Modeling wizard for confidential business processes

Andreas Lehmann and Niels Lohmann

Universität Rostock, Institut für Informatik, 18051 Rostock, Germany
andreas.lehmann@uni-rostock.de niels.lohmann@uni-rostock.de

Abstract. One driver of business process management is the opportunity to reduce costs by outsourcing certain tasks to third-party organizations. At the same time, it is undesirable that delicate information (e. g., trade secrets) "leak" to the involved third parties, be it for legal or economic reasons. The absence of such leaks — called *noninterference* — can be checked automatically. Such a check requires an assignment of each task of the business process as either confidential or public. Drawbacks of this method are that (1) this assignment of every task is cumbersome, (2) an unsuccessful check requires a corrected confidentiality assignment although (3) the diagnosis and correction of information leaks is a nontrivial task. This paper presents a modeling prototype that integrates the noninterference check into the early design phase of an interorganizational business process. It not only allows for instant feedback on confidentiality assignments, but also for an automated completion of partial assignments toward guaranteed noninterference.

1 Introduction

One advantage of systematic business process management is the ability to outsource individual tasks or whole subprocesses to third parties. Beside cost reduction, this allows the business owner to concentrate on the core business (specialization), to ensure that certain regulations are met (standardization), or to exclude liability. This trend has led to the cloud computing paradigms such as SaaS, IaaS, or PaaS.

The distributed execution of a business process adds new challenges, as a business process is usually a very sensitive asset of a company. In this setting, *confidentiality* of sensible information is paramount. Though the interplay with third parties can be regulated by contracts, a business owner should never entirely trust other agents. This paper focuses on confidentiality and studies *noninterference* (i. e., the absence of information leaks) as central correctness criterion. Noninterference guarantees that third parties cannot deduce information on the execution of confidential tasks by observing the execution of outsourced tasks. It is important to note that *this problem cannot be tackled by security instruments such as encryption or access control*. Noninterference needs to be checked at design time using a suitable business process model, because detecting information leaks at runtime would be too late.

Recent literature [1, 2, 3] introduced noninterference checks and reported promising numbers on the required resources for those checks. The prerequisite of the checks is a fully annotated model; that is, each task is annotated whether it is confidential. This has two major drawbacks. First, it is a tedious job for the modeler to make this decision for every task. Though it may be obvious for a few tasks (e. g., business decisions), it is rather arbitrary for a lot of other tasks (e. g., pure routing constructs). At the same time, the modeler has to face the dilemma that outsourcing too many tasks brings the danger of information leaks while outsourcing too few tasks raises the overall costs. Second, if the check is unsuccessful, the detected information leakage must be understood, fixed, and rechecked. This introduces another loop to the modeling process. Hence, an integrated, continuous check similar to a wizard or a spell checker with instant error detection and correction proposals would be desirable to support the modeler.

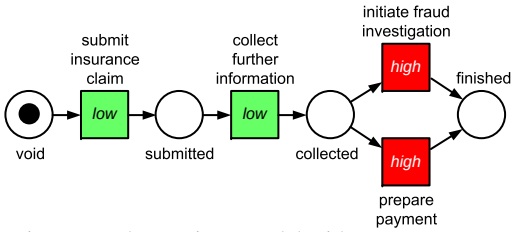


Fig. 1. Running example: Petri net model with a secure outsource strategy.

This paper makes three contributions toward this goal. First, we support the modeling of confidential business processes by instant feedback with respect to noninterference. Second, we reduce the modeling effort by completing partial confidentiality assignments while guaranteeing noninterference. We achieve this by presenting a compact representation of all valid assignments. Third, we sketch the applicability of these techniques in the early design phase of business process modeling.

The rest of this paper is organized as follows. The next section provides an overview of modeling and analyzing noninterference using business process models. Section 3 discusses how all valid assignments of a business process model can be characterized efficiently in terms of runtime and memory. We also provide experimental results. The completion of partial assignments and the integration into a business process modeling tool is presented in Sect. 4. Section 5 is dedicated to related work before Sect. 6 concludes the paper.

2 Background

2.1 Informal introduction

Figure 1 depicts a business process, expressed as a Petri net, we shall use as running example for the rest of the paper. In this process, an insurance claim is handled in the following way: After the claim is submitted to the insurance company, further details are collected. Based on these collected facts, a decision is made to either initiate a fraud investigation due to suspicious information or to prepare the payment. In either case, the process is completed. Of course, this is only a toy example, but it can be seen as a high level view on a larger business process in which the fraud investigation itself consist of several subprocesses. Considering outsourcing within this process, basically any task is a possible candidate to reduce the insurance company's costs — in particular as this process will be executed very often. A budget analyst may therefore suggest to outsource all tasks in order to keep the process as cheap as possible, but due to the impact of a fraud investigation (despite the question whether it is justified), the insurance company may consider especially this task to be *confidential*. In contrast to the fraud investigation the submit task is supposed to be outsourced to a call center. Confidential tasks must not be outsourced. A different approach would be to perform all tasks within the company, which is indeed secure according to confidentiality, but at the same time very expensive. So neither idea (outsource everything or outsource nothing) is suitable for the management. Considering our running example with four tasks, there exist theoretically $2^4 = 16$ possible outsourcing strategies. Hence, the question raises, which one to choose with respect to costs and confidentiality.

Not all of the possible outsourcing strategies are secure with respect to confidentiality. Examples such as outsourcing everything may be obviously insecure, but let us consider a tradeoff where only the fraud investigation is performed in-house, whereas all remaining tasks are outsourced. Outsourcing the mutually exclusive tasks prepare payment while

keeping the task initiate fraud investigation confidential is also not secure: Assume that all participants know the whole process (due a service agreement) and are aware of all outsourced tasks. Note that the second assumption is not unrealistic, because also outsourcing each task to a different company cannot ensure that those companies do not reallocate their tasks to another (for instance the same) company or collaborate together. So in case a participant can observe the occurrence of the task prepare payment, he can deduce that the task initiate fraud investigation was not performed, and vice versa. Therefore, it is not sufficient to only keep initiate fraud investigation confidential, but the task prepare payment needs to be confidential as well to obtain a secure outsourcing strategy. This type of example can be called a *conflict relation* between two tasks (cf. the red colored tasks in Fig. 1).

Another example of such a dependency is a *causal relation* between two tasks. Consider the task submit insurance claim to be confidential in an outsourcing strategy where all other tasks are outsourced. A participant can deduce that the task submit insurance claim has been executed after he observes the execution of task collect information.

Note that *both conclusions can be deduced from the fact that information flow between third parties and the insurance company — regardless of the use of a perfect encryption and a perfect access control.* Therefore, traditionally security mechanism are not suitable to express and ensure the mentioned confidentiality. To reason about those dependencies of confidential and public tasks, noninterference is a suitable property. In the rest of this section, we shall provide the formal background of our formalisms and sketch the verification of noninterference for business processes.

2.2 Noninterference

Noninterference is a property in terms of *information flow control*, which provides a powerful abstraction to certify confidentiality and integrity properties [4]. In the context of this paper, we concentrate on confidentiality, but integrity can be seen as a dual problem to confidentiality. Therefore a business process model is divided into two security domains (*high* for confidential, *low* for public). In the previous section, we used the term outsourcing as use case for the public domain. A flow happens whenever information meant to remain in the confidential domain *leaks* to the public domain. A business process model is assumed secure if it enforces noninterference; that is, the actions in the *high* domain do not produce an observable effect (*interference*) in the *low* domain. Interferences thus open up the possibility to deduce information about confidential behavior. If exploited, these *covert channels* violate the security requirements [5].

We shall first present how noninterference can be formalized using Petri nets. Based on this, we shall sketch the noninterference analysis in terms of Petri nets. We employ Petri nets as they combine a graphical notation with a formal semantics. Furthermore, mappings from common modeling languages, such as WS-BPEL, BPMN, and EPC, exist [6]. Finally, Petri nets are extensively used to reason about the correctness of business processes which yield to the availability of mature tool support. We consider *safe* Petri nets for the rest of the paper; that is, each place holds at most one token on all reachable markings. This eases our definitions, but introduces no restriction, because in the domain of business processes we focus on sound process models, which by definition are bounded and therefore can be expressed by safe Petri nets as well.

Definition 1 (Petri net). *A* Petri net $N = [P, T, F, m_0]$ *consists of two finite and disjoint sets P of* places *and T of* transitions*, a* flow relation $F \subseteq (P \times T) \cup (T \times P)$*, and an* initial marking m_0*. A* marking $m \subseteq P$ *represents a state of the Petri net and is visualized as a distribution of tokens on the places. Let* $x \in (P \cup T)$*. The* preset *of x is the set*

${}^\bullet x = \{y \mid [y, x] \in F\}$*, the* postset *of* x *is* $x^\bullet = \{y \mid [x, y] \in F\}$*. Transition* t *is* enabled *in marking* m *iff,* ${}^\bullet t \subseteq m$*. An enabled transition* t *can* fire*, transforming* m *into the new marking* m' *with* $m' = (m \setminus {}^\bullet t) \cup t^\bullet$*. The firing of one transition is denoted as* $m \xrightarrow{t} m'$ *and a sequence* $\sigma \in T^*$ *of transitions transforming* m *to* m' *is denoted as* $m \xrightarrow{\sigma} m'$*. A marking* m' *is* reachable *from* m*, if* $m \xrightarrow{*} m'$ *(with* $*$ *for an arbitrary sequence).*

To reason about noninterference, we need an extension to the Petri net definition. Therefore, transitions are partitioned into the two security domains *high* and *low* necessary for the noninterference property. We call such an extended Petri net a *labeled Petri net*, because transitions are *labeled* with a security domain. We shall use the term "Petri net" from now on as short form for "labeled Petri net".

Definition 2 (Labeled Petri net). *A* labeled Petri net $N = [P, L, H, F, m_0]$ *is a Petri net* $[P, L \cup H, F, m_0]$ *with* $L \cap H = \emptyset$*.*

Running example. In Fig. 1 the *high* security domain represents tasks that should remain confidential (red colored) and the *low* security domain represents tasks that are supposed to be outsourced (green colored).

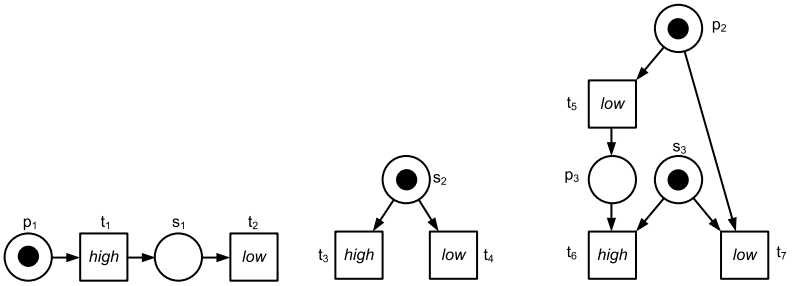
Earlier, we gave an intuition of noninterference in our setting and mentioned already two possible kinds of problems: the conflict case and the causal case. In the conflict case, there is a mutual exclusion between two tasks which different security domains. As tasks are represented by transitions and mutually exclusion is expressed as a common preplace, this case can be straightforwardly expressed in terms of Petri nets. The causal case can be expressed similarly: a transition with security domain *high* produces a token on a place which is the preset of another task in the *low* security domain. Busi and Gorrieri [2] defined those two types as follows:

Definition 3 (Causal and conflict places s). *Let* $N = [P, L, H, F, m_0]$ *be a labeled Petri net. The place* s *is a* potential causal place *if there exists a transition* $l \in s^\bullet \cap L$ *and a transition* $h \in {}^\bullet s \cap H$*. A potential causal place* s *is an* active causal place *if, additionally, a marking* m *is reachable such that a transition sequence* σ *exists with* $m \xrightarrow{h\sigma l} m'$ *and* $s \notin t^\bullet$ *for all* $t \in \sigma$*.*

The place s *is a* potential conflict place *there exists a transition* $l \in s^\bullet \cap L$ *and a transition* $h \in s^\bullet \cap H$*. A potential conflict place is an* active conflict place *if, additionally, a marking* m *is reachable such that a transition sequence* σ *exists with* $m \xrightarrow{h} m'$*,* $m \xrightarrow{\sigma l} m''$ *and* $s \notin t^\bullet$ *for all* $t \in \sigma$*.*

Figure 2 illustrates the two types of possible interference places, the causal case (a) and the conflict case (b). In the causal case, the *low* labeled transition t_2 can only fire after the *high* labeled transition t_1 has fired, so the fact that t_1 (and its corresponding confidential task) has fired is leaked. In the conflict case, the two transitions t_3 and t_4 are mutually exclusive, which means that from firing of the *low* labeled transition t_4 one may deduce that the *high* labeled transition t_3 has not fired.

Based on Defs. 2 and 3 the analysis of noninterference is carried out with *Positive place-based noninterference* (PBNI+) [2]. The idea is that causal and conflict places encode the two possible leaks from the *high* to the *low* domain. A Petri net is called *secure* in terms of noninterference if each of its places is neither an active causal nor an active conflict place. One may think PBNI+ is a structural property which can be decided on the net structure. Whereas potential places are indeed identified by the net *structure*, the decision whether a potential place is an active place depends on the *behavior* of the net (viz. the presence of a transition sequence σ) [2]. Figure 2(c) depicts an example in



(a) active causal place (b) active conflict place (c) potential conflict place

Fig. 2. Patterns for potential causal and conflict places s

which the place s_3 is a potential but not an active conflict place; that is, the structural conflict between t_6 and t_7 is not decided by s_3, but by p_2. How active places can be identified is part of the following section.

2.3 Verification of noninterference

According to Def. 3, each active place can be described as a triple $[s, h, l]$ of a place s, a *high* labeled transition h and a *low* labeled transition l. More than one *high* or *low* labeled transition in the preset or postset of s yield to more triples with s. A place s is active if at least one triple is active according to Def. 3. In earlier work [1], we showed that active triples can be identified by reachability checks.

Running example (cont.). Considering the business process of our running example without any given confidentiality requirements, one would identify two potential places. The place collected is a potential causal and a potential conflict one, whereas place submitted is only a potential causal one. The corresponding triples are: (1) [collected, initiate, prepare], (2) [collected, prepare, initiate] (both with potential conflict place collected), (3) [collected, collect, initiate], (4) [collected, collect, prepare] (both with potential causal place collected), and (5) [submitted, submit, collect] (with potential causal place submitted). The introduced order is used in the continuations of the running exmaple.

3 Characterization of all valid assignments

3.1 Motivation

As mentioned in the introduction, even a quick PBNI+ check has several drawbacks: First, it requires a complete confidentiality assignment; that is, each transition has to be labeled with either *high* or *low*. This means that the modeler needs to make a manual decision for each transition whether the modeled task is confidential or public. Such choices can be very arbitrary, yet still affect overall noninterference. That said, if an information leak was detected, the assignment has to be manually corrected and re-checked. This results in a constant interruption of the design workflow of a business process.

To this end, we present in this section the first contribution of this paper: a characterization of all valid confidentiality assignments given a partial (or even empty) assignment. This allows to apply our approach in the very early stage of modeling and also supports the assignment of existing, but not labeled, business process models. Such a characterization has several benefits:

1. In case the set of valid assignments is empty, we can provide *immediate feedback* to the modeler. This immediate feedback is important as it can be correlated to the most recent action of the modeler, whereas later feedback would require a search for the transition whose labeling introduced an information leak.
2. If in turn the set of valid assignments is not empty, it can be used to *efficiently check* refined assignments, because any further assignment of transitions only restricts the set and a complete recalculation is not required.
3. Finally, a complete characterization allows to reason about *implicit information*. If, for instance, a certain transition is assigned to *high* in all valid assignments, then this should be immediately presented to the modeler.

Whereas previous work [1] showed that a noninterference check is typically very fast, a naive enumeration of all possible assignment has two major downsides:

1. The number of assignments grows exponentially in the number of transitions. Even with an average checking time of 24 milliseconds [1] the exponential blowup makes this enumeration not applicable to industrial models with hundreds of transitions.
2. Even if we can determine the valid assignments, an explicit representation is infeasible due to the same exponential blowup. However, only a complete set of all valid assignments gives the modeler maximal freedom to come up with an optimal outsourcing plan.

The rest of this section is dedicated to reduce the complexity required to calculate the characterization. Thereby, we first tackle the runtime by reducing the number of required checks. Then, we present a compact representation to store all valid assignments. We conclude the section with experimental results conducted with industrial business process models. The mentioned applications of the characterization and the integration into a modeling tool is subject of Sect. 4.

3.2 Reducing the number of checks

A given business process model with T tasks (transitions) has $2^{|T|}$ possible outsourcing strategies (assignments) in case no confidentiality assignment is given. For each assignment, one has to decide whether it is secure using PBNI+. Typically more than one potential place with at least one triple exists for each assignment. The number of all triples can be estimated by $O(|P| \cdot |T| \cdot (|T|-1))$, following from the structure of the triples. Together with $2^{|T|}$ possible assignments one would end up with $O(2^{|T|} \cdot |P| \cdot |T| \cdot (|T|-1))$ checks to investigate all possible assignments. This makes the approach infeasible for industrial business process models with hundreds of tasks. This section will show how this number of checks can be decreased dramatically.

Based on the definition of PBNI+, one active place is enough to violate the noninterference property. Consequently, one active triple (one active place may be detected by more than one triple) is also enough to violate PBNI+. For each assignment, it is therefore sufficient to find one active triple to convict it invalid.

Previously, we showed that triple checks can be performed independently [1] for a given assignment. In fact, this does not reduce the number of triples, but all potential critical assignments follow from the structure of the Petri net. Thus, for PBNI+ only potential causal and conflict places are relevant, which can be expressed as triples. For instance, four assignments from the process in Fig. 1 have no potential triple and are hence valid without the need of a check. For all other assignments only specific parts (the triples) of the net are interesting and necessary to decide PBNI+. Those triples follow from the net structure (one place and two connected transitions) and the labeled security

domains, therefore it is sufficient to check every possible triple only once. Each triple reasons about two transitions and especially their labeling and therefore influences $2^{|T|-2}$ assignments. The number of checks is so decreased from $O(2^{|T|} \cdot |P| \cdot |T| \cdot (|T| - 1))$ to $O(|P| \cdot |T| \cdot (|T| - 1))$.

Running example (cont.). Considering the running example with the initial confidentiality requirements, where the task initiate fraud investigation shall be confidential and the task submit insurance claim is supposed to be outsourced, further checks can be ruled out. As the labeling for two tasks is fixed, the triples (1) and (4) are left to be checked for all assignments.

3.3 Compact representation

The previous subsection showed how a few checks are sufficient to verify noninterference in all possible assignments. As motivated earlier, a characterization of all valid assignments allows for various applications, ranging from immediate feedback to the completion of incomplete assignments. Thereby, an explicit naive enumeration of all valid assignments is infeasible due to exponential blowup.

Each failed check can be understood as a constraint that needs to be satisfied by all assignments to be valid. As discussed earlier, each triple consists of a place s, a *high*-labeled transition h and a *low*-labeled transition l. If the respective check fails, the constraint $\neg(label(h) = high \wedge label(l) = low)$ expresses that this assignment is invalid. Our particular domain allows to use Boolean formulae to encode the constraints. Thereby, we use transition names as variables, the value *true* and *false* for the label *high* and *low*, respectively. The above constraint can be expressed as $\neg(h \wedge \neg l)$. The conjunction of such constraints provides an implicit description of all valid assignments, and its length is polynomial in the size of the business process model.

Given the formula φ for the whole process, any truth assignment of the variables (i. e., any assignment of *high* or *low* to the transitions) that satisfies φ characterizes a valid assignment. Given a complete assignment, this check is linear in the number of transitions. The satisfiability check (i. e., does there exist an assignment that satisfies φ?) is known to be $\mathcal{NP}$-complete for arbitrary Boolean formulae. However, the special structure of our constraints makes this a 2-SAT instance for which linear satisfiability checks are known. To check whether the assignment of certain variables can be deduced by the formula, we proceed as follows. Given a transition t, it must be assigned *true* in all satisfying assignments iff $(\varphi \implies t)$ is a tautology. Likewise, t must be assigned *false* iff $(\varphi \implies \neg t)$ is a tautology. As tautology checks can be reduced to satisfiability, checking for implicit assignment has the same linear complexity.

However, extensions of the noninterference check (e. g., intransitive noninterference [7] which additionally considers downgrading transitions with a new *downgrade* security domain) may allow for more than two security domains which would leave the 2-SAT structure and make these problems $\mathcal{NP}$-complete. To make our approach applicable in the future, we decide to represent the satisfying assignments symbolically using *binary decision diagrams* (BDDs) [8]. For our approach, BDDs offer three advantages: (1) they can represent large sets of bit vectors very compactly, (2) operations such as conjunction or deducing implicit variable assignments are very efficient, and (3) they allow for *typically* very efficient algorithms to check for satisfiability (though they can, of course, not rule out the exponential worst case).

To calculate a BDD that characterizes all valid assignments, we begin with a BDD that models a tautology; that is, allows any assignment. Then, we check for each potential causal and conflict triple whether it is an actual violation of noninterference (i.e., active

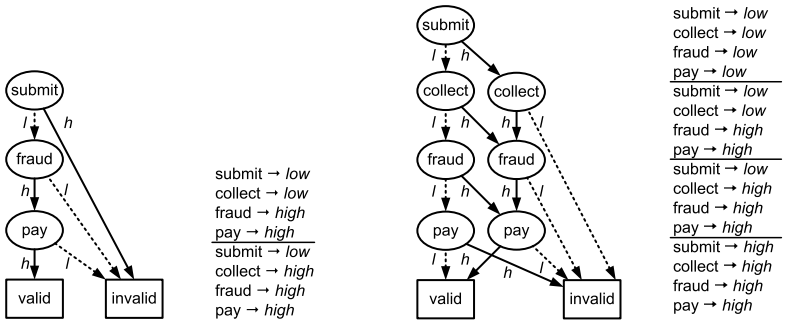


Fig. 3. Running example: BDDs

places). In case a violation is found, the respective (partial) assignment is excluded by adding a constraint to the BDD. At any time, implicit or "static" assignments can be derived from the BDD using the tautology approaches described earlier. Alternatively, BDDs offer more efficient structural methods to check whether the value of a single variable is determined. Such implicit assignments can then be passed to the modeler as a support to avoid redundant manual assignments.

Running example (cont.). Figure 3(a) depicts the BDD that represents all valid assignments of the running example. The oval nodes are labeled with transition names and represent decisions whether to label the transition as *high* (continuous outgoing arrow) or *low* (dashed outgoing arrow). After a sequence of decisions, either the node "valid" or "invalid" is reached which describes the status of the resulting assignment. Note that Fig. 3(a) does not mention the "collect" transition: This means that either label is valid for this transition, resulting in two valid assignment (cf. Fig. 3(b)). We can further derive that the prepare payment task must be confidential in any case. In case no initial assignment is given, the resulting BDD (cf. Fig. 3(c)) characterizes two additional valid assignments: setting all transitions to *high* or all transitions to *low* (cf. Fig. 3(d)).

3.4 Experimental results

For experimental evaluation, we used a library of 559 industrial business process models from different business branches, including financial services, ERP, supply-chain, and online sales. Details on the benchmark set and the translation into Petri nets are provided by Fahland et al. [9]. All of these nets are sound and safe. Based on their origins, the transitions are not labeled for security analysis. To this end, this is a good start for our approach, because we want to characterize all valid assignments.

As summarized in Tab. 1, a characterization for an average process is calculated in 90 milliseconds consuming 8.62 MB of memory and contains 107 nodes. The results of our experiments with industrial business processes are very promising. Even for the biggest process only 282 checks in contrast to more than 2^{100} checks are performed in less than 3 seconds. Thereby the current implementation consumes less than 10 MB of memory and also the biggest characterization can be presented with 1,314 nodes. In all cases no transition was labeled; that is, the numbers reflect the worst case. If a user labels some transitions the problem size decreases noticeable, therefore model support for industrial business processes is feasible.

Table 1. Experimental results of the 559 industrial business processes.

	min	avg	max	running ex.
transitions (exponent of problem size)	1	20	100	4
causal triples (cf. Fig. 2(a))	3	34	242	3
conflict triples (cf. Fig. 2(b))	0	4	90	2
possible assignments (main factor for checks)	2	$> 10^6$	$> 10^{30}$	16
total number of triples (necessary checks)	3	38	282	5
nodes in BDD representation (cf. Fig. 3(c))	7	107	1,317	7
computation time (sec)	0.01	0.09	2.26	0.01
memory consumption (MB)	8.54	8.62	9.45	8.54

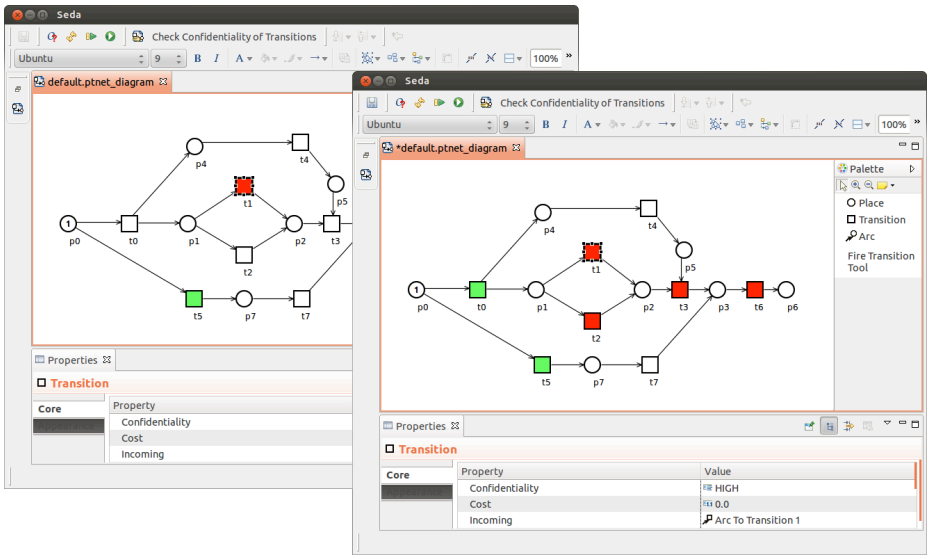


Fig. 4. Screen shots of the editor prototype. After assigning a few transitions (left), implied assignments are calculated automatically (right).

4 Tool integration and modeling wizard

As mentioned earlier, the complete characterization of all valid assignments is just a means to an end: In this section, we report on the prototypic integration of the discussed techniques into an Eclipse-based business process modeling tool and sketch several supported use cases.

To show the basic applicability of our approach, we implemented a proof of concept prototype[1] that is basically a reasoner between a verification library and a modeling tool. As editor, we extended Seda[2], an open source Eclipse-based Petri net modeling tool. Seda offers the usual functionality to model and simulate Petri nets, and was extended to label the transitions with the confidentiality levels *high* and *low*, cf. Fig.4 (left).

[1] Freely available at `http://service-technology.org/anica`.
[2] Freely available at `http://service-technology.org/seda`.

By pressing a button "Check Confidentiality of Transitions", the modeler can check the current assignment for noninterference[3]. In case a leak is detected, a respective error message is displayed. As our experiments showed in Sect. 3.4, the worst case check took about 2 seconds, so a frequent or continuous background check is possible. In fact, a complete verification is only necessary in case the net structure is changed. If only the assignment of transitions is changed, the validity of the assignments can be quickly checked using the calculated BDD. As a consequence, a failed check can be immediately correlated to the last edit actions and avoids a tedious search within the model for the "scapegoat" transitions.

In case the current assignment is valid, the reasoner evaluates the refined BDD and returns all implicit assignments of transitions; that is, transitions where only one possible labeling allows for a completion toward a complete valid assignment. By automatically labeling such transitions, the modeler gets an immediate feedback on his choices and does not need to make these redundant assignments manually. Of course, unwanted choices can be undone.

Figure 4 (right) depicts the automated completion of the assignment on the left. This small example shows the impact on the whole net after assigning only two transitions. As a result, only two transitions remain unlabeled. For these transitions, any labeling is still free to choose. For further details on the tool the interested is referred to [10].

5 Related work

Security has received a lot of attention in the business process management community. In the huge area of security we focus on information flow security. Contributions related to information flow security in the business process management community can be classified as follows.

Explicit information flow. Former research clearly focus on explicit information flow and due the extensive use of Petri nets as formalism for business processes, Petri nets are also often used to reason about security. Atluri et al. [11] proposed a Chinese wall security model for decentralized workflows. Kang et al. [12] used a separation strategy for a similar purpose and provide tool support. Yildiz and Godart [13] focused on information flow policies of different principals. Barletta et al. [14] and Shafiq et al. [15] used colored Petri nets (CPN) and concentrate on role based access control to express separation of duty. Juszczyszyn [16] used CPN for mandatory access control in distributed systems. Zhang and Xiao [17] also used CPN to express and reason about the strict integrity policy, comparable with the approach of Knorr [18]. Huang and Kirchner [19] used CPN for the specification and composition of security policies. Barkaoui et al. [20] are concerned of the data consistency in a multilevel security policy according to information flow rules of the Bell-LaPadula model. In contrast to our work different properties are considered and model support is not in their scope.

Other formalism are used as well to reason about information flow security in business processes [21, 22, 23, 24], but for almost all approaches an automated translation from existing business process modeling languages to used formalisms is missing.

Implicit information flow. Implicit information flow analysis of business processes is a young research strand in the business process management community. In 2009, Busi and Gorrieri [2] defined noninterference properties for Petri nets, which inspired the community of business process management.

[3] A screen cast is available at `http://youtu.be/L7mbIHkGb7A`.

Atluri and Huang [25] presented a kind of CPNs which can be used to automatically detect and prevent violating task dependencies. Beside explicit data flows they also consider implicit data flows. Frau et al. [3] proposed the Petri Net Security Checker, which implements PBNI+ checks for Petri nets. Accorsi et al. [26] introduced information flow nets to capture business process transformations to automatically label the model with classes to consider other properties, including data flow-based properties, separation of duties and declassification. This is indeed another kind of model support. Similar to Frau et al. [3] and Accorsi et al. [26] we presented a verification technique [1] based on reachability and implemented it in the tool Anica.

All realizations so far focus clearly on the verification of a completely assigned business process whereas we provide here a support for this necessary assignment. Only Accorsi and Wonnemann offer some kind of support in the Security Workflow Analysis Toolkit [26], but their support is based on the expression of properties and not on the business process itself. The paper [27] is an unreviewed extended abstract of this paper where we proposed first ideas. In this paper we added the concrete implementation and provided the formal details for the whole approach.

6 Conclusion and future work

Noninterference is an important correctness criterion for business processes that is orthogonal to classical security properties such as encryption or access control. As runtime checks do not make much sense, it has to be considered at the early design phase of a business process. So far, only the verification of noninterference was considered, whereas the design of confidential business processes was neglected. In this paper, we aimed at filling this gap by providing modeler support. We investigated how confidentiality assignments can be efficiently checked and automatically completed and integrated this "modeling wizard" into a prototypic business process modeling tool to demonstrate principal applicability.

So far, we only considered qualitative checks, but made no differentiation between assignments as long as they are valid. Consequently, the consideration of quantitative properties such as costs is an interesting direction of future research. This could allow for the completion of an assignment toward an optimum such as a cheapest outsourcing plan.

Acknowledgement. The authors cordially thank Dirk Fahland for the integration of the reasoner into Seda. This work was partially funded by the German Research Foundation in the project WS4Dsec in the priority program Reliably Secure Software Systems (SPP 1496).

References

1. Accorsi, R., Lehmann, A.: Automatic information flow analysis of business process models. In: BPM 2012. LNCS 7481, Springer (2012) 172–187
2. Busi, N., Gorrieri, R.: Structural non-interference in elementary and trace nets. Mathematical Structures in Computer Science **19**(6) (2009) 1065–1090
3. Frau, S., Gorrieri, R., Ferigato, C.: Petri net security checker: Structural non-interference at work. In: FAST 2008. Volume 5491 of LNCS., Springer (2008) 210–225
4. Denning, D.E., Denning, P.J.: Certification of programs for secure information flow. Commun. ACM **20**(7) (1977) 504–513
5. Lampson, B.W.: A note on the confinement problem. Commun. ACM **16**(10) (1973) 613–615

6. Lohmann, N., Verbeek, H., Dijkman, R.M.: Petri net transformations for business processes – a survey. LNCS ToPNoC **II**(5460) (2009) 46–63
7. Gorrieri, R., Vernali, M.: Foundations of security analysis and design vi. Springer (2011) 125–151
8. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. Computers **C-35**(8) (1986) 677–691
9. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. Data Knowl. Eng. **70**(5) (2011) 448–466
10. Lehmann, A., Fahland, D.: Information flow security for business process models - just one click away. In: BPM Demo 2012. (2012)
11. Atluri, V., Chun, S.A., Mazzoleni, P.: A chinese wall security model for decentralized workflow systems. In: ACM CCS 2001. CCS '01, ACM (2001) 48–57
12. Kang, M.H., Froscher, J.N., Sheth, A.P., Kochut, K., Miller, J.A.: A multilevel secure workflow management system. In: CAiSE '99. LCNS, Springer (1999) 271–285
13. Yildiz, U., Godart, C.: Design and implementation of information flow-sensitive business processes. In: ECOWS '08, IEEE Computer Society (2008) 177–186
14. Barletta, M., Ranise, S., Viganò, L.: A declarative two-level framework to specify and verify workflow and authorization policies in service-oriented architectures. Serv. Oriented Comput. Appl. **5**(2) (2001) 105–137
15. Shafiq, B., Masood, A., Joshi, J., Ghafoor, A.: A role-based access control policy verification framework for real-time systems. In: WORDS '05, IEEE Computer Society (2005) 13–20
16. Juszczyszyn, K.: Verifying enterprise 's mandatory access control policies with coloured Petri nets. In: WETICE '03, IEEE Computer Society (2003) 184
17. Zhang, Z.L., Hong, F., Xiao, H.J.: Verification of strict integrity policy via Petri nets. In: ICSNC '06, IEEE Computer Society (2006) 23
18. Knorr, K.: Multilevel security and information flow in Petri net workflows. Technical report, Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis (2001)
19. Huang, H., Kirchner, H.: Formal specification and verification of modular security policy based on colored Petri nets. IEEE Trans. Dependable Secur. Comput. **8**(6) (2011) 852–865
20. Barkaoui, K., Ayed, R.B., Boucheneb, H., Hicheur, A.: Verification of workflow processes under multilevel security considerations. In: CRiSIS, IEEE (2008) 77–84
21. Attali, I., Caromel, D., Henrio, L., Del Aguila, F.L.: Secured information flow for asynchronous sequential processes. Electron. Notes Theor. Comput. Sci. **180**(1) (2007) 17–34
22. Bossi, A., Focardi, R., Piazza, C., Rossi, S.: Transforming processes to check and ensure information flow security. In: AMAST '02, Springer (2002) 271–286
23. Harris, W.R., Kidd, N., Chaki, S., Jha, S., Reps, T.W.: Verifying information flow control over unbounded processes. In: FM. Volume 5850 of LNCS., Springer (2009) 773–789
24. Kovács, M., Seidl, H.: Runtime enforcement of information flow security in tree manipulating processes. In: ESSoS. Volume 7159 of LNCS., Springer (2012) 46–59
25. Atluri, V., Huang, W.K.: An extended Petri net model for supporting workflow in a multilevel secure environment. In: DBSec 1996. IFIP Conference Proceedings 79, Chapman & Hall (1997) 240–258
26. Accorsi, R., Wonnemann, C., Dochow, S.: SWAT: A security workflow toolkit for reliably secure process-aware information systems. In: ARES 2011, IEEE (2011) 692–697
27. Lehmann, A., Lohmann, N.: Model support for confidential service-oriented business processes. In: ZEUS 2012, Bamberg, Germany (2012)