2	quotations
d_3	claim form
d_4	assessed claim form
d_5	claim type
d_6	claim value
d_7	completed claim form
d_8	validated vehicle/household quotations
d_9	validated vehicle accident details/ household incidence
d_{10}	approved claim

Table 1. 'Insurance claim workflow' — data items

As a result of these shortcomings a breakthrough of workflow technology has not taken place. Not many companies use workflow technology in a large scale.

In a realistic environment the size of a Petri net workflow may grow very large. Fortunately, Petri net tools such as Design/CPN [13] can manage Petri nets with several thousand places and transitions. So the size should cause no serious limitations.

If a company uses a WfMS we suggest to implement the proposed mechanism as an add-on module. Commercial WfMSs are typically not based on Petri nets. Nevertheless, every workflow engine is capable of providing the state of its workflow instances. Since workflow propagation in itself is a complex issue, the access control mechanism does not try to redesign and/or simulate the workflow process but will merely get the workflow state from the workflow engine and manage a workflow net for every instance. This information together with a security policy for the different workflows can be used to apply the mechanism introduced in this paper. If log information from previous activities is taken into consideration, it might be possible to enforce SoD at the same time.

In the formal descriptions of the model some restrictive assumptions were made which shall be loosened in the future, providing the following research topics:

- The rw security policy can be extended by other privileges like execute, update, and append. As indicated, this implies an extension of the workflow specification. Future work will focus on different policies going beyond rw.

- In this paper the data items have been described as attributes of transitions. With *Coloured Petri Nets* [13] they could be presented as individual tokens. In this way the data flow could be described as a subnet of the workflow net. Existing Petri net methods could be

applied on the data flow. Furthermore, the information flow within such a net is an interesting research area.

- The security of workflows is increased by a reduction of data misuse possibilities. The confidentiality of the workflow data is increased. In addition to confidentiality there are other security objectives like availability, integrity, and non-repudiation. Future research will deal with the question how these other objectives can be integrated in Petri net workflows.

Acknowledgments

This work was supported by the Swiss National Science Foundation under grant SPP-ICS 5003-045359. The author thanks Jan Eloff, Joachim Kreutzberg, and Kurt Bauknecht for the discussion of this paper.

References

[1] N. R. Adam, V. Atluri, and W.-K. Huang. Modeling and Analysis of Workflows Using Petri Nets. *Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management*, 10(2):131–158, March 1998.

[2] V. Atluri and W.-K. Huang. An Authorization Model for Workflows. In *Proceedings of the 4th European Symposium on Research in Computer Security*. Springer, 1996.

[3] E. Bertino, E. Ferrari, and V. Atluri. A Flexible Model Supporting the Specification and Enforcement of Role-based Authorizations in Workflow Management Systems. In *Proceedings of the 2nd ACM Workshop on Role-based Access Control*, Fairfax, Virgina, 1997.

[4] E. Bertino, E. Ferrari, and V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Transactions on Information and System Security*, 2(1):65–104, February 1999.

[5] C. J. Bussler. Policy Resolution in Workflow Management Systems. *Digital Technical Journal*, 6(4), 1994.

[6] D. Cholewka, R. Botha, and J. Eloff. A Contex-Sensitive Access Control Model and Prototype Implementation. In J. H. Eloff and S. Qing, editors, *16th Annual Working Conference on Information Security*, pages 341–350. IFIP TC 11, Kluwer Academic Publishers, August 2000.

[7] D. E. R. Denning. *Cryptography and Data Security*. Addison-Wesley, 1983.

[8] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[9] L. Harn and H. Lin. Integration of User Authentication and Access Control. *IEE Proceedings-E*, 139(2), March 1992.

[10] R. Holbein. *Secure Information Exchange in Organisations — An Approach for Solving the Information Misuse Problem*. PhD thesis, Universität Zürich, 1996.

[11] R. Holbein, S. Teufel, O. Morger, and K. Bauknecht. Need-to-know Access Controls for Medical Systems. In *Proceedings IFIP TCII WG 11.2 Small Systems Security*, Copenhagen, 1997.

[12] S. Jablonski, M. Böhm, and W. Schulze, editors. *Workflow-Management — Entwicklung von Anwendungen und Systemen — Facetten einer neuen Technologie*. dpunkt.verlag, 1997.

[13] K. Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use, Volume 1-3*. EATCS Monographs on Theoretical Computer Science. Springer, 1997.

[14] L. G. Lawrence. The Role of Roles. *Computers & Security*, (12):15–21, 1993.

[15] F. Leymann and W. Altenhuber. Managing Business Processes as an Information Resource. *IBM Systems Journal*, 33(2):326–348, 1994.

[16] E. W. Mayr. An Algorithm for the General Petri Net Reachability Problem. *SIAM Journal*, 13(3):441–460, 1984.

[17] A. Oberweis. *Modellierung und Ausführung von Workflows mit Petri-Netzen*. Teubner-Reihe Wirtschaftsinformatik. B.G. Teubner, 1996.

[18] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik der Universität Bonn, 1962.

[19] W. Reisig. *Petri Nets — An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[20] R. Sandhu. Role-Based Access Control Models. *IEEE Computer*, 29(2):34–47, Feb. 1996.

[21] R. K. Thomas and R. S. Sandhu. Towards a Task-based Paradigm for Flexible and Adaptable Access Control in Distributed Applications. In *Proceedings of the 1992-1993 ACM SIGSAC New Security Paradigms Workshop*, pages 138–142, Little Compton, RI, 1993.

[22] R. K. Thomas and R. S. Sandhu. Conceptual Foundations for a Model of Task-based Authorizations. In *Proceedings of the 7th IEEE Computer Secuity Foundations Workshop*, pages 66–79, Franconia, NH, June 1994.

[23] R. K. Thomas and R. S. Sandhu. Task-based Authorization Controls (TBAC): Models For Active and Enterprise-oriented Authorization Management. In *Proceedings of the 11th IFIP WG 11.3 Conference on Database Security*, Lake Tahoe, CA, August 1997.

[24] W. van der Aalst. Verification of Workflow Nets. In P. Azema and G. Balbo, editors, *Application and Theory of Petri Nets 1997, volume 1248 of Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, 1997.

[25] W. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

[26] Workflow Management Coalition. *Terminology and Glossary*. http://www.aiim.org/wfmc/, Feb. 1999. Document Number TC-1011.

[27] S. Yen and C. Laih. On the Design of Dynamic Access Control Scheme with User Authentication. *Computer & Mathematics with Applications*, 25(7):27–32, April 1993.