14

MODELING AND ANALYZING SEPARATION OF DUTIES IN WORKFLOW ENVIRONMENTS

Konstantin Knorr and Henrik Stormer
Department of Information Technology
University of Zurich, Switzerland
[knorr, stormer] @ifi.unizh.ch

Abstract With the rise of global networks like the Internet the importance of workflow systems is growing. However, security questions in such environments often only address secure communication. Another important topic that is often ignored is the separation of duties which is an important part of a company's security policy to prevent fraud. This paper introduces a prototype that supports the graphical modeling and analysis of separation of duties in workflow environments. Security officers can use this tool to design and analyze the security rules associated with workflow specifications.

Keywords: Fraud Control, Separation of Duties, Workflow

1. INTRODUCTION

It is well known that many computer related criminal activities are performed by insiders [6]. One of the most prominent threats is fraud that is particularly difficult to detect in computerized environments such as workflow systems. Therefore it is of great importance to implement mechanisms to prevent such illegal activities. Separation of duties (SoD) has been identified by many authors as an efficient mechanism to prevent fraud within organizations [1, 2, 3, 20]. SoD guarantees that certain critical tasks can only be done by a collusion among individuals. It is in particular useful when applied to dynamic processes such as workflows. The physical and logical separation of tasks and of their performing subjects can improve the prevention of fraudulent activities.

SoD has been in use long before the computer era. One prominent example is the 'four-eye-principle' that is found in many environments such as health

care. Another example: Most military systems require (at least) two persons to launch a nuclear missile.

This paper introduces MASoD, a prototype supporting the graphical Modeling and Analysis of SoD-rules in workflow environments. Workflows are computer-understandable business processes whose modeling, administration, and execution is supported by a software package called workflow management system (WfMS). The effort associated with introducing a WfMS in an organization is tremendous. Especially (1) the implementation of the new system, integration and interfacing with existing systems, and (2) the identification, redesign and specification of the processes to be automated are time and resource consuming. Concerning the second point, we share the view of Huang and Atluri [9] who argue that security officers should utilize existing data when designing security policies. Therefore, MASoD is capable of importing existing workflow specifications, and provides users with organizational and process information to generate SoD-rules. Most workflow systems allow users to model workflows graphically. In continuation of this practice, SoD should be modeled in the same way. Also, in the past visual languages have proved to be user-friendly.

The paper has the following structure: Section 2 discusses the background topics of workflow management and SoD. A sample business process is introduced in Section 3. While Section 4 focuses on the graphical modeling of SoD-rules on top of existing workflow specifications, these rules are analyzed in Section 5. Section 6 concludes the paper with a discussion and an outlook.

2. BACKGROUND

2.1. WORKFLOW MANAGEMENT

Business processes represent an essential part of the commercial activity of a company. Especially for frequent processes, support for automation is an important topic. Workflow management is an emerging technology in the area of applied computer science dealing with this issue. A WfMS is a software system that supports the modeling, execution, and administration of business processes. Defined in four words, a workflow is a *computer-understandable business process.* Before a workflow can be executed it has to be described in a way the WfMS is able to understand. This description is called workflow specification and is done during the so-called *build time* of a WfMS. 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 [4, 7, 15].

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 manage and forward process-specific data items to the

subjects. The main elements of a workflow specification are: (1) tasks, (2) the control flow, and (3) subjects/roles — or more general the organizational structure. The basic building blocks of a workflow are its tasks whose temporal and logical order is given by the control flow. To describe a task, the activity and the corresponding subjects have to be specified. Subjects can be associated with persons, but also processes and computer programs such as the workflow engine are possible subjects. A subject executes a task by creating new and/or using already existing data items.

Prominent workflow specification languages are Petri Nets [16], State Charts [23] and the Workflow Process Definition Language proposed by the Workflow Management Coalition [24] — a non profit organization dealing with standardization in workflow systems.

In the last few years, the role concept has proven to be very successful in commercial settings. A role defines a group of human beings with a special knowledge or skill. Usually, a specific role is tied to each task in a workflow. Doing this, the administration of users is simplified since subjects can be added or removed without changing the workflow specification. In practice, an important application of the role concept is security. Access rights are not granted to single persons but to roles which simplifies and cheapens security administration. This is called role based access control [14, 17].

2.2. SEPARATION OF DUTIES

Gligor et al. [8] define SoD as "a policy to ensure that failures of omission or commission within an organization are caused only by collusion among individuals and, therefore, are riskier and less likely, and that chances of collusion are minimized by assigning individuals of different skills or divergent interests to separate tasks". Clark and Wilson [5] stress the importance of SoD mechanisms in commercial settings. Within a business process context, SoD-rules express task dependencies and should be part of a company's security policy. Only recently has the combination of workflow management and SoD found considerable interest in the research community [1, 2, 22].

In a workflow context, SoD has to be divided and extended into static and dynamic SoD. Static SOD enforces certain rules during build time of the workflow and is therefore applied to the workflow specification. In contrast, dynamic SoD is enforced during run time. For the remainder of this paper we will concentrate on dynamic SoD. Note that dynamic SoD is based on the history of a business process and can therefore only be enforced during run time of a WfMS.

3. SAMPLE PROCESS

The last section introduced several concepts on an abstract level. This section gives a sample business process that will be used for illustration purposes

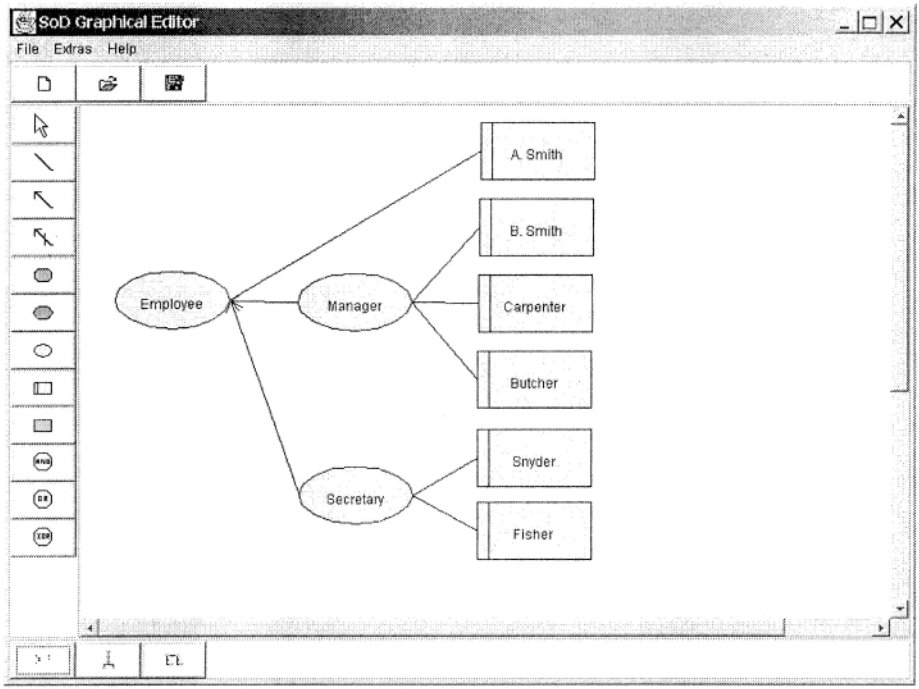

Figure 1 Sample organizational model. Six subjects and three roles are defined.

throughout the remainder of the paper. The business process deals with travel expense reimbursement. The workflow consists of four tasks: in a first task (*submit*), an employee applies for the reimbursement of his travel expenses by filling out an application form. Two managers have to approve this report (tasks *approve 1* and *approve 2*). Finally, based on the approval of the managers, a secretary will transfer the money to the employee's bank account (*pay*). Figure 3 gives a visualization. A discussion of the Notation will follow in Section 4.2. Note that the workflow engine creates an instance of this workflow for every travel of an employee.

Formally, the set of roles encompasses the manager, secretary, and employee role. Let the subjects of the company be Carpenter, Butcher, Snyder, Fisher, and the brothers A. Smith and B. Smith. All six subjects are employees, Carpenter, Butcher and B. Smith are managers, Fisher and Snyder are secretaries. Note that every manager (or secretary) is an employee, too. The partial order of roles builds up a so called role hierarchy. Figure 1 shows the subjects, roles, and the role hierarchy as modeled with MASoD.

Now we are ready to give SoD examples. According to the process definition different tasks are performed by different roles (the pay task by a secretary

and the approve 1 task by a manager). This corresponds to static SoD-rules. However, dynamic SoD is of greater interest for this paper. The following rules are reasonable for this specific business process:

1 A manager should not be allowed to approve his/her own travel expense claim.

2 B. Smith should not be allowed to approve the claim of his brother A. Smith.

3 A secretary should not be allowed to transfer the refund of his/her own travel expenses.

4 A manager should not be allowed to perform both approval tasks in the same workflow instance.

These examples clearly state the need for a formal model for SoD that takes into account the state of the underlying business process

4. MODELING OF SOD-RULES

Process and role definitions are modeled graphically in most workflow systems. Therefore, the modeling of SoD should also be done graphically. However, most of the existing SoD-languages are difficult to adapt for graphical notations. Besides, those languages are not intuitive and designed for security experts. We argue that the use of SoD depends on a simple language. Therefore, we introduce SSoDL (Simple SoD Language) and show how to graphically model SSoDL-rules in this section.

The modeling of SoD-rules requires the existence of organizational and process models. Therefore, MASoD (our prototype) supports the modeling of three different components:

- roles, subjects, and role hierarchy
- process
- SoD-rules

Note that the models should be created in the order indicated in the listing.

Section 4.1 will briefly discuss how MASoD handles organizational and process models. However, the major focus is on the graphical modeling of SoD-rules in Section 4.2.

For the comfort of the reader, Figure 2 gives a summary of all symbols used for the three models supported by MASoD. All the symbols will be explained in the following subsections.

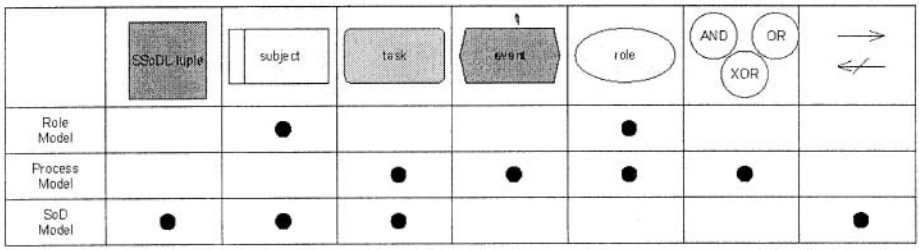

Figure 2 Symbols used in MASoD. Each row of the table represents a modeling component. Each column is one or more symbols. A bullet indicates that the symbol(s) is/are used in the modeling component.

4.1. GRAPHICAL MODELING OF ROLES AND PROCESSES

MASoD is designed to import existing process definitions. If the tool is used on a stand-alone basis, a workflow can also be specified graphically as an EPC (Event-driven Process Chain)— a model that was introduced by Scheer [21] and is frequently used in Europe, especially in German speaking countries. EPCs are an important part of a more complex product family called ARIS that allows for modeling all major components of a company.

We now discuss how the organizational structure is modeled using the MA-SoD editor. Figure 1 shows the organizational definition corresponding to the sample discussed in Section 3. Roles are represented as ellipses, subjects as rectangles, and the role membership as lines connecting subjects with roles. The role hierarchy is illustrated as roles connected with arrows. Six subjects and three roles are defined. Carpenter, for instance, is assigned the manager role. Because of the role hierarchy, Carpenter is an employee, too.

When the organizational model has been designed, the process modeling can commence. Note that we use EPC terminology. Every workflow is identified with one EPC. An EPC consists of an alternating chain of events and functions. It starts and ends with an event. A function is used to model a task. Only roles that have previously defined in the organizational model can be applied to functions, not subjects. Graphically, a function is represented as a rounded box, an event as a sexangle. One role has to be tied to each function which is visualized through a connector. Arrows represent the control flow. Branchings are modeled using AND, OR, and XOR nodes. Figure 3 gives the EPC according to our example. Scheer [21] gives a more comprehensive introduction to EPCs.

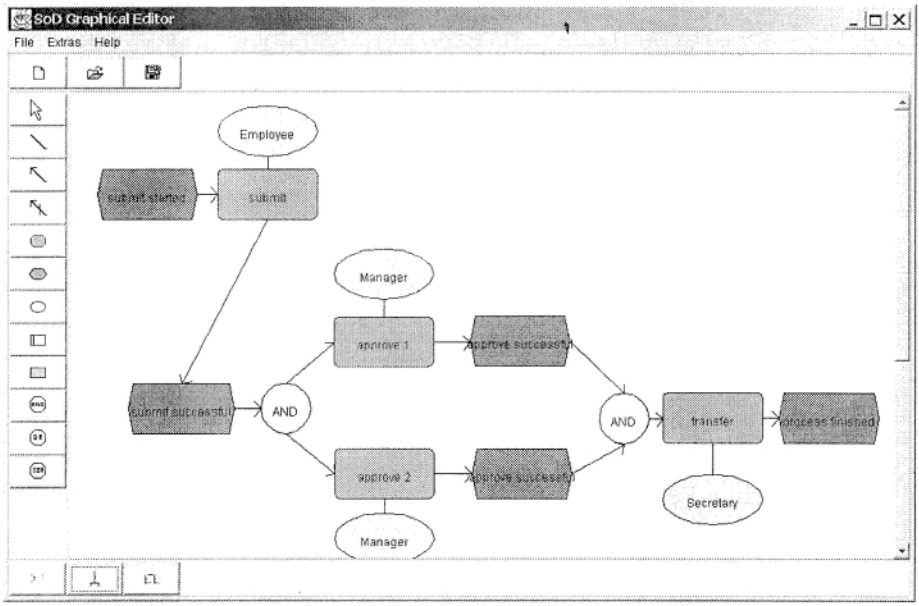

Figure 3 Sample process model with four tasks.

4.2. GRAPHICAL SOD MODELING

The basis of our graphical model is SSoDL (Simple SoD Language) which describes SoD-rules. These rules have the following form:

$$(s_1, t_1) \rightarrow (s_2, t_2) \tag{1}$$

or

$$(s_1, t_1) \not\rightarrow (s_2, t_2) \tag{2}$$

Rule (1) represents a **delegation of duties**. If subject s_1 has performed task t_1, then subject s_2 has to perform task t_2. Rule (2) represents a **separation of duties**, if subject s_1 has performed task t_1, then subject s_2 must not perform task t_2. An example from Section 3: (Butcher, approve l) $\not\rightarrow$ (Butcher, approve 2).

Note that the legitimacy of a subject to perform a task is given by the role membership of the subject, i.e. only subjects that are member of the role associated with task t_1 are allowed to perform task t_1. In an EPC, for each task there is exactly one role. Note also that $s_1 = s_2$ is possible, but that t_1 must preceed t_2. The partial order of the tasks is given by the underlying EPC.

The rule that a manager should not do both approval tasks would result in 12 SoD-rules if rules could only be defined on a subject/task level, which would be quite unsatisfactory in environments with a large number of subjects.

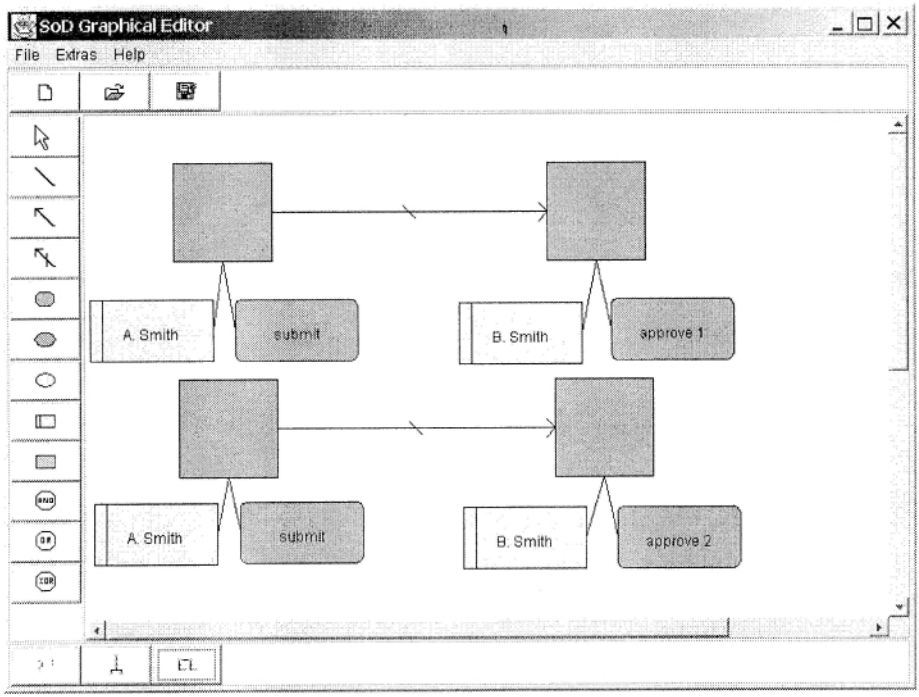

Figure 4 Graphical modeling of the brother example.

Therefore, we introduce the ?-notation: For s_1 and s_2 in (1) and (2), a ? can be inserted, $(?, t_1) \rightarrow (s_2, t_2)$ means that rules of type (1) are generated in the MASoD analyzer (cf. Section 5) for all subjects that are member of the role associated with task t_1.

The ?-notation can also be used on both sides of (1) and (2). Example: $(?, t_1) \not\rightarrow (?, t_2)$. In this case, subject/task rules of type (2) are generated for all subjects who can perform task t_1 and t_2. Note that this practice requires a role hierarchy or the membership of subjects to multiple roles. An example from Section 3: The two rules (?,submit) $\not\rightarrow$ (?, approve 1) and (?,submit) $\not\rightarrow$ (?, approve 2) would prevent managers from approving their own travel claims.

We will now discuss how to model these rules graphically. The subject/task tuples are illustrated as rectangles (subjects) and rounded boxes (tasks or functions in EPC terminology). Note that MASoD only allows for subjects and tasks which have previously been defined in the organizational and process model. To represent subject/task tuples, subjects and tasks are connected to a square. Separation and delegation arrows ($\rightarrow$ and $\not\rightarrow$) are used between two squares. The graphical modeling is illustrated using the example rules 2 and 4 from the sample business process. The first rule discussed is the brother example from

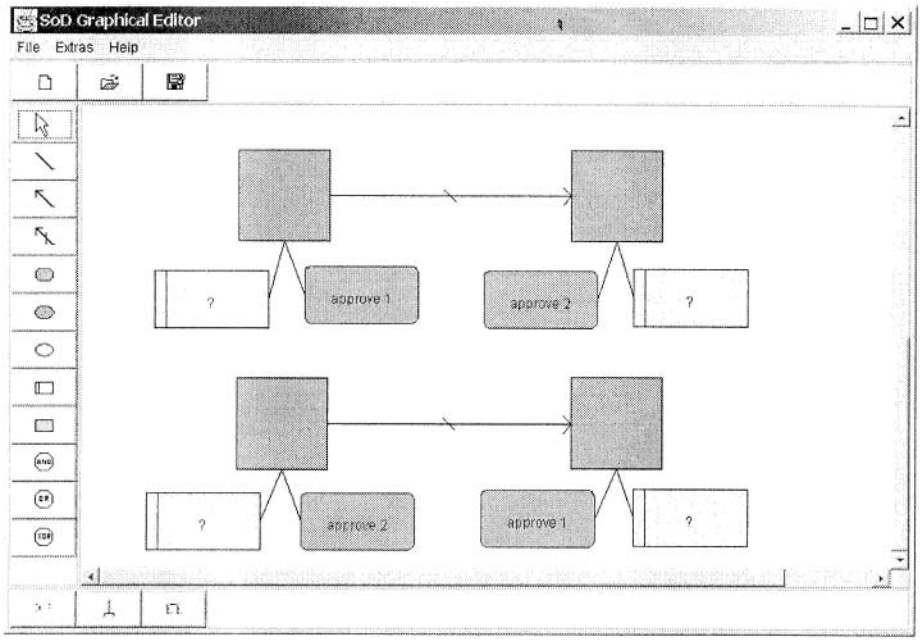

Figure 5 Second graphical SoD example with ?-notation.

Section 3. If A. Smith has done the submit task, then B. Smith is not allowed to do one of the approve tasks. Using the SSoDL syntax, this rule can be expressed as:

```
(A. Smith, submit) ↛ (B. Smith, approve 1)
(A. Smith, submit) ↛ (B. Smith, approve 2)
```

A graphical model of this rule in the MASoD editor is shown in Figure 4. Note the one-to-one relation between the SSoDL rules and the graphical notation. In the fourth SoD example rule in Section 3, a manager is not allowed to do both approve tasks. This can be expressed using the ?-notation:

```
(?, approve 1) ↛ (?, approve 2)
(?, approve 2) ↛ (?, approve 1)
```

Figure 5 shows the graphical modeling of this example. Note that the analyzer is creating the following six rules by substituting the ?:

```
(Butcher, approve 1) ↛  (Butcher, approve 2)
(B. ~Smith, approve 1) ↛  (B. ~Smith, approve 2)
(Carpenter, approve 1) ↛  (Carpenter, approve 2)
(Butcher, approve 2)  ↛ (Butcher, approve 1)
```

(B.˜Smith, approve 2) $\not\rightarrow$ (B.˜Smith, approve 1)
(Carpenter, approve 2) $\not\rightarrow$ (Carpenter, approve 1)

5. SOD ANALYSIS

Section 4 introduced SSoDL, a language for SoD, and its graphical modeling. On the basis of this, we will now show how to analyze the resulting SoD-rules. This section only gives an overview. The detailed description of the SoD analysis including a comprehensive description of the Prolog program and the analysis technique can be found in [12].

The rules for each process are contained in a so-called *Rule Base* (RB). The RB will only consist of rules as shown in (2). That means that the ?-notation has to be removed (cf. Subsection 4.2) and that rules of type (1) have to be transformed to rules of type (2) according to the following equivalence:

$$(s_1, t_1) \rightarrow (s_2, t_2) \Longleftrightarrow [(s_1, t_1) \not\rightarrow (x, t_2) \quad \forall x \in S \setminus \{s_2\}]$$

where *S* is the set of all subjects. For a discussion of the complexity of this transformation see [12].

The analysis of the SoD-rules makes use of logical programming — to be precise Prolog. As a preparation, the process model (EPC) is transformed into a Petri net [10, 18]. The transformation step is well understood [13, 19]. The Petri net representation can be seen as a formalized version of the EPC. Now, the following steps are necessary:

- Transformation of the organizational structure into facts of the logical program
- Transformation of the Petri net into facts of the logical program
- Transformation of the SSoDL-rules into facts of the logical program

After the transformations, MASoD checks if the rules are sound, i.e. if the rules obey to the organizational and process structure. Example: the rule

(A. Smith, pay) $\rightarrow$ (Butcher, submit)

is not sound since A. Smith is not a secretary and the submit task precedes the pay task. Given a sound rule-base, the program generates all valid execution chains of the workflow. An execution chain consists of subject/task tuples that represent the firing order of the underlying Petri net. If there is no contradiction to the SoD rule-base, the execution chain is called valid. Examples from our sample: Let L1, L2, L3 be three chains with

```
L1  =  [(Fisher, submit), (A. Smith, approve 1),
       (Butcher, approve 2), (Snyder, pay)]
L2  =  [(Fisher, submit), (Butcher, approve 1),
       (Butcher, approve 2), (Fisher, pay)]
L3  =  [(Fisher, submit), (Carpenter, approve 1),
       (Butcher, approve 2), (Snyder, pay)]
```

`L1` is no execution chain since A. Smith is no legitimate subject for one of the approve tasks (a manager is needed). `L2` is an execution chain but not SoD valid since Fisher transfers his/her own travel expenses and Butcher does both approval tasks. `L3` is SoD valid.

The Prolog program generates 28 valid execution chains in our example:

```
1.   [(A.Smith, submit), (Carpenter, approve 1),
     (Butcher, approve 2), (Fisher, pay)]
2.   [(A.Smith, submit) , (Carpenter, approve 1),
     (Butcher, approve 2), (Snyder, pay)]
     ...
28.  [(Snyder, submit), (Carpenter, approve 1),
     (Butcher, approve 2), (Fisher, pay)]
```

A further analysis of these 28 valid execution chains yields the following:

- For every employee in the example, the travel expense workflow can be finished. No workflow has to be canceled due to too restrictive SoD-rules. Although this statement may seem obvious for the example, in a more complex setting it is not.

- Every workflow requires four different persons for its execution. Without the SoD-rules two persons would be sufficient (e.g. first three tasks by a manager, last one by a secretary).

- A further analysis of the valid execution chains can be used to do 'workload management'. In the example, the managers Carpenter and Butcher have more work because of the 'brother rule' (cf. Table 1).

- The analysis shows that the rule base of the example contains no contradicting rules.

Technically, MASoD generates three files: the organizational, Petri Net and SoD Prolog facts. Then, the Prolog program is called that reads the files, processes them as input, and generates an output file. The data from the output file is read into MASoD and is graphically displayed.

$N(s,t)$	A.Smith	B.Smith	Carpenter	Butcher	Snyder	Fisher
submit	4	4	4	4	4	4
approve 1	0	8	10	10	0	0
approve 2	0	8	10	10	0	0
pay	0	0	0	0	14	14

Table 1 Statistical analysis of the sample workflow. $N(s, t)$ is the number of valid execution chains in which subject s performs task t

6. CONCLUSION AND OUTLOOK

Current surveys [6] show that fraudulent activities performed by insiders are a tremendous threat for the commercial success of a company. Therefore, appropriate counter-measures such as SoD have to be found and implemented.

The MASoD prototype supports the modeling of the organizational structure, processes, and SoD-rules. Our SoD-modeling provides a graphical, user-friendly approach to define SoD-rules. Furthermore, these rules can be checked for compliance with workflow specifications to prevent flaws during run time of the system. We tried to keep the complexity of the rule language (SSoDL) on a comprehensible level in order to provide a user-friendly graphical interface. Therefore, the expressiveness of SSoDL is smaller in comparison to other SoD-languages [1, 2, 3]. Nevertheless, our model catches most of the real-life rules and is intuitive to use.

Future research will encompass the following issues:

- Extension of SSoDL (AND, XOR, and OR connectors)
- Empirical test of the expressiveness and applicability of MASoD in a real-life setting
- Better support of import and export facilities to and from other workflow specifications such as the Workflow Process Definition Language proposed by the Workflow Management Coalition.
- The modeling and analysis of SoD rules is only a small part of a larger picture. The enforcement of SoD during run time, i.e. during the execution of workflow instances, is an important issue. Knorr [11] introduced an architecture for a workflow system based on Petri nets. We plan to use this architecture and enforce SoD based on SSoDL.

References

[1] Gail-Joon Ahn and Ravi Sandhu. The RSL99 Language for Role-based Separation of Duty Constraints. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 28-29 1999.

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

[3] Christoph J. Bussler. Policy Resolution in Workflow Management Systems. *Digital Technical Journal*, 6(4), 1995.

[4] Andrzej Cichocki, Abdelsalam Helal, Marek Rusinkiewicz, and Darrell Woelk. *Workflow and Process Automation — Concepts and Technology.* Kluwer Academic, 1998.

[5] David R. Clark and David R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings of the 1987 IEEE Symposium Security and Privacy*, pages 184–l94, Oakland, CA, 1987.

[6] Computer Security Institute. Issues and Trends — CSI/FBI Computer Crime and Security Survey. `http : //www . gocsi . com/summary. html`, 1999.

[7] Dimitrios Georgakopoulos, Mark Hornick, and Amith Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[8] Virgil D. Gligor, Serban I. Gavilla, and David Ferraiolo. On the Formal Definition of Separation-of-Duty Policies and their Composition. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, Oakland, CA, 1998.

[9] Wei-Kuang Huang and Vijayalakshmi Atluri. SecureFlow: A Secure Web-enabled Workflow Management System. In *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, pages 83–94, Fairfax, VA, October 28-29 1999.

[10] Kurt Jensen. *Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use, Volume 1.* EATCS Monographs on Theoretical Computer Science. Springer, 1992.

[11] Konstantin Knorr. WWW Workflows based on Petri Nets. In *Proceedings of the Ninth International Conference on Information Systems Development*, Kristiansand, Norway, August 2000.

[12] Konstantin Knorr and Harald Weidner. Analyzing Separation of Duties in Petri Net Workflows. In *Proceedings of the First International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security*, St. Petersburg, May 21-23 2001.

[13] Peter Langner, Christoph Schneider, and Joachim Wehler. Prozessmodellierung mit ereignisgesteuerten Prozessketten (EPKs) und Petri-Netzen. *Wirtschaftsinformatik,* 39(5):479–489, 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] Andreas Oberweis. *Modellierung und Ausführung von Workflows mit Petri-Netzen.* Teubner-Reihe Wirtschaftsinformatik. B.G. Teubner, 1996.

[17] *5th ACM Workshop on Role-Based Access Control,* Berlin, July 2000.

[18] Wolfgang Reisig. *Petri Nets—An Introduction.* EATCS Monographs on Theoretical Computer Science. Springer, 1985.

[19] P. Rittgen. Paving the Road to Business Process Automation. In Martin Bichler and Harald Mahrer, editors, *Proceedings of the 8th European Conference on Information Systems (ECIS),* volume 1, pages 313-319, Vienna, Jul. 2000.

[20] Ravi Sandhu. Separation of Duties in Computerized Information Systems. In *Proceedings of the IFIP WG11.3 Workshop on Database Security,* Halifax, UK, September 18-21 1990.

[21] August-Wilhelm Scheer. *Wirtschaftsinformatik. Studienausgabe. Referenzmodelle für industrielle Geschäftsprozesse.* Springer, 1998.

[22] Henrik Stormer, Konstantin Knorr, and Jan Eloff. A Model for Security in Agent-based Workflows. *Informatik • Informatique,* 6:24–29, Dec. 2000.

[23] Dirk Wodtke and Gerhard Weikum. A Formal Foundation for Distributed Workflow Execution Based on State Charts. In *Proceedings of the international Conference on Database Theory,* Greece, 1997.

[24] Workflow Management Coalition. *Interface 1: Process Definition Definition Interchange – Process Model.* Workflow Management Coalition, 1998. Document Number TC-1016-P.