PROCESS ANALYSIS AS FIRST STEP TOWARDS AUTOMATED BUSINESS SECURITY

Complete Research

Zahoransky, Richard, University of Freiburg, Freiburg, Germany, zahoransky@iig.uni-freiburg.de
Holderer, Julius, University of Freiburg, Freiburg, Germany, holderer@iig.uni-freiburg.de
Lange, Adrian, University of Freiburg, Freiburg, Germany, lange@iig.uni-freiburg.de
Brenig, Christian, University of Freiburg, Freiburg, Germany, brenig@iig.uni-freiburg.de

Abstract

Companies are promoting the implementation of automation in business processes to face the high level of competitive pressure induced in the course of globalization and interconnected environments. Additionally, a series of corporate scandals obliges companies to implement regulatory and legally binding rules. Required compliance analysis of those bindings is costly and time consuming, eating up the previously gained competitive advance. Our toolkit SWAT meets this contradiction by introducing automatic compliance analysis techniques. It provides a pattern based analysis approach with predefined patterns that contain security and compliance statements which are easily reusable. This approach allows subsequent analyses with minimal user interaction as a first step towards an automatic contemporary compliance analysis of workflows. Based on mathematically sound analysis techniques, our tool graphically shows violation examples for effortless interpretation of analysis results and identification of inconformities. In this paper we present the possibilities of our toolkit to analyse process models as well as process logs. We demonstrate its underlying techniques and concepts on a real world ordering process.

Keywords: Process Security, Compliance Analysis, Automated Workflow.

1 Automation and Compliance in Business Processes

The last decades have been characterized by a series of corporate scandals with significant socioeconomic impacts. The bank Société Générale lost nearly 5 billion Euro due to fraudulent transactions performed by one trader in 2008 [Clark and Jolly, 2008]. Another example represents the former energy company Enron, where fraudulent accounting led to one of the most severe bankruptcies in U.S. history at the beginning of the millennium [Hays and Driver, 2013]. These events caused by companies in a wide range of sectors, induced decision-makers to introduce a large number of regulatory and legally binding rules to protect the interests of various stakeholders. They are enacted to increase the transparency and traceability of business operations. One of these legal rules is the U.S. Sarbanes-Oxley Act, which obliges companies to comply with certain accounting requirements and responsibilities [Sarbanes and Oxley, 2002]. In addition to legal binding rules, companies may desire to comply with additional standards, norms and internal guidelines as further security rules, which necessitates them to adapt their organizational structure accordingly. Non-compliance can result in operational risks with considerable economic threats.

At the same time, modern companies operate in increasingly globalised and interconnected environments with a high level of competitive pressure. This calls for means to foster the productivity and efficiency in order to gain and ensure competitive advantages. The optimization of business processes provides

opportunities to achieve this overarching corporate objective. Here, the underlying concept of process orientation is, that every outcome of value for the company is the result of the proper orchestration of business activities [Dumas et al., 2013; Weske, 2012]. An adequate modeling and automation of repeating business processes offers potentials to enhance the efficiency of business operations. The resulting significance of information technology (IT) for companies is undisputed [Davenport and Short, 2003].

Besides realizing economic business objectives, IT also provides opportunities to increase the effectiveness and efficiency of the compliance realization. The detection of compliance violations in process instances can be tedious, time-consuming and as well costly for companies [Sykes, 2015]. Security audits in the course of compliance management are mostly performed manually, on a random basis and cover merely already concluded business activities [Mercuri, 2003]. Therefore, it is not surprising that prevailing monitoring systems lack the needed efficiency and effectiveness to deliver satisfying results in the light of automated business processes. This development necessitates automated compliance controls regarding repeating business activities to ensure credible and fast results that do not eat up the efficiency originally gained by automated business processes.

Our Security Workflow Analysis Toolkit (SWAT)[1] addresses this yet open issue by holistically analysing the security of business processes (or so-called workflows) by considering the control and information flow and resources before (ex-ante) and after (ex-post) their execution. Our toolkit is an artefact instantiation to answer the design-science research question if it is possible to automate security analyses in terms of compliance analysis. We identify single steps that contribute in reaching this goal. Our contribution is a working implementation on which we will further work towards the goal of automated workflow security analysis and optimization.

The remainder of this paper is structured as follows: In section 2 we give an overview of related work and introduce the research context followed by an overview of the building blocks of SWAT in section 3. The next section presents our example purchase order process, which is used throughout the rest of the paper. Section 5 describes the security analysis of this process with our toolkit for design-time flaws. In section 6 we present the technique used to check process logs for compliance. After presenting the experiment results in section 7, we finish with envisaged research and the conclusion.

2 Research Context and Related Work

In business process management, different security-related topics are investigated, for example compliance and conformance, the structure of workflows and resilience of business processes. To check the compliance of processes, logs that capture the executed activities of each process instance can be analysed [Accorsi and Stocker, 2012; Accorsi, Stocker, and Müller, 2013]. The structural behaviour of business processes is well studied in [Aalst, 2009; Adam, Atluri, and Huang, 1998; Lohmann et al., 2008]. However, they focus rather on functional properties than on security-related issues. Formal methods exist for the verification of access control [Armando and Ponta, 2010; Atluri, Chun, and Mazzoleni, 2001; Barletta, Ranise, and Viganò, 2009] and for delegation [Atluri and Warner, 2005; Crampton and Khambhammettu, 2008; Holderer, Accorsi, and Müller, 2015]. Further research has been done in the resilience context, e.g. on how to find and analyse business models that are still functional despite failures and can recover from stress [Wang and Li, 2007; Zahoransky, Brenig, and Koslowski, 2015; Zahoransky, Koslowski, and Accorsi, 2014]. Despite this research, there is little tool support available, which would transfer this knowledge into practical work. The ProM framework [Dongen et al., 2005] is a tool which focuses on the analysis of process logs. It supports the reconstruction of a process model based on logs as well as the direct analysis of the log using queries in linear temporal logic. Another tool called Disco [Günther and Rozinat, 2012] has also specialized on log analysis and brings user filters to pick out traces of exceptional behaviour. Depending on the used filters, security or performance relevant information can be obtained. However,

[1] Download and documentation: `http://iig-uni-freiburg.github.io/#swat`

neither Disco nor the ProM framework are specifically tailored for security-oriented analysis. Additionally, these tools neither provide the analysis of business processes during design-time nor support storing and retrieving of analyses. Every time an auditor conducts an analysis he has to start from scratch. We see this as a requirement for our toolkit and approach those drawbacks in order to provide a tool for academia as well as businesses with a security-oriented software platform. An automated security and compliance tool should provide functionality to reuse analyses on process logs which saves time and decreases the potentially error prone burden of redoing the same work again. With this in mind, there is also the necessity to provide means to construct analysis rules without deep knowledge of the underlying analysis method. This would allow a fast way of creating new analyses as well as testing and instantiating newly implemented rules. Table 1 outlines these requirements on the existing tools ProM and Disco.

	ProM	Disco
security-oriented	(✓)	✗
reusable analysis	✗	(✓)
process model support	(✓)	✗
process log support	✓	✓
analysis on models	✗	✗
abstract rule editor	(✓)	✗

Table 1. Tool comparison: ✓ applicable, (✓) partial applicable, ✗ not applicable.

Neither ProM nor Disco are explicitly security-oriented tools, as they only partly support security relevant analyses. With SWAT we aim to fill this gap. Our literature review and search for tool support shows that the analysis on log models is still missing. While ProM can load process models, it does not provide possibilities for security analysis based solely on these models. Disco does not support process models at all. However, this would be a step towards a holistic security and compliance analysis as it would allow for ex-ante analysis before the business process gets instantiated and the ex-post analysis, which searches for compliance problems after the process has finished.

3 Building Blocks

Figure 1 presents an overview of the different building blocks of SWAT. In general, SWAT uses workflow-related models and logs as input. Models also include contexts which define subjects (i.e. users or software agents), objects and activities.

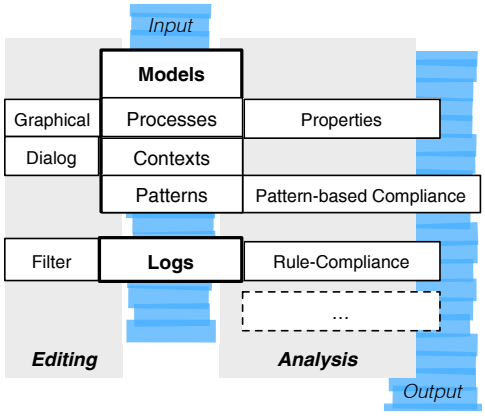


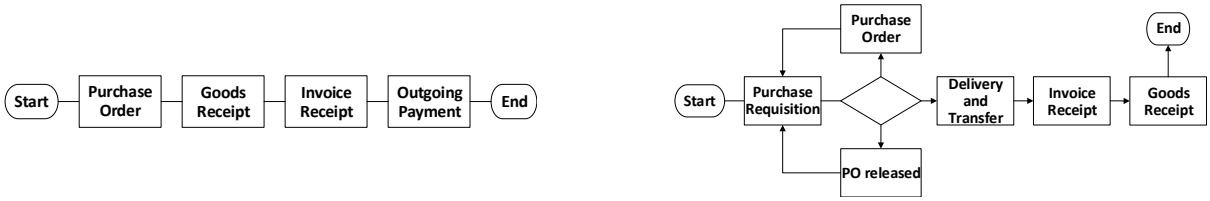
Figure 1. Building blocks of SWAT.

Logs contain executed process instances. Most of these inputs can directly be created and edited in SWAT. Based on these inputs, SWAT allows to perform different kinds of analyses.

In the following sections, the depicted components are explained in more detail based on a real world running example presented in the next section.

4 Purchase Order Process

In this paper, we use a running example to demonstrate the capabilities of SWAT to analyse process models and process logs. We have an anonymised real world order process log containing one year of execution. The intended blueprint of the process in Figure 2a is linear. It contains the purchase order, the receipt of goods and invoice as well as the payment.



(a) Blueprint of the order process. *(b) Extracted order process model.*

Figure 2. Example process of a purchase order.

We analysed the log and extracted a more detailed model for the order process (see Figure 2b). Here we see a less linear behaviour than projected in the blueprint of the order process. It starts with a purchase requisition. After this the purchase may need to be released or can be processed directly in some cases. It is not sure if this behaviour was intended. Depending on the subcontractor, firstly an order form needs to be updated and can then be sent. After placing the order, the incoming invoice is processed and the goods are receipted.

To illustrate the capabilities of SWAT, we assume the following additional constraints in this process:

- c_1 The purchase order may not be placed by the same person who sends the outgoing order (segregation of duty), so that a purchase cannot be forged.
- c_2 Goods must be receipted any time the process finishes.
- c_3 Before a purchase order may be placed, a purchase must be requisitioned.

The extracted process model, the process log itself and the constraints are used in the following section to explain how SWAT represents and analyses models and checks constraints over a process log.

5 Process Models

SWAT's capabilities include an ex-ante (before execution) and ex-post (after execution) analysis. It is able to analyse a workflow before instantiation as well as the actual behaviour of a workflow based on process logs. Depending on the analysis perspective different security and compliance analyses are possible. The models used for these two cases are discussed in the subsequent sections. Besides the process models itself, corresponding contexts of a process exist, which usually capture an access control model. Moreover, compliance patterns may describe specific properties of a process model that may not be violated or must be ensured.

5.1 Process Model Specification

Different specifications for process models exist in literature. Popular meta models to specify the control flow of a process are UML, event-driven process chain (EPC) [Hommes, 2004], BPMN [Damrau,

2010] and Petri nets [Petri, 1962]. We use Petri nets to model workflows because of their intuitively understandable graphical representation and their ability to capture the control flow as well as the state of a workflow. Additionally, Petri nets are mathematically specified, which yields the capacity to make reliable and certifiable statements. They consist of three elements: transitions, places and flow relations. The flow relation connects places to transitions and vice versa. Places may contain tokens that are consumed and produced by transitions according to the given flow relations and additional rules [Peterson, 1981]. Besides place/transition nets (P/T-nets) and coloured Petri nets (CPNs), SWAT is also able to work with information flow nets (IF-nets) [Stocker and Böhr, 2013], which are able to capture data and resource behaviour, too. Because of its popularity among process designers, SWAT also allows for transformation of BPMN models into P/T-nets.

5.2 Editing Process Models

To create Petri nets, SWAT comes with an editor named Wolfgang[2], fully compliant with the Petri net markup language standard (PNML) [Hillah et al., 2009]. It is also available as a standalone tool and supports the creation of P/T-nets and CPNs. SWAT additionally supports nets to IF-nets, which consider data elements and their access rights as well as the context in which the workflow is executed. Wolfgang can be used to prepare Petri nets for workflow-specific analysis or to simply model Petri nets for other purposes.

We implemented a BPMN import [Stackelberg et al., 2014] to speed up Petri net creation. The BPMN to Petri net transformation traverses the BPMN document and creates an initial Petri net on which the user can work. The transformation traverses the BPMN model in two steps. In the first run, all the components are transferred to corresponding Petri net elements (see Figure 3). For example, an AND-gateway is transferred to a Petri net element where one transition splits up into two places and activities are transferred to a place-transition pair. In the following steps, the single elements are connected according to the control flow of the BPMN model.

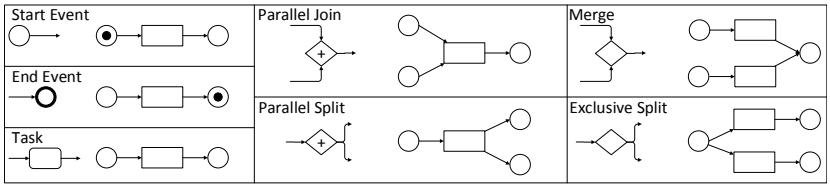


Figure 3. List of supported BPMN to Petri net transformations.

5.3 Analysing Process Model Properties

Besides the creation and editing of Petri nets, Wolfgang allows to explore the process behaviour manually with a so-called *token game*. This stepwise simulation of the Petri net's behaviour can help process designers to understand the control flow of their processes in more detail and may help to eliminate errors. Moreover, net properties that are required when using Petri nets to model business processes like validity, soundness and boundedness as well as the workflow-specific structural and soundness properties can be checked [Murata, 1989]. Figure 4 depicts the running example from Figure 2b as a Petri net.

5.4 Contexts

Contexts for workflow models may contain subjects, roles and their assignment to activities which represent a task in a workflow. The context base in SWAT contains the subjects, objects and activities

[2] Download and mentation: `http://iig-uni-freiburg.github.io/#wolfgang`

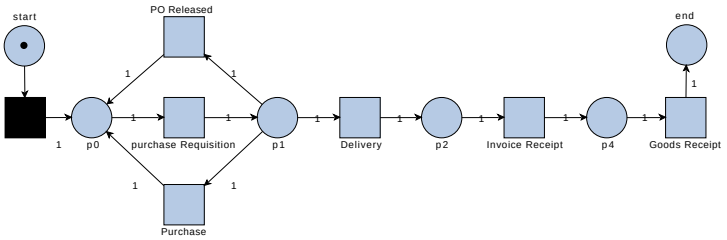


Figure 4. The running example as Petri net (modelled with Wolfgang).

(SOA-base) involved within workflow execution. The process context extends the SOA-base with data usage permissions (i.e. read, write, create or delete access on attributes which are connected to process activities) and access control models, used to decide which subjects can execute which activities, e.g. role-based access control (RBAC) or access control lists (ACL). IF-nets moreover require an analysis context which classifies the security level (based on [Bell and LaPadula, 1973]) of process activities and objects, the clearance level of subjects executing process activities, and subject descriptors, which assign subjects to process activities. These contexts build the foundation for further and more detailed analyses and are placed in the SEWOL[3] library.

5.5 Patterns

Patterns represent security requirements in SWAT. There are different pattern types. Atomic patterns consist of a single rule, whereas composite patterns combine multiple atomic ones. Order patterns not only describe the occurrence of events, but also their temporal order. SWAT also captures other information flow and organisational pattern types[4]. To ease the use of SWAT, we implemented a list of these security-related patterns so that they are directly usable to analyse a business process. A pattern editor helps the user to choose between the possible analyses, with each pattern holding a set of parameters which the user must enter. The pattern editor provides a list of possible entries for each parameter (see Figure 5) to eliminate potential faulty user input. Also, through this approach, a user can abstract from the underlying temporal logic language, which is completely hidden by SWAT.

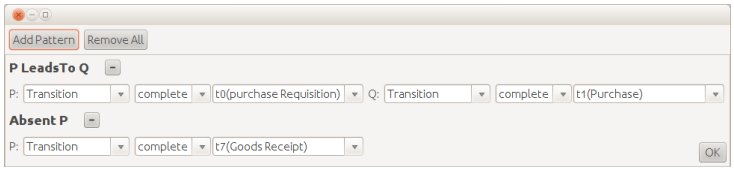


Figure 5. Rule editor with two enabled rules.

For analysing Petri nets, SWAT uses a model checking technique that allows for the efficient search for the compliance of rule-based models. In our terminology these rules correspond to the aforementioned patterns. We use the probabilistic model checker PRISM [Kwiatkowska, Norman, and Parker, 2002] which is able to analyse discrete-time Markov chains (DTMC) [Markov, 1971]. For such a model, PRISM can evaluate different properties, including probabilistic ones. For example: “What is the probability that a certain state is reached?”, or, if the model was in a certain state once: “Is it always true that eventually another state will follow?”.

[3] http://iig-uni-freiburg.github.io/#tools
[4] http://doku.telematik.uni-freiburg.de/general-pattern-description

5.6 Transformation to Markov Chains

A Petri net that is to be analysed by SWAT requires further transformation into a DTMC model. For this, a model mapping is implemented within SWAT that allows to transform P/T-nets, CPNs and IF-nets into Markov chains.

Markov chains are defined as a tuple (S, s_{init}, P, L) with S as the set of states, the initial state s_{init}, $P : S \times S \to [0,1]$ as the probability matrix from one state to another for all $s \in S$ and the labelling $L : S \to 2^{AP}$ as set of atomic propositions to each state $s \in S$. Let $N = (P, T, F, M_0)$ be a Petri net with a set of places P, a set of transitions T, a set of flow relations F and an initial marking M_0. The behaviour of the net is described by its possible fire sequences

$$\Phi = M_0 \xrightarrow{t_1} \ldots \xrightarrow{t_n} M_n \in L(N, M_0).$$

By this, $M_n(o) = 1$ is the terminal marking, which has at least one token in the output place o. Further, let $R(N, M_0)$ be the set of all reachable markings.

We built our DTMC model, which simulates the Petri net in the following: for each reachable marking $M_n \in R(N, M_0)$ we generate a state in the DTMC model. The overall set of states then is

$$S = \{(t, M) | M', M \in R(N, M_0), t \in \{t | M' \xrightarrow{t} M \in \Phi\}\},$$

where s_{init} is set to M_0. The probability matrix P is created the following way. For each pair (s_1, s_2) where a relation flow exists, the state transition probability is computed for three different cases, where $s_1 = (t_1, M_1)$ and $s_2 = (t_2, M_2)$:

- If $s_1 \neq s_2$ and $M_1(o) = 1$ the probability is 0 because this is a terminal state and the token is in the output place o.
- If $s_1 = s_2$ and $M_1(o) = 1$ the probability is set to 1 because the DTMC model requires that each state has state changes so we add a self loop to stay in the termination state.
- Otherwise the transition probability is set to $\frac{1}{|\{t | (N, M_1)[t\rangle\}|}$. For the given marking M_1 this is the transition probability that one of the enabled transitions in the underlying Petri net will fire from this state.

The generated model is transferred into PRISM as a so-called module. Modules are textual representations of the DTMC model and contain the used variables and the valid transitions between states. Figure 6 shows an example of a DTMC model in PRISM. In this example a never ending coin toss is modelled, with the variable x denoting the result of the coin toss.

```
1 dtmc
2 module M1
3 x:[0..1] init 0;
4 [a] x=0-> 0.5:(x'=0)+0.5:(x'=1);
5 [b] x=1-> 0.5:(x'=0)+0.5:(x'=1);
6 endmodule
```

Figure 6. PRISM DTMC module example: repeated coin tossing.

PRISM can check this model against expressions in linear and conditional temporal logic (LTL and CTL). Based on Petri nets SWAT can produce the corresponding textual representation of the DTMC model and holds the representation of a set of rules that can be applied in PRISM. Depending on the expression the model should be checked against, PRISM can generate witnesses or counterexamples which are displayed by SWAT without further user interaction or interpretation. The use of an external model checker ensures high performance and validity, as those programs are also used in the verification of hardware or software. When new model checking techniques evolve, SWAT can take part in this progress by switching to the next version of the model checker.

5.7 Analysis with Patterns

For compliance checking, SWAT includes a set of different patterns which are automatically transferred to the model checker query rules. In the following, we introduce some patterns to explain the function of SWAT in more detail.

The atomic pattern called *Absent P* examines for the absence of a specific state. This may be the occurrence of a single transition or a specific state of the underlying Petri net. Formally, the pattern's LTL formula in the PRISM language is $P =? [G(!(current\_trans = P))]$, which accepts every path, where P does not fire. The PRISM variable *current_trans* is a variable we include in the PRISM model which holds the last transition that would fire to reach the new state. The string "$P =?$" tells PRISM to return the probability of reaching such a state.

PRISM does also check for CTL-style properties, where it is able to generate counterexamples or witnesses. The pattern from above can be stated as CTL formula as $A[G\, current\_trans\ != P]$, where P is the transition for which the pattern should find its absence. It means: "all paths have to satisfy the property that P does not fire".

A further atomic pattern *Exists P* determines if a path exists in the Petri net where transition P is fired. The pattern in LTL-style is $F(P)$ and its representation in PRISM results in $P =? [F(current\_trans = P)]$.

The order pattern *P leads to Q* determines if there exists a path within the reachability graph from transition P to transition Q. In LTL-style the pattern is formalized as $G(P \rightarrow X(Q))$. In PRISM the pattern translates as $P =? [G((current\_trans = P) => (F(current\_trans = Q)))]$.

Another pattern allows the indication if two activities P and Q always appear exclusively in a process model (combined occurrence). For all allowed paths within the model this pattern determines true, when only one of the two activities appears in a single path. In temporal logic this pattern is formalized as $(\mathscr{F}(P) \rightarrow G(\neg Q)) \wedge (\mathscr{F}(Q) \rightarrow G(\neg P))$.

SWAT's process log analysis also utilizes temporal logic but uses a different model checker, which is explained in the following chapter.

6 Process Logs

Besides the ex-ante analysis of models, SWAT allows ex-post analyses on process logs by using the same patterns that translate to different formulae with the same meaning for the ex-ante analysis. Process logs contain information about executed instances of a process. There are different system environments in which a process or a workflow may be enacted, e.g. workflow management systems (WfMS), other process-aware information systems (PAIS), or enterprise resource planning systems (ERP). A log may contain different information, depending on the specific implementation. To abstract from such different implementations we firstly provide a mathematical definition of a process log.

A process log $w \in W$ is a multiset of traces $t \in T$. It stores information on one specific process, where each trace holds the information of one instance of this process. Each trace contains the sequence of activities $a \in A$ which took place in the regarded instance. Every activity is stored as an entry σ within the trace. Each entry holds additional information like the subject s or the timestamp τ on the specific activity. Formally a log w with n traces is defined as $w = (t_1, \ldots, t_n)$ with each trace containing a sequence of the corresponding events $t = (\sigma_1, \ldots, \sigma_m)$ where σ is a tuple (a, s, r, ψ, τ) with

a	activity of this event
s	subject who processed the activity
r	role of the subject
ψ	activity type e.g. start or completion
τ	timestamp of this event

For a better tool support, SWAT provides parsing and serializing functionalities, so that process logs can be imported from the two common log file formats XES [Günther and Verbeek, 2014; Verbeek et al., 2011] and MXML [Aalst et al., 2003]. Thus, most logs created with other tools can be imported in SWAT.

6.1 Filters

SWAT provides the possibility to filter and preprocess logs. Filtering a given set of real world process logs before analysis is essential. Otherwise, logs may contain incomplete process instances due to its selected timeframe and distort the analysis. For example, there may be a constraint requiring that "invoice receipt" must be followed by "goods receipt". However, a corresponding log may contain an instance capturing a process execution only until "invoice receipt" – although "goods receipt" was executed afterwards, but outside the selected timeframe. Analysis of such unfiltered instances therefore could result in erroneously suspected constraint violations. Additionally, we provide filters to only select cases with a minimum or a maximum number of events, or specific attribute values for subjects, roles, objects or activities.

6.2 Analysis with Rules

The information about executed instances in process logs can be used to analyse business processes after their execution (ex-post) to find compliance problems that are not evaluatable on the business process model alone. For that we use the analysis method SCIFF [Alberti et al., 2008]. It uses an underlying Prolog engine to analyse the compliance of logs with the given rules. The log entries are transferred to Prolog facts containing the log entries. Prolog rules are generated from the predefined patterns by SWAT. These facts and rules build up the knowledge base, which can be queried by Prolog goals. SCIFF analyses the compliance of rules per trace, which is why it builds the knowledge base for every trace all over again. This brings the advantage, that SCIFF tracewisely reports results for the compliance analysis and the knowledge bases for each iteration stay small.

As an example we show a trace of our running example. The events for our running example are the following tuples from a single trace:

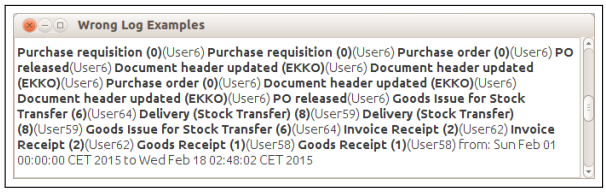
$$
\begin{aligned}
\sigma_1 &= (\text{“Purchase requisition”}, \text{“User25”}, \varnothing, \text{“start”}, 1417561200)\\
\sigma_2 &= (\text{“Purchase requisition”}, \text{“User25”}, \varnothing, \text{“complete”}, 1417561200)\\
\sigma_3 &= (\text{“Purchase order”}, \text{“User6”}, \varnothing, \text{“start”}, 1417561200)\\
\sigma_4 &= (\text{“PO released”}, \text{“User6”}, \varnothing, \text{“start”}, 1417561200)\\
\sigma_5 &= (\text{“Purchase order”}, \text{“User6”}, \varnothing, \text{“complete”}, 1417631613)\\
\sigma_6 &= (\text{“PO released”}, \text{“User6”}, \varnothing, \text{“complete”}, 1417631674)\\
\sigma_7 &= (\text{“Delivery and Transfer”}, \text{“User59”}, \varnothing, \text{“start”}, 1418598000)\\
\sigma_8 &= (\text{“Delivery and Transfer”}, \text{“User59”}, \varnothing, \text{“complete”}, 1418600968)\\
\sigma_9 &= (\text{“Invoice Receipt”}, \text{“User62”}, \varnothing, \text{“start”}, 1418770800)\\
\sigma_{10} &= (\text{“Invoice Receipt”}, \text{“User62”}, \varnothing, \text{“complete”}, 1418778095)\\
\sigma_{11} &= (\text{“Goods Receipt”}, \text{“User50”}, \varnothing, \text{“start”}, 1418943600)\\
\sigma_{12} &= (\text{“Goods Receipt”}, \text{“User50”}, \varnothing, \text{“complete”}, 1418966265)
\end{aligned}
$$

Translated to Prolog facts the events σ_1, σ_2 and σ_3 are represented as follows:

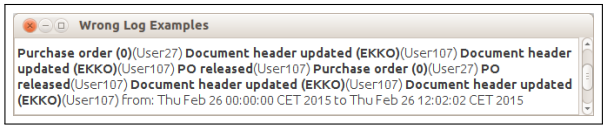
```
event('Purchase requisition','User25',null,start,1417561200).
event('Purchase requisition','User25',null,complete,1417561200).
event('Purchase order','User6',null,start,1417561200).
```

The role in this example is `null` since the underlying ERP did not log the role assignment. It does not affect the analysis as long as it is not evaluating the roles.

In the SCIFF language a rule has the form of an implication *head* $\leftarrow$ *body*, where *head* holds if *body* is true. This is translated to a Prolog rule to evaluate the given log entries as compliant to the given rule or not [Alberti et al., 2008]. For this, SCIFF uses tuProlog [Denti, Omicini, and Calegari, 2015], which is a Prolog engine implemented in Java. Thus it doesn't require the host to have a Prolog environment



(a) Counterexamples for segregation of duty (constraint c_1).



(b) Violation example of constraint c_3.

Figure 7. Process log constraint violations in SWAT.

explicitly installed. A rule in first-order logic that can be evaluated given the knowledge base might look like the following. Here the segregation of duty constraint c_1 from our running example is checked:

$$\neg\exists\sigma.(\text{execActivityOfTrace}(\psi_1, \mathit{PurchaseRequisition}, \sigma) \\ \wedge \text{execActivityOfTrace}(\psi_2, \mathit{PurchaseOrder}, \sigma) \wedge \psi_1 = \psi_2)$$

For argumentation, SCIFF tries to falsify the rule by a counterproof. This is accomplished by negating the Prolog clause. If the negated rule is successfully solved, a counterproof for the rule has been found and SCIFF reports that the rule is unfulfilled. The corresponding Prolog rule from above is the following:

```
rule_false :-(
  not((
    event('Purchase requisition',ASub,_,_,_),
    event('Purchase order',BSub,_,_,_),
    ASub \= BSub
  ))
).
```

The inner block of the rule `rule_false` evaluates for events with the specified names and assumes the executing subjects to be different. The remaining variables are not bounded and thus can be anonymous. Due to the outer negation only events where the same subjects executed the two activities are considered compliant. Since SCIFF creates the knowledge base for each trace, it does not need to be specified that both events belong to the same trace. For the example trace `rule_false` would not find a counterproof, why it would return true.

The following section contains both our ex-ante analysis on models and ex-post analysis on logs that are conducted on our example from section 4.

7 Experiment Results on the Purchase Order Process

We demonstrate the functionality of the compliance analysis for process models and logs of SWAT based on the introduced running example. We have evaluated the constraints c_1, c_2 and c_3 from section 4 on the log. It contains 11597 cases, on which we found that 33,91 % of the cases violated c_1. Constraint c_2 was violated in 45,07 % of the cases and 17,39 % of the instances violate constraint c_3 (of all cases where both activities existed). Single traces of the violating instances can be displayed in SWAT, including their timestamps and executing subjects. A screenshot of an example violation for constraint c_1 is shown in Figure 7a. For c_3, an example violation is provided in Figure 7b. In these examples, additional activities are present (e.g. “Document header update”), that are processed and logged by the system internally but do not participate in the actual business process.

We show and compare our results with Disco and ProM. The outcome is summarized in Table 2. ProM shows results identical to those of SWAT as it uses the same model checking approach. However, the results from Disco differ slightly for constraint c_3. Disco does not provide precise numbers for its filter results. Instead they are rounded to full percentage values. Constraint c_1 could not be analyzed with Disco.

Constraint	SWAT	ProM	Disco
c_1	33,91 % (3932 cases)	33,91 % (3932 cases)	not supported
c_2	45,07 % (5227 cases)	45,07 % (5227 cases)	45 % (no case count)
c_3	17,39 % (2017 cases)	17,39 % (2017 cases)	15 % (no case count)
c_3 (on model)	visual counterexample	not supported	not supported

Table 2. Found violations of SWAT compared to ProM and Disco.

With SWAT we also analysed if there exists a path within the model where a process order is delivered without preceding clearance (see also c_3). For our example from Figure 4 this is the case for some paths, as shown in the results of Figure 8. Here the individual activities that lead to the incident are marked by a red frame surrounding the single activities to represent a counterexample. Conformance checking based on process models is neither supported by ProM nor Disco.

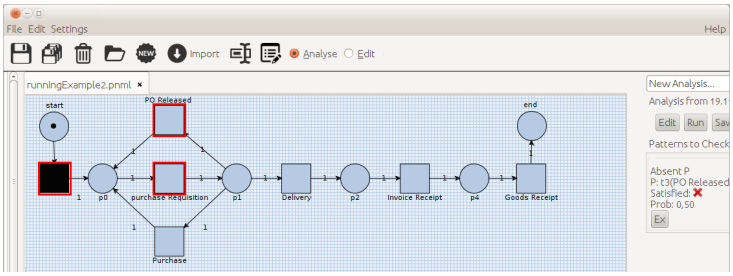


Figure 8. Counterexample for a missing activity.

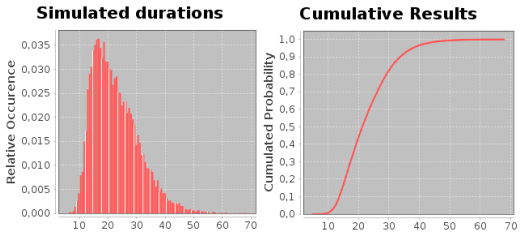
In our examples we have successfully found different security-related problems in the process model itself as well as in the logs. Although the example is small, the same patterns can be applied to larger models and logs. This is supported by the use of the external model checker PRISM which can cope even with large models. For logs, the complexity remains in the linear space as traces are examined one by one. This means that only small parts of the log must be held in memory for analysis.

8 Envisaged Research

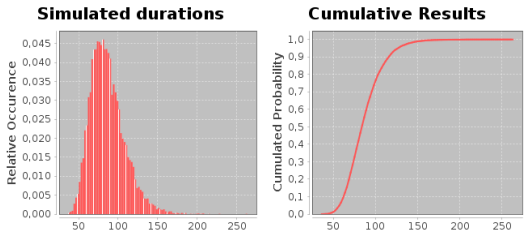
With SWAT we are currently working on new techniques and methods that cover security in a broader sense. We want to address questions of availability, automated delegation and ex-post analysis over multiple instances of a workflow.

Considering availability as part of security, we plan to include resilience measurement and analysis based on process simulation. The results of the simulation will show the influence of missing resources for a business process model and also provide means to assign remaining resources to guarantee the best achievable outcome. Figures 9a and 9b show simulation results for two processes. For each process the graphs display the simulation results as probability distribution on the required time to finish and as cumulated results. Our Monte Carlo simulation is done with Resource Timed Petri nets (RTPN), an extension to Petri nets on which we are currently working. It adds time and resource notations to the classical Petri net similar to [Ramchandani, 1974] but with arbitrary timing functions.

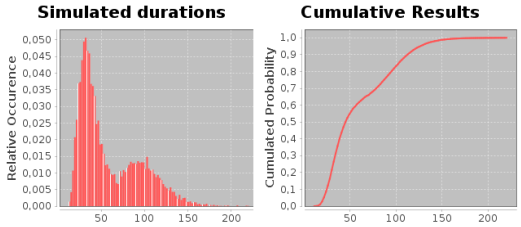
SWAT's simulation technique also allows the simultaneous simulation of parallel processes with shared resources. Figures 9c and 9d show simulation results with two processes in parallel. They share resources and thus they interfere with each other. In this example the first process (Figures 9a and 9c) is highly influenced when its resources get occupied by the other process. Without interference it almost certainly ended after one hour. In the second case its probability to end after one hour dropped to 62 %. Process models can be linked to process logs to extract real world data regarding time and resource consumption of activities and use it to drive the simulation. In addition we plan to differentiate the time behaviour of



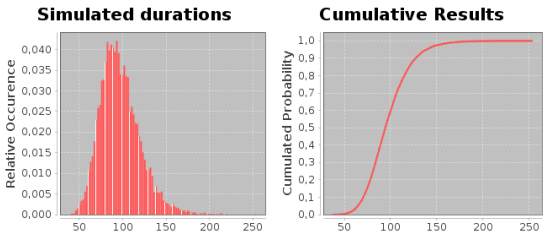
(a) Isolated Simulation results for process $P1$.



(b) Isolated Simulation results for process $P2$.



(c) Simultaneous simulation results for process $P1$.



(d) Simultaneous simulation results for process $P2$.

Figure 9. SWAT process simulation of two processes $P1$ and $P2$ (time axis in minutes).

activities according to the actual used resources. With the obtained information the simulation can be used to evaluate different scenarios and how well the given model will perform under various circumstances. The acquired knowledge from the simulation brings up possibilities to optimally plan the allocation of resources within a business process architecture. We are working on an approach that will compute strategies under a given scenario on how to schedule remaining resources most efficiently. This decision support will be able to allocate or take resources away from less important or interruptible processes and provide them to critical instances which would not be able to tolerate discontinuities. In the example from Figure 9, process $P2$ could be slowed down in order to speed up process $P1$. With such an approach, it will be feasible to continuously adapt a business process architecture on changing environments, if for example resources get lost or new processes get integrated. To model the importance of single processes we introduce weights which resemble how vital this process is to the overall performance of the business process architecture.

In our running example in section 4 we have shown the ex-post analysis of a business process. We have checked the constraints on single traces. In many cases however, frauds or compliance violations can only be detected when information from different instances are involved and considered. Such a constraint over multiple instances could limit the number of instances with the same subject assignments for some activities in a specified interval or could limit the total order value of a single employee per day. Additionally, constraints are not limited to refer to instances of one process, but also constraints involving different processes are possible, e.g. limiting the total number of process executions per day to prevent employees from over-working. We are currently creating a model checker which supplements the SCIFF functionalities by these inter-instance and inter-process constraints and can also consider object values, access control models, relational connections between subjects and aggregates on log traces for forensic analyses of logs. In the end this model checker will replace SCIFF as log analysis tool.

When working with large process logs which were created over a long time, it can happen that the underlying model's procedures change. This leads to inconsistent activity names and sequences, which must be considered when defining constraints. Therefore, we're working on trace clustering algorithms, which group similar traces considering different parameters like activity order, task duration and object usage. Constraints can be defined on groups of similar traces. On this basis, we will give the opportunity to look for behavioural deviations in logs, which distinguish themselves by not fitting in the groups of similar process execution cases.

Moreover, non-resilient allocation of human resources and blocking access control policies may result in processes that cannot be completely executed during enactment and cause exceptional situations. Here, our goal is to develop an approach that catters for the detection of such obstructions and policywise sound workarounds that allow their execution. By that we want to face known shortcomings of current methods to resolve such situations. On the one hand, policy overriding (e.g. Break-Glass [Petritsch, 2014]) mainly ignores given policies and requires costly auditing afterwards. On the other hand, classic delegation demands the delegator to be available and involves the danger of collusion. Therefore, we are currently investigating how an automated delegation based on security and resilience measures may look like. Specifically, besides the process model itself, the contexts, access control models and corresponding constraints are taken into account. Moreover, logs provide fact-based information on subjects involved in process executions, too. Based on these ingredients, methods to determine the next best delegate are developed and integrated into SWAT to provide more specific evidence on how a reliable solution in an organizational software system should be constituted.

9 Conclusion

Our contribution is the software artefact SWAT, a toolkit for streamlined security analysis based on mathematically sound methods for process logs and models, which is a unique and new approach as it supports ex-ante and ex-post analyses. As discussed, SWAT consolidates features from different toolkits with additional capabilities. It fills the gaps described in Table 1 and uses proven and reliable techniques, which allow to include new analyses using temporal logic instead of programming rules from scratch. New analysis rules are integrable into the available rule editor. SWAT can store and reuse analyses to quicken recurring compliance checks. For ease-of-use, we incorporated a set of rules that can be parametrized by the rule editor and checked against models and logs. This approach ensures that users can apply different analyses without needing deeper knowledge of temporal logic, as it would be required for example when using ProM. Through this abstraction level SWAT helps to overcome the hurdle for business process analysis and to decrease the potential of accidental incorrect operation. With SWAT a new approach for compliance checks is available that can be carried out on process models instead of process logs. This allows a holistic security and compliance analysis of the processes itself as well as their actual behavior. Results are displayed visually to understand conflicts or points of violation within the process model and the logs. We have the opinion that this tool will help scientists, process auditors and people responsible to find and create new compliance properties and other rules that analyse different aspects of business process specifications and process logs.

The analyses which we have carried out on this process were stored, including their parameters. This mechanism enables that future analyses on the same process can be redone without further revision to save time and money on future compliance audits which greatly increases efficiency and thus helps companies to keep their competitive advantage. After the first initial set-up of a workflow audit, SWAT enables fully automatic compliance and security analysis from now on without additional expert knowledge. This actively aids the concept of efficient, automated business processes since their productivity is no longer diminished by tedious, time consuming and costly compliance audits.

SWAT is a starting point for further analyses on processes. In future work, we plan to implement additional features covering a broader sense of security to enable and improve business processes to sustain their functionality under changing and turbulent environments and to investigate the interaction between multiple, parallel processes.

References

Aalst, W. M. van der (2009). "Workflow model analysis." In: *Encyclopedia of Database Systems*. Springer, pp. 3551–3551.

Aalst, W. M. Van der, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. Weijters (2003). "Workflow mining: A Survey of Issues and Approaches." *Data & knowledge engineering* 47 (2), 237–267.

Accorsi, R. and T. Stocker (2012). "On the exploitation of process mining for security audits: the conformance checking case." In: *Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012*. Ed. by S. Ossowski and P. Lecca. ACM, pp. 1709–1716. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2232051. URL: http://doi.acm.org/10.1145/2245276.2232051.

Accorsi, R., T. Stocker, and G. Müller (2013). "On the exploitation of process mining for security audits: the process discovery case." In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013*. Ed. by S. Y. Shin and J. C. Maldonado. ACM, pp. 1462–1468. ISBN: 978-1-4503-1656-9. DOI: 10.1145/2480362.2480634. URL: http://doi.acm.org/10.1145/2480362.2480634.

Adam, N. R., V. Atluri, and W.-K. Huang (1998). "Modeling and analysis of workflows using Petri nets." *Journal of Intelligent Information Systems* 10 (2), 131–158.

Alberti, M., F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni (2008). "Verifiable agent interaction in abductive logic programming: the SCIFF framework." *ACM Transactions on Computational Logic (TOCL)* 9 (4), 29.

Armando, A. and S. E. Ponta (2010). "Model checking of security-sensitive business processes." In: *Formal Aspects in Security and Trust*. Springer, pp. 66–80.

Atluri, V., S. Chun, and P. Mazzoleni (2001). "A Chinese wall security model for decentralized workflow systems." In: *Proceedings of the 8th ACM conference on Computer and Communications Security*. ACM, pp. 48–57.

Atluri, V. and J. Warner (2005). "Supporting conditional delegation in secure workflow management systems." In: *Proceedings of the tenth ACM symposium on Access control models and technologies*. ACM, pp. 49–58.

Barletta, M., S. Ranise, and L. Viganò (2009). "Verifying the Interplay of Authorization Policies and Workflow in Service-Oriented Architectures." *arXiv preprint arXiv:0906.4570*.

Bell, D. E. and L. J. LaPadula (1973). *Secure computer systems: Mathematical foundations*. Tech. rep. DTIC Document.

Clark, N. and D. Jolly (2008). *Societe Generale loses $7 billion in trading fraud*. (Visited on 11/20/2015).

Crampton, J. and H. Khambhammettu (2008). "On delegation and workflow execution models." In: *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, pp. 2137–2144.

Damrau, J. (2010). *The Process: Business Process Modeling Using BPMN*.

Davenport, T. H. and J. E. Short (2003). "Information technology and business process redesign." *Operations management: critical perspectives on business and management* 1, 97.

Denti, E., A. Omicini, and R. Calegari (2015). "tuProlog: Making Prolog Ubiquitous." *Book Reviews* 2015 (05).

Dongen, B. F. van, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. Van Der Aalst (2005). "The ProM framework: A new era in process mining tool support." In: *Applications and Theory of Petri Nets 2005*. Springer, pp. 444–454.

Dumas, M., M. La Rosa, J. Mendling, and H. A. Reijers (2013). *Fundamentals of business process management*. Springer.

Günther, C. W. and A. Rozinat (2012). "Disco: Discover Your Processes." *BPM (Demos)* 940, 40–44.

Günther, C. W. and H. Verbeek (2014). "XES-standard definition."

Hays, K. and A. Driver (2013). *Former Enron CEO Skilling's sentence cut to 14 years*. (Visited on 11/20/2015).

Hillah, L., E. Kindler, F. Kordon, L. Petrucci, and N. Treves (2009). "A primer on the Petri Net Markup Language and ISO/IEC 15909-2." *Petri Net Newsletter* 76, 9–28.

Holderer, J., R. Accorsi, and G. Müller (2015). "When four-eyes become too much: a survey on the interplay of authorization constraints and workflow resilience." In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*. Ed. by R. L. Wainwright, J. M. Corchado, A. Bechini, and J. Hong. ACM, pp. 1245–1248. ISBN: 978-1-4503-3196-8. DOI: 10.1145/2695664.2699497. URL: http://doi.acm.org/10.1145/2695664.2699497.

Hommes, L. J. (2004). *The evaluation of business process modeling techniques*. TU Delft, Delft University of Technology.

Kwiatkowska, M., G. Norman, and D. Parker (2002). "PRISM: Probabilistic symbolic model checker." In: *Computer performance evaluation: modelling techniques and tools*. Springer, pp. 200–204.

Lohmann, N., P. Massuthe, C. Stahl, and D. Weinberg (2008). "Analyzing interacting WS-BPEL processes using flexible model generation." *Data & Knowledge Engineering* 64 (1), 38–54.

Markov, A. (1971). "Extension of the limit theorems of probability theory to a sum of variables connected in a chain."

Mercuri, R. T. (2003). "On auditing audit trails." *Communications of the ACM* 46 (1), 17–20.

Murata, T. (1989). "Petri nets: Properties, analysis and applications." *Proceedings of the IEEE* 77 (4), 541–580.

Peterson, J. L. (1981). "Petri net theory and the modeling of systems."

Petri, C. A. (1962). *Kommunikation mit Automaten*. Dissertation, Schriften des IIM 2. Bonn: Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn.

Petritsch, H. (2014). *Break-Glass - Handling Exceptional Situations in Access Control*. Springer. ISBN: 978-3-658-07364-0. DOI: 10.1007/978-3-658-07365-7. URL: http://dx.doi.org/10.1007/978-3-658-07365-7.

Ramchandani, C. (1974). "Analysis of asynchronous concurrent systems by timed Petri nets."

Sarbanes, P. and M. Oxley (2002). "Be it enacted by the Senate and House of Representatives of the United States of America in Congress assembled, Sarbanes-Oxley Act of 2002." *Public Law* 107, 204.

Stackelberg, S. von, S. Putze, J. Mülle, and K. Böhm (2014). "Detecting Data-Flow Errors in BPMN 2.0." *Open Journal of Information Systems (OJIS)* 1 (2), 1–19.

Stocker, T. and F. Böhr (2013). "IF-Net: A Meta-Model for Security-Oriented Process Specification." In: *Security and Trust Management*. Springer, pp. 191–206.

Sykes, M. (2015). *The Value of Time: Solving the Cost of Compliance*. URL: http://marketsmedia.com/the-value-of-time-solving-the-cost-of-compliance/ (visited on 11/27/2015).

Verbeek, H., J. C. Buijs, B. F. Van Dongen, and W. M. Van Der Aalst (2011). "XES, XESame, and ProM 6." In: *Information Systems Evolution*. Springer, pp. 60–75.

Wang, Q. and N. Li (2007). "Satisfiability and resiliency in workflow systems." In: *Computer Security–ESORICS 2007*. Springer, pp. 90–105.

Weske, M. (2012). *Business process management: concepts, languages, architectures*. Springer Science & Business Media.

Zahoransky, R. M., C. Brenig, and T. Koslowski (2015). "Towards a process-centered resilience framework." In: *ARES 2015 - 10th International Conference on Availability, Reliability and Security - Workshop FARES*.

Zahoransky, R. M., T. Koslowski, and R. Accorsi (2014). "Toward Resilience Assessment in Business Process Architectures." In: *Computer Safety, Reliability, and Security*. Springer, pp. 360–370.