Designing Secure Business Processes with SecBPMN

Mattia Salnitri · Fabiano Dalpiaz ·
Paolo Giorgini

Abstract Modern information systems are increasingly large and consist of an interplay of technical components and social actors (humans and organizations). Such interplay threatens the security of the overall system, and calls for verification techniques that enable determining compliance with security policies. Existing verification frameworks either have a limited expressiveness that inhibits the specification of real-world requirements, or rely on formal languages that are difficult to use for most analysts. In this paper, we overcome the limitations of existing approaches by presenting the SecBPMN framework. Our proposal includes: (1) the SecBPMN-ml modeling language, a security-oriented extension of BPMN for specifying composite information systems; (2)

M. Salnitri

Department of Information Engineering and Computer Science,

University of Trento, Italy

E-mail: mattia.salnitri@unitn.it

F. Dalpiaz

Department of Information and Computing Sciences,

Utrecht University, the Netherlands

P. Giorgini

Department of Information Engineering and Computer Science,

University of Trento, Italy

the SecBPMN-Q query language for representing security policies; and (3) a query engine that enables checking SecBPMN-Q policies against SecBPMN-ml specifications. We evaluate our approach by studying its understandability and perceived complexity with experts, running scalability analysis of the query engine, and through an application to a large case study concerning air traffic management.

Keywords Information systems · Security policies · BPMN · Compliance

1 Introduction

Information systems are becoming increasingly large, complex, and decentralized. Air Traffic Management (ATM) systems, smart grids, and smart cities are not just monolithic software systems, but rather they are socio-technical systems [10, 54] that consist of multiple autonomous, heterogeneous, and mutually interdependent social (humans, organizations) and technical (software, embedded systems, etc.) components. Humans are an essential part of socio-technical systems; they constitute, along with the organizations they shape, the social part of such systems, through the creation of a complex network of social dependencies.

The complexity of these systems and their blending with society calls for new design techniques, for their crashes entail severe effects in the broader societal context where the systems operate [54]. Furthermore, due to the large amount of private and confidential information that they manage, their design shall treat information assurance and security as primary concern that is analyzed from both a social/organizational and a technical perspective [44].

Business process models are an adequate design abstraction to support the design of socio-technical systems, for they enable specifying the interactions between humans, organizations, and technical systems. However, two challenges shall be addressed to effectively use business process models in the design of secure socio-technical systems: (i) including explicit primitives for modeling se-

curity aspects, and (ii) developing automated verification techniques to check the compliance of a business process model with certain security policies.

To overcome the security modeling challenge, languages have been proposed that extend BPMN (Business Process Modelling and Notation) [39]—the de-facto standard for representing business processes—with security annotations that constrain individual BPMN elements [44, 59]. However, these annotations do not support specifying security policies about the admissible behavior of the whole process. Some other BPMN extensions employ a predefined set of policies—BPMN patterns which specify the desirable behavior—but they do not allow the definition of custom security policies [7, 35].

Concerning the verification challenge, existing techniques suffer from two main limitations: (i) they focus on general-purpose policies and provide no explicit support to security policies [2, 18, 31, 45], thereby relying on a too generic vocabulary; and (ii) they include a too limited set of policy types, mainly concerning access control [34, 35, 44], thereby covering a little part of the information security spectrum.

In this paper, we address both challenges through SecBPMN, a framework for modeling and verifying the compliance of a business process model with security policies. We take BPMN-Query (BPMN-Q) [2]—a query language that enables expressing and verifying generic queries over a BPMN model—as our baseline. We extend BPMN-Q with a number of annotations for expressing security policies. Our contributions are as follows:

1. The SecBPMN-modeling language (SecBPMN-ml) extends BPMN with annotations to express security aspects in a business process model.
2. The SecBPMN-Query (SecBPMN-Q) language, an extension of BPMN-Q for specifying security policies as queries.
3. A software toolset for modeling in SecBPMN-ml, specifying security policies in SecBPMN-Q, and checking for compliance of SecBPMN-Q queries against SecBPMN-ml diagrams.

4. An evaluation of the SecBPMN framework that includes (i) a study of the understandability and perceived complexity with expert modelers; (ii) a scalability analysis of the compliance verification engine; and (iii) an application to a large-scale case study in the ATM domain.

This paper extends our previous work [48] with the following: (i) we present a process to support analysts in using SecBPMN; (ii) we conduct a study with expert modelers about understandability and perceived complexity; (iii) we run experiments to analyze the scalability of our verification engine; (iv) we provide an extended description of the related works.

The research conducted for the original paper and for this extension follows the six steps proposed in the Design Science Research Methodology (DSRM) [42]: (i) *problem identification and motivation*, we identified the need of representing security aspects in business processes, and we looked for existing solution in the existing literature; (ii) *definition the objectives for a solution*, we determined the best solution that an adequate solution is a framework that enables modeling business processes with security aspects, representing security policies and verifying the latter against the former; (iii) *design and development*, we identified BPMN as our baseline modeling language, the set of concepts to be added and how to extend BPMN; (iv) *demonstration*, by applying it to a case study, we show the validity of the solution we created; (v) *evaluation*, we conducted an online survey with experts and ran a set of scalability tests; (vi) *communication*, through the publication of the original paper [48], the present extension, and technical reports available on [52].

The rest of the paper is structured as follows. Section 2 describes our baseline. Sections 3 and 4 introduce SecBPMN-ml and SecBPMN-Q, respectively. Section 5 presents our proposed process for using the SecBPMN framework. Section 6 reports on our evaluation, while Section 7 discusses related works. Finally, Section 8 presents our conclusions and outlines future directions.

2 Baseline

In this section, we introduce the baseline of our research: the BPMN-Q language for querying business process models, and the RMIAS security reference model that constitutes a comprehensive high-level ontology of security.

While BPMN is adequate for expressing the interactions among the components in a complex socio-technical system, it does not natively support the verification of compliance with certain security properties that should hold in the model. For example, when modeling the landing procedure in ATM, one may want to verify that in the process it is always the case that pilots will confirm the landing trajectory of the plane.

Visual analysis of BPMN models works only for small scenarios, but it becomes ineffective when many models exist, or when they are as large as hundreds of elements. Moreover, when safety and security properties are at stake, relying on an informal analysis is not an option, due to the harmful effects of adopting a model that violates them.

BPMN-Q is a diagrammatic query language which partially overcomes this limitation, by expressing properties concerning business process models through graphical queries that can be checked against a BPMN model [3]. These queries can be seen as patterns that a given BPMN model should comply with. BPMN-Q introduces relations that are functional to define the queries, i.e., the concepts of path, negative path and negative flow.

Figure 1 shows an example of a BPMN-Q query, using the SWIM ATM case study[1] that we use throughout this paper. The query enables checking whether the flight plan (Reference Business Trajectory or simply RBT) is approved and if the landing documents are checked at least once. The query will match against any business process model which: (i) contains an activity labeled "Plane RBT generation service" and such activity generates the data object "RBT [Proposed]" (the text within brackets denotes the state of the

---

[1] The System Wide Information Management (SWIM) [15] is a next-generation communication system for the secure exchange of information among ATM decision makers.

data object); (ii) contains a path, i.e., a sequence of BPMN elements connected through a control flow, that connects the first activity to a parallel gateway; (iii) contains a path that connects the gateway to "Control Tower communication service" that generates the data object "RBT [Accepted]"; (iv) contains another path that connects the gateways to an activity with any label ("@Y") that reads the data object "Landing documents [Approved]".
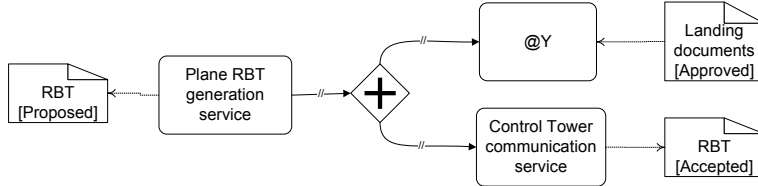
Fig. 1 A BPMN-Q query for the SWIM ATM case study

BPMN-Q enables expressing generic properties over BPMN elements, but does not provide any explicit modeling primitive for specifying security properties. We intend to overcome this limitation by defining security annotations that adhere with a state-of-the-art reference model for information security.

A prominent family of reference models relies on the Confidentiality Integrity Availability (CIA) triad [40]. However, their adequacy has been questioned for they characterize a too limited set of properties of a system [41]. Richer models exist, such as McCumber's cube [33], which conceives system security from three perspectives: information states, critical information characteristics, and security measures. The Business Model for Information Security (BMIS) [22] addresses four interconnected elements: organization design and strategy, people, process, and technology.

In our work, we choose the Reference Model on Information Assurance and Security (RMIAS) [8], which was assembled through an analysis and classification of security aspects proposed by the most known reference models on information assurance and security. As far as our knowledge goes, RMIAS proposes the most comprehensive set of security aspects, for it aggregates and classifies security aspects proposed in the most reknown reference models on security

and information assurance, such as the Confidentiality-Integrity-Availability (CIA) triad and BMIS [22]. The security aspects proposed in RMIAS are listed in Table 1.

Table 1 Security aspects covered by the RMIAS reference model [8]

Name	Definition
Accountability	An ability of a system to hold users responsible for their actions (e.g. misuse of information).
Auditability	An ability of a system to conduct persistent, non-by passable monitoring of all actions performed by humans or machines within the system.
Authenticity	An ability of a system to verify identity and establish trust in a third party and in information it provides.
Availability	A system should ensure that all system's components are available and operational when they are required by authorised users.
Confidentiality	A system should ensure that only authorised users access information.
Integrity	A system should ensure completeness, accuracy and absence of unauthorised modifications in all its components.
Non-Repudiation	The ability of a system to prove (with legal validity) occurrence/non-occurrence of an event or participation/non-participation of a party in an event.
Privacy	A system should obey privacy legislation and it should enable individuals to control, where feasible, their personal information (user-involvement).

3 SecBPMN-ml: a modeling language for secure business processes

We extend BPMN with security annotations to cover each of the security aspects in the RMIAS reference model (Table 1). Every annotation has a graphical syntax and has to be linked with an existing element of a BPMN model: an activity, a data object, or a message flow. Moreover, annotations have attributes that are used to specify detailed information on the security

mechanisms[2] that enforce the policy. Attributes are optional except for the one linking the annotation with a BPMN element.

More precisely, SecBPMN-ml extends the subset of BPMN that serves for specifying orchestrations, which enables expressing interactions among information system components: activities, gateways and data objects. Each security annotation is formalized in terms of one or more predicates, one for every type of BPMN element that the annotation can be linked with.

We designed the graphical syntax of the primitives following Moody's guidelines for increasing the usability and comprehensibility of modeling languages [36]. Moody classifies the visual differences between graphical elements of modeling languages in 8 visual variables: horizontal and vertical position, shape, size, color, brightness, orientation and texture. Graphical elements that represent similar concepts should share as many variables as possible, while they should be easily distinguishable among themselves, i.e. they have at least one visual variable not in common.

Security annotations share three common visual variables: they all have an orange fill color, a solid texture, and a circular shape; they differ for the icon in the middle of the circle. Every security annotation has a visual distance of three from non-security annotations, i.e., they can be easily recognized from other elements of the modeling language, and a visual distance of one from other security annotations.

We decided to use icons instead of abstract symbols because icons are deemed as easier to remember and faster to recognize [36]. For example, the icon for the availability security annotation is a clock face, which should recall the concept of "time" and, therefore, should be easily linked to the definition of availability security concepts (see Table 1). Leitner et al. [26–28] conducted empirical studies to propose guidelines for representing a set of security aspects. We did not apply these suggestions because they conflict with the recommendation by the security experts that helped us define the security

---

[2] The low level (software and hardware) functions that implement the controls imposed by the policy [50]

annotations and, moreover, the set of security aspects Leitner et al. took into account covers only partially the security aspects proposed in RMIAS.

Figure 2 shows a model in SecBPMN-ml of the negotiation process for the RBT between a control tower and the pilots of a flight. This process heavily relies on the human participants of the socio-technical system, since most tasks are human-conducted activities.

Constraints of the software tool we developed (available online on [52]) impose to deviate from BPMN 1.1 specifications: SecBPMN allows the specification of only one business process for each diagram. Therefore, subsequent activities executed by different participants, for instance two activities that exchange a message, must be connected with a sequence flow. For example, in Figure 2 Create report and Examine results are linked with a control flow, i.e. they are part of the same business process, even if the former is executed by co-pilot and the latter by Control tower.

The security annotations specify the security aspects that the implemented services and human activities will comply with. The annotations are defined in Table 2 and explained below. Security annotations can be associated only with tasks, data objects and message flows.

In particular, security annotations cannot be linked to events, which represent changes in the environment of a socio-technical system. Events could be associated with two security annotations: auditability and non-repudiation. The former is equivalent to modeling an event in SecBPMN-ml, for its presence in a model implies its monitoring. For what concerns the latter security annotation, an event in BPMN 1.1 is "something that happens during the course of a business process". If an event is external, i.e. not generated by the business process, then its existence is publicly available, and there is no need of a proof of its existence. Instead, if an event is internal, i.e. it is generated as consequence of an internal action, then the non-repudiation shall be applied to the action that generated the event.

Accountability. It applies only to activities—thus, only one corresponding predicate called AccountabilityAct exists—, and expresses the need of mon-

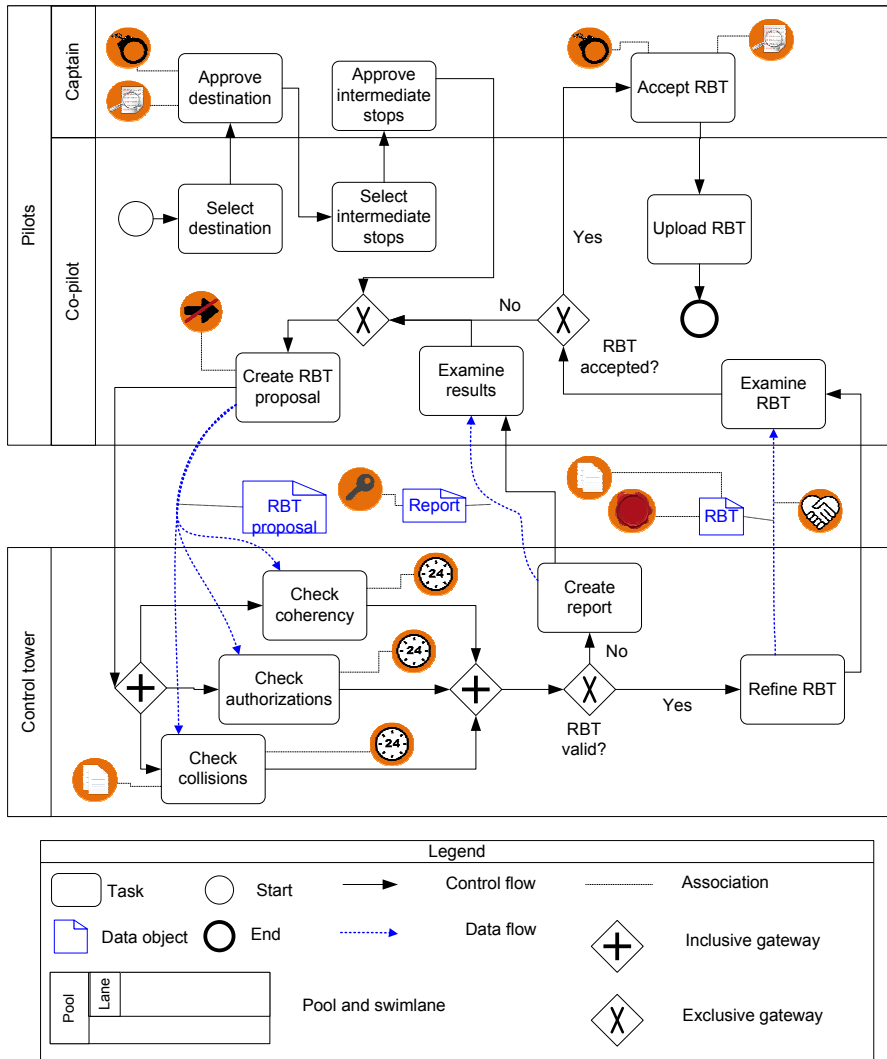**Fig. 2** Example of a SecBPMN-ml business process model

itoring a set of users when executing the activity. The predicate has three parameters: the activity a that is being monitored, a set of security mechanisms enfBy used to enforce accountability for the activity, and the set of users monitored which are monitored.

If the activity is executed by a user that is not in monitored, the security property is satisfied without using the enforcement mechanism. This situation

Table 2 Security annotations in SecBPMN: predicates and their graphical syntax

AccountabilityAct (a: Activity, enfBy: {SecMechanisms}, monitored: {Users})	
AuditabilityAct (a: Activity, enfBy: {SecMechanisms}, frequency: Time) AuditabilityDO (do: DataObject, enfBy: {SecMechanisms}, frequency: Time) AuditabilityMF (mf: MessageFlow, enfBy: {SecMechanisms}, frequency: Time)	
AuthenticityAct (a: Activity, enfBy: {SecMechanisms}, ident: Bool, auth: Bool, trustValue: Float) AuthenticityDO (do: DataObject, enfBy: {SecMechanisms})	
AvailabilityAct (a: Activity, enfBy: {SecMechanisms}, level: Float) AvailabilityDO (do: DataObject, enfBy: {SecMechanisms}, authUsers: {Users}, level: Float) AvailabilityMF (mf: MessageFlow, enfBy: {SecMechanisms}, level: Float)	
ConfidentialityDO (do: DataObject, enfBy: {SecMechanisms}, readers: {Users}, writers: {Users}) ConfidentialityMF (mf: MessageFlow, enfBy: {SecMechanisms}, readers: {Users}, writers: {Users})	
IntegrityAct (a: Activity, enfBy: {SecMechanisms}, personnel: Bool, hardware: Bool, software: Bool) IntegrityDO (do: DataObject, enfBy: {SecMechanisms}) IntegrityMF (mf: MessageFlow, enfBy: {SecMechanisms})	
NonRepudAct (a: Activity, enfBy: {SecMechanisms}, execution: Bool) NonRepudMF (mf: MessageFlow, enfBy: {SecMechanisms}, execution: Bool	
PrivacyAct (a: Activity, enfBy: {SecMechanisms}, sensitiveInfo: {Info}) PrivacyDO (do: DataObject, enfBy: {SecMechanisms}, sensitiveInfo: {Info})	

would typically occur with trusted users that do not need to be monitored. Security designers can specify the keyword ALL in monitored, to indicate that all users are held for their actions.

Consider, for example, the predicate AccountabilityAct("Approve destination", {RBAC}, {juniorPilots}), which details one of the accountability security annotations in Figure 2. The first attribute details the activity linked with the

security annotation, the second one indicates that RBAC (Role-Based Access Control) [16] will be used to enforce accountability, while the third one specifies that only junior pilots have to be monitored while executing that activity.

Auditability. This security annotation comes in three variants, expressing different types of auditability in a business process: (i) AuditabilityAct indicates that it should be possible to keep track of all the actions performed by the executor of the activity a when trying to execute that activity; (ii) AuditabilityDO indicates that it should be possible to keep track of all the actions (e.g., write, read, store) concerning a data object do; (iii) AuditabilityMF indicates that it should be possible to keep track of all the actions executed to handle the communication (send/receive actions) within a message flow mf.

The predicates share two parameters: enfBy to express the set of security mechanisms to be used, and frequency to specify how often the security checks are performed. If frequency is set to zero, continuous verification is required.

For instance, consider the predicate AuditabilityAct("Approve destination", {}, 10d), which formalizes one of the auditability annotations in Figure 2. It applies to activity Approve destination, does not require a specific technology for checking auditability, and requires audits to be performed every 10 days.

Authenticity. It comes in two versions, depending on which BPMN elements the annotation applies to. AuthenticityAct imposes that the identity and/or authenticity of the users of activity a are verified. The attribute enfBy is the set of security mechanisms to be used, while trustValue is the minimum level of trust [23] the executor of activity a must have. If attribute ident is true, anonymous users should not take part in the execution of the activity, while if auth is set to true, the identity of users should be verified. AuthenticityDo indicates that it should be possible to prove the data object do is genuine: the fact that do was not modified by unauthorized parties, and it contained proofs of the identity of the entities who generated and/or modified it.

For example, consider the predicate AuthenticityDO ("RBT", {TLS, X.509}), which formalizes an authenticity security annotation in Figure 2. The predi-

cate specifies that the integrity of RBT data object should be guaranteed using TLS (Transport Security Layer) and X.509 security mechanisms.

Availability. It applies to three BPMN elements, hence we defined three different versions: (i) AvailabilityAct specifies that the activity a should be ready for execution whenever the activity is encountered in the control flow of the business process; (ii) AvailabilityDO specifies that the data object do should be available when required by the authorized users specified in the attribute authUsers; (iii) AvaliabilityMF specifies that it should always be possible to communicate through the message flow mf.

The predicates share two parameters: enfBy, described above, and level, i.e., the minimum time percentage that the resource (i.e., activity, data object or message flow, depending on the variant of availability annotation) should be available. In AvailabilityDO, security designers can specify that all users are authorized to request the data object, simply specifying the keyword ALL in the attribute authUsers.

For instance, the predicate AvailabilityAct("Check coherency", {SAVE}, 99.5) specifies that Check coherency has to process at least 99.5% of the total requests, using the SAVE (Source Address Validity Enforcement) [29] protocol to prevent denial of service attacks.

Confidentiality. It has two variants: ConfidentialityDO specifies that the data object do can be accessed only by authorized users, and ConfidentialityMF specifies that only authorized users can use (i.e., send or receive through) the message flow mf. Both predicates share three parameters: enfBy, already described; readers, i.e., the set of users that are authorized to read the data object (or receive from the message flow); writers, i.e., the set of users that are authorized to write the data object do (or send through the message flow). The attributes readers and writers allow the usage of the keyword ALL to specify that all the users are authorized.

For example, consider the predicate ConfidentialityMF (mf("Refine RBT", "Examine RBT"), {TLS, RBAC}, {towerControl, controlAuthority, RBTOwner},

{towerControl, RBTOwner}), which details one of the confidentiality annotations in Figure 2. It specifies that only the users controlTower, controlAuthority and RBTOwner can receive from the message flow between Refine RBT and Examine RBT, and only RBTOwner and controlTower can send data objects through that channel. This security annotation must be enforced using both TLS and RBAC security mechanisms.

Integrity. It comes in three variants: (i) IntegrityAct specifies that the functionalities of activity a should be protected from intentional corruption. Attributes personnel, hardware and software determine which entities—involved in the execution of the a—are protected from intentional corruption [17]; (ii) IntegrityDO specifies that the data object do should be protected from intentional corruption; (iii) IntegrityMF specifies that every message exchanged through mf should be protected from intentional corruption. All the predicates share the attribute enfBy.

For instance, the predicate IntegrityAct("Check collisions", {} , false, true, true) specifies one of the integrity annotations in Figure 2. It indicates that software and hardware used to execute Check collisions will be protected from intentional corruption, e.g., unauthorized modifications of the software or hardware robbery.

Non-Repudiation. It comes in two variants, depending on the element it applies to: NonRepudiationAct and NonRepudiationMF. The former indicates that the execution (or non-execution) of activity a should be provable, while the latter specifies that the usage (or non-usage) of the message flow mf should be verifiable. Both predicates share two attributes: enfBy, already described before, and execution. The latter specifies that: (i) a proof of execution of activity a or a proof of usage of the communication channel mf shall be provided, if set to true; (ii) a proof of non-execution for a or a proof of non-usage for mf shall be provided, if set to false.

For example, the predicate NonRepudiationAct("Create RBT proposal", {}, false) defines one of the non-repudiation annotations in Figure 2. It specifies

that it should be possible to prove that Create RBT proposal has never been executed. There are no constraints on the security mechanisms that have to be implemented because the parameter is an empty set.

Privacy. It has two variants: (i) privacyACT specifies that activity a should be compliant with privacy legislation, and it should let users to control their own data; (ii) privacyDO is similar to the former one, but is targeted to a specific data object do. Both predicates share two parameters: enfBy, already described, and sensitiveInfo, i.e., the set of sensitive information to protect.

For example, consider the predicate PrivacyDO("Report", {} , {name, surname, dateOfBirth}), which refines one of the privacy annotations in Figure 2. It specifies that, if the content of Report is published, name, surname and date of birth information shall be anonymized as required by law, e.g., only partial information can be published.

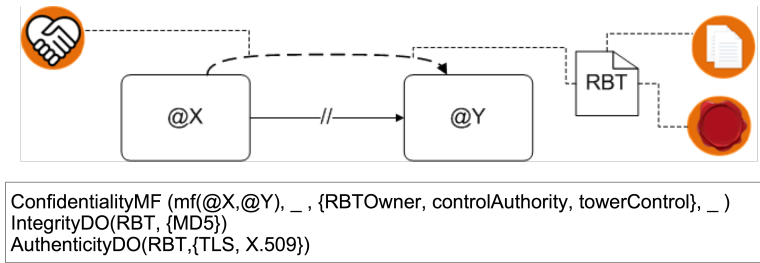## 4 Modeling and verifying security policies

We introduce the SecBPMN-Q language, an extension of BPMN-Q query language, to model security policies using the security annotations that we describe in Table 2. Our query language permits the graphical modeling of security policies, which is a useful feature to facilitate the communication among modelers and with other stakeholders.

Consider, for example, a textual policy such as "*The RBT document must be authenticated and it must be sent through a secure channel which assures the information will not be sniffed or modified by third parties, implementing TLS and X.509 security mechanisms*". Figure 3 shows an equivalent modeling of this policy in SecBPMN-Q.

Beside the two generic tasks and the path, that are elements of BPMN-Q, the model features a message flow (represented as a dashed arrow) that transfers a data object called "RBT". When executed, this query will match any message flow between two activities which exchange the "RBT" data object. The confidentiality annotation linked with the message flow requires the

communication channel to assure the data object will be received only by "RBTOwner", "controlAuthority" and "towerControl". Moreover, the "RBT" data object has to be protected by unauthorized modifications, implementing the "MD5" security mechanism specified by integrity annotation, and its originality has to be provable using "TLS" and "X.509" security mechanisms, specified in the authenticity annotation.

Note that some attributes are not specified, meaning that the security designer is imposing fewer constraints on the specific security mechanism; for example, enfBy and writers parameters of ConfidentialityMF are not defined in Figure 3 (see the underscore wildcard), hence the predicate will be satisfied regardless the security mechanisms implemented or the set of users authorized to send data objects through the channel.

ConfidentialityMF (mf(@X,@Y), _ , {RBTOwner, controlAuthority, towerControl}, _)
IntegrityDO(RBT, {MD5})
AuthenticityDO(RBT,{TLS, X.509})

Fig. 3 Example of a security policy and predicates expressed with SecBPMN-Q

In order to verify if a SecBPMN-ml business process is compliant with security policies modeled with SecBPMN-Q, we extended the BPMN-Q engine as described in Algorithm 1. The algorithm takes in input a SecBPMN-ml business process and a SecBPMN-Q security policy, and it verifies if there exists a path in the business process that satisfies the security policy. For each path (Line 4), the algorithm verifies if the security annotations of the business process are of the same type of those in the security policy (Line 8) and if they are linked to the same SecBPMN-ml element (Line 9). If so, the security annotations of the security policy are verified against the security annotations in the business process (Line 10).

Algorithm 1 Compliance check of a security policy

COMPLIANCE(*SecBPMN-ml bp*, *SecBPMN-Q secPolicy*)

1   *paths* ← FINDPATH(*bp*, *secPolicy*)

2   **if** *paths* = ∅ **then**

3   **return false**

4   **for each** *path* ∈ *paths* **do**

5   *satisfied* ← **true**

6   **for each** *secAnnPolicy* ∈ GETSECURITYANNOTATIONS(*secPolicy*) **do**

7   **for each** *secAnnPath* ∈ GETSECURITYANNOTATIONS(*path*) **do**

8   **if** *secAnnPolicy.type* = *secAnnPath.type* **then**

9   **if** CHECKTARGET(*secAnnPath*, *secAnnPolicy*) **then**

10   *satisfied* ← SATISFIES(*secAnnPath*, *secAnnPolicy*) ∧ *satisfied*

11   **if** *satisfied* **then**

12   **return true**

13   **return false**

A security annotation of a business process satisfies a security annotation of a security policy if all the attributes of the former are more restrictive of the attributes of the latter. The function satisfies, defined in Algorithm 2, checks this property. As first step, Algorithm 2 checks if the security mechanisms specified in the security annotation of the policy are all specified in the security annotation of the business process (Line 1); if not, it returns false, meaning that the security policy specifies at least a security mechanism that is not implemented in the business process. After that, depending on the type of annotation, the algorithm performs different checks:

– *Accountability* (lines 4-5): are all the monitored users specified in the policy also monitored by the business process?

– *Auditability* (lines 6-7): is the frequency of the checks specified in the business process higher than or equal to the one specified in the business process?

– *Authenticity* (lines 8-11): if the attribute ident is true in the security annotation specified in the security policy (every user has to be identified), then is the same attribute specified in the business process also true? The

Algorithm 2 Pseudo-code of function "satisfies"

SATISFIES(*SecurityAnnotation SecAnnPath*, *SecurityAnnotation SecAnnPolicy*)

1   **if** (*secAnnPolicy.enfBy* $\not\subseteq$ *secAnnPath.enfBy*) **then**

2   **return** false

3   **switch** (*SecAnnPolicy.type*)

4   **case** `AccountabilityAct` :

5   **return** (*SecAnnPolicy.monitored* $\subseteq$ *SecAnnPath.monitored*)

6   **case** `AuditabilityAct` $\vee$ `AuditabilityDO` $\vee$ `AuditabilityMF` :

7   **return** (*SecAnnPolicy.frequency* $\leq$ *SecAnnPath.frequency*)

8   **case** `AuthenticityAct` :

9   **return** ((*SecAnnPolicy.ident* $\rightarrow$ *SecAnnPath.ident*)$\wedge$

10   (*SecAnnPolicy.auth* $\rightarrow$ *SecAnnPath.auth*)$\wedge$

11   (*SecAnnPolicy.trustValue* $\leq$ *SecAnnPath.trustValue*))

12   **case** `AvailabilityAct` $\vee$ `AvailabilityDO` $\vee$ `AvailabilityMF` :

13   **return** (*SecAnnPolicy.value* $\leq$ *SecAnnPath.value*)

14   **case** `ConfidentialityDO` $\vee$ `ConfidentialityMF` :

15   **return** ((*SecAnnPolicy.readers* $\supseteq$ *SecAnnPath.readers*)$\wedge$

16   (*SecAnnPolicy.writers* $\supseteq$ *SecAnnPath.writers*))

17   **case** `IntegrityAct` :

18   **return** ((*SecAnnPolicy.personnel* $\rightarrow$ *SecAnnPath.personnel*)$\wedge$

19   (*SecAnnPolicy.hardware* $\rightarrow$ *SecAnnPath.hardware*)$\wedge$

20   (*SecAnnPolicy.software* $\rightarrow$ *SecAnnPath.software*))

21   **case** `NonRepudiationAct` $\vee$ `NonRepudiationMF` :

22   **return** (*SecAnnPolicy.exeution* $\leftrightarrow$ *SecAnnPath.execution*)

23   **case** `privacyAct` $\vee$ `privacyMF` :

24   **return** (*SecAnnPolicy.sensitiveInfo* $\subseteq$ *SecAnnPath.sensitiveInfo*)

same criteria is used also for attribute auth. The trustValue defined in the security annotation of the security policy has to be less or equal that the value defined in the one specified in the business process, since the security annotation is satisfied when the trust provided by the executor of the activity is higher than that required by the policy.

– *Availability* (lines 12-13): is the value specified in the business process higher than the value specified in the policy?

- *Confidentiality* (lines 14-16): is the set of authorized users specified in the business process a subset of the set of authorized users of the security annotation of the security policy?

- *Integrity* (lines 17-20): if the personnel attribute (of IntegrityAct) is true in the security policy, is it also true in the business process? The same criteria applies for hardware and software. The other two variants of integrity do not need special criteria because they are characterized only by the attribute enfBy, that is already checked in the first two lines of the algorithm.

- *Non-repudiation* (lines 21-22): is the attribute execution set to the same value in both security annotations?

- *Privacy* (lines 23-24): is the set of sensitive information specified in the security policy included in the set specified in the business process?

Our developed software tool, available on [52], permits to model SecBPMN-ml and SecBPMN-Q diagrams and fully implements the algorithms described in this paper.

When a SecBPMN-Q security policy is checked, the interface of our engine returns which ones among the analyzed business processes have at least one path (graphically highlighted in the business process) that satisfies a given security policy. Figure 4 shows the result of the SecBPMN-Q query shown in Figure 3 with the SecBPMN-Q process shown in Figure 2. The path highlighted in Figure 4 satisfies the security policy in Figure 3: (i) the first activity of the path, "Refine RBT", is linked with a message flow to the last activity of the path, "Examine RBT"; (ii) the message flow is used to exchange the data object "RBT" and it assures confidentiality of the transferred data object; (iii) integrity and authenticity of the "RBT" data object are preserved. When the predicates that detail the security annotations of the security policy are less restrictive than the predicates of the business process, the path—and, thus, the business process—satisfies the security policy.
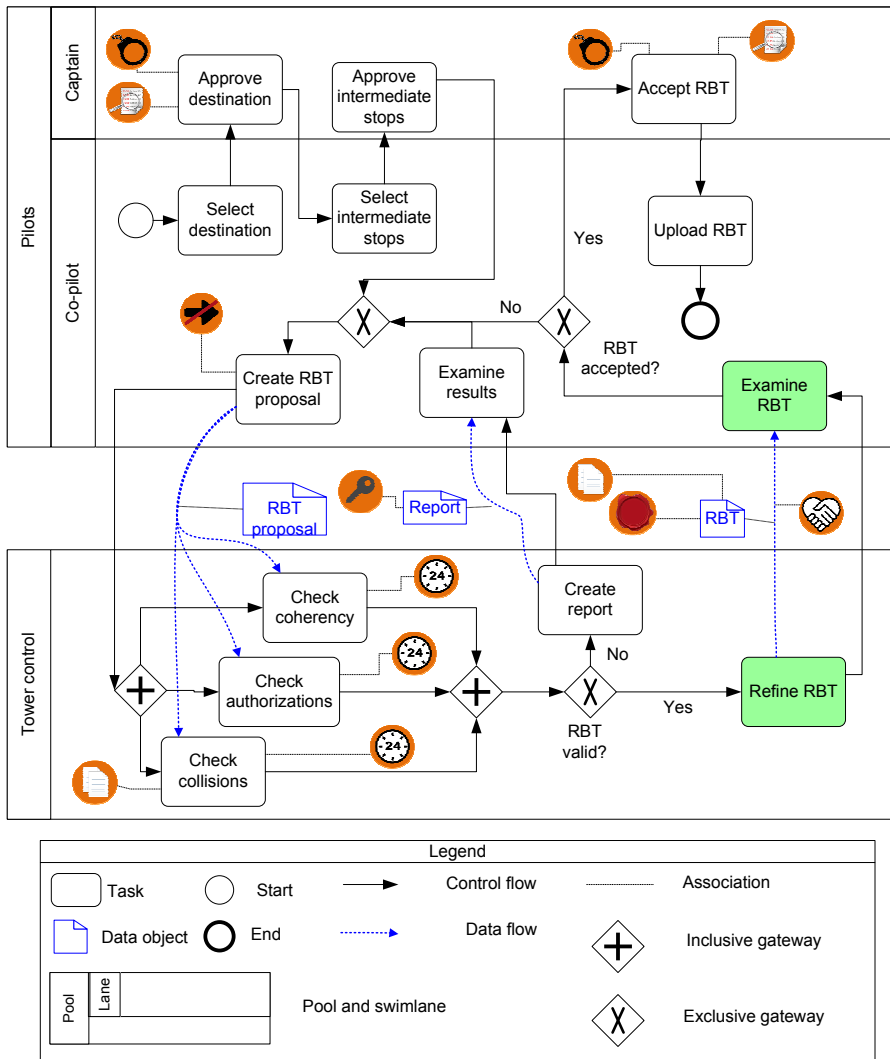
Fig. 4 Result of the query based on SecBPMN-Q policy in Figure 3 against the SecBPMN-ml model in Figure 2

5 A process for SecBPMN

We describe a reference process that details how to use the SecBPMN framework to keep the business processes describing an information system compliant with the security policies specified by the stakeholders. The process is conducted by a team composed of *security experts*—who have the necessary

security knowledge—, and *business analysts*—who are capable of modeling the business processes that the information system will execute.

The process, illustrated in Figure 5, starts with a study of the context, where business analysts, security experts, and stakeholders engage with each other in order to produce a textual description of the business processes and of the security requirements. This activity is conducted with the use of traditional techniques for requirements elicitation and organizational analysis.
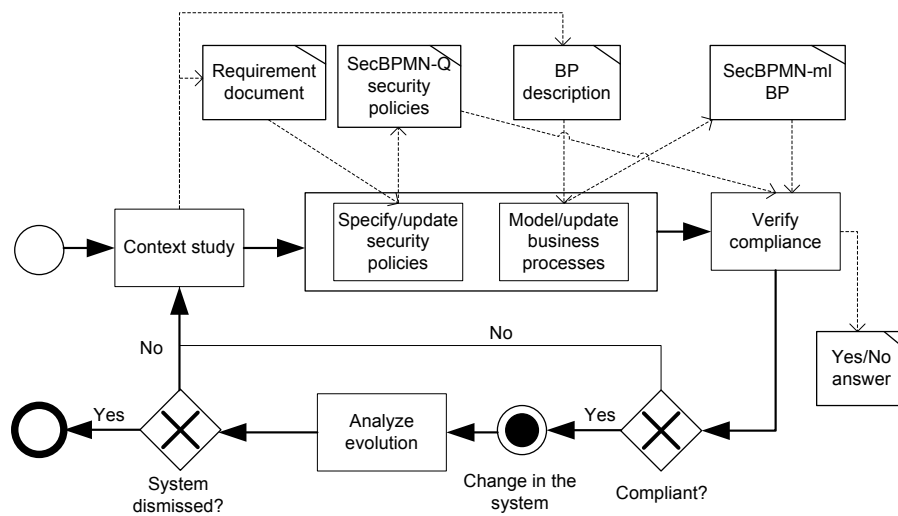

Fig. 5 Suggested process when using SecBPMN framework

The documentation produced in the first step feeds two tightly-coupled activities: modeling the business processes using SecBPMN-ml and specifying security policies using SecBPMN-Q. These two activities are conducted in parallel following an iterative and agile cycle, and they feed each other.

The specification or update of the security policies, which results in the creation of SecBPMN-Q queries, is led by the security experts, with the support of business analysts to evaluate the functional aspects of the security policies. For example, when designing an ATM information system, the best method to protect the communication with the control tower from network

attacks would be to block all communications, but such policy cannot be put into effect, as if would hinder the control tower's operation.

The modeling or update of the business process, which results in the definition of SecBPMN-ml models, is led by the business analysts, with the support of security experts to enrich the diagrams with the annotations that specify the security choices. In the previous ATM example, the business analysts would model the business processes that describe the interaction between the control tower and the other participants, while the security experts will enrich the business process models with security choices; for example, specifying the message flow to preserve the integrity of the transmitted data.

The last step consists in the automated verification—using the SecBPMN verification engine—of the compliance between SecBPMN-ml business processes with SecBPMN-Q security policies. If non-compliance is identified, the security policies and/or the business processes are updated by initiating a new iteration in the process.

Complex information systems may need post-deployment adaptations to cope with external changes. For example, in the ATM system, a renewal in the luggage distribution system would require a new procedure, but such procedure must be verified against all the existing security policies before the new information system is deployed. Security policies can change too; for example, if the privacy legislation changes, all the relevant security policies shall be updated, thereby requiring all business processes to be re-verified against the updated policies. In order to accommodate these situations, our process supports continuous monitoring; when changes occur in the system, or policies are modified, an analysis is conducted to determine whether the system should be evolved—initiating a new iteration in our process—or dismissed.

6 Evaluation

We evaluate the SecBPMN framework in three ways. First, we conduct an empirical study with experts to evaluate the understandability and the perceived

complexity of our modelling languages (Section 6.1). Second, we run a scalability analysis of our verification engine, to determine its suitability for large business process models and security policies (Section 6.2). Third, we evaluate the applicability of our framework on a large ATM case study (Section 6.3).

6.1 Modeling languages understandability and perceived complexity

We designed and conducted an experiment to test the understandability and the perceived graphical complexity of SecBPMN-ml and SecBPMN-Q; we define the latter concept as semiotic clarity [36] and diagrammatic complexity [36]. Our experiment was conducted through an online survey as a way to maximize the number of subjects.

6.1.1 Empirical experiment design

The design of our experiment was conducted following Wohlin's guidelines [58]. We used the Goal, Question, Metric (GQM) template [4] to define the scope and objectives of the survey; in particular, the GQM template specifies: (i) the focus of the experiment; (ii) the objective of the experiment; (iii) the variables to test; (iv) the subjects; and (v) the context of the experiment.

Table 3 shows the two GQM templates for our experiment. The first part of the experiment analyzes SecBPMN-ml for evaluating its perceived graphical complexity and understandability. The experiment targets the main roles involved in the process (Figure 5): security experts and business process modelers. The latter category is a prominent subset of business analysts. The evaluation is performed by asking the subject to read SecBPMN models. The second part of the experiment compares SecBPMN-Q with a formal approach for expressing policies, i.e., Computational Tree Logic (CTL) [14] formulas.

Table 4 shows the hypotheses that we tested with the experiments. We divide hypothesis into two sets, one per experiment: the first set compares SecBPMN-ml with BPMN models with a textual description of the security policy (BPMNts); the second set compares SecBPMN-Q with CTL.

Table 3 GQM template for our experiments

Analyze SecBPMN-ml
for the purpose of evaluation
with respect to their perceived graphical complexity and understandability
from the point of view of the security experts and business process modellers
in the context of reading SecBPMN models.
Analyze SecBPMN-Q
for the purpose of compare it with CTL formulas
with respect to their perceived graphical complexity and understandability
from the point of view of the security experts and business process modellers
in the context of reading security policies.

We chose BPMNts, instead of BPMN, in order to compare SecBPMN with a modeling language with the same expressiveness. Other languages, such as SecureBPMN, can express only a part of the security aspects and, therefore, the comparison would be unfair. For the same reason we chose to compare SecBPMN-Q with CTL: they can express the same type of time patterns.

Table 4 Hypotheses of the experiments

Experiment 1: SecBPMN-ml vs. BPMNts
H0-1.1: SecBPMN-ml is more complex than BPMNts
H1-1.1: SecBPMN-ml is less complex than BPMNts
H0-1.2: SecBPMN-ml is less understandable than BPMNts
H1-1.2: SecBPMN-ml is more understandable than BPMNts
H0-1.3: SecBPMN-ml is more complex and less understandable than BPMNts
H1-1.3: SecBPMN-ml is less complex and more understandable than BPMNts
Experiment 2: SecBPMN-Q vs. CTL
H0-2.1: CTL is preferable to SecBPMN-Q for communication with stakeholders
H1-2.1: SecBPMN-Q is preferable to CTL for communication with stakeholderss

We opted for convenience sampling as a means to recruit subjects: we did spread the word about the survey through mailing lists, used by security experts and business process modelers. We left the survey available on line for 20 days, and then we analyzed the answers.

To evaluate the perceived complexity and the readability of SecBPMN and BPMNts, we created three pairs of diagrams, each consisting of a SecBPMN-ml diagram and BPMNts diagram. To ensure a fair comparison, both diagrams modeled the same business processes, with the same security choices. We also use the same layout, except of the message flow that was colored differently (we discuss the implications in Section 6.1.2).

The survey was structured in different parts (for the details see [52]):

− General questions concerning the background of the subject;
− An introduction to SecBPMN-ml;
− Questions on a small-size business process (SecBPMN-ml/BPMNts);
− Questions on a slightly bigger business process (SecBPMN-ml/BPMNts);
− Questions on a medium-size business process (SecBPMN-ml/BPMNts);
− An introduction to SecBPMN-Q;
− Questions on security policies (SecBPMN-Q/CTL).

6.1.2 Validity of our experiments

We report the main threats to the validity of our experiment, using Wohlin's categorization [58].

Threats to conclusion validity. The relevant threats in this category are the following: (i) low statistical power, as we cannot determine the size of the mailing lists and how many respondents in advance; (ii) random irrelevancies in the experimental setting, for we opted for an on line survey, thereby having no control on external factors which could affect the results of the experiment; (iii) random heterogeneity of subjects, as we distribute the survey on line and we were not able to select adequate participants. Looking at the obtained results, the statistical power threat is only partially addressed: while 30 respondents do not yield strong statistical power, the number is in the average for PhD studies [32]; concerning the third threat, most of the participants had knowledge in business process modeling (28 out of 30 had experience in either BPMN, Petri nets, or UML activity diagrams), and with the subjects having

a knowledge above average in information security average $\bar{x} = 3.13$ (range from 1 to 5), standard deviation $\sigma = 1.41$.

Threats to internal validity. The only relevant threat is <u>mortality</u>. We mitigated such threat by allowing subjects to interrupt the survey at the end of each part. 10% of the subjects interrupted the experiment after the questions about small-size business process, 7%, of the remaining subjects interrupted it in the following part, and 3% in the part about medium-size business processes. Overall, we collected results on the small-size processes from all subjects, on slightly larger models from 90% of the subjects, and on medium-size models from 80% of the sample.

Threats to construct validity. This type of validity is threatened by the <u>restricted generalizability across constructs</u>. In other words, some constructs (diagrams) can influence the valuation of other diagrams. In our case, the danger is that by looking at a diagram in one notation, the user already gets a sense about its meaning, and is facilitated in understanding the alternative notation. We mitigated this threat in different ways. First, since we aimed to assess the SecBPMN framework, we presented our notation first, so that most of the cognitive effort was put on understanding the process using our languages. Second, we modeled the same business processes throughout the survey but with different level of details. For the same threat, we avoid to influence subjects with factors that are not tested in the survey using for each part the same layout and the same detail. Construct validity is also threaten by a <u>difference in the coloring of the diagrams</u>: while we did our best to keep the layouts as similar as possible between business processes of the same pairs, the message flows of the SecBPMN-ml diagrams are colored in blue while the same message flows in BPMN diagrams are black. Another threat to construct validity is <u>hypothesis guessing</u>, where the subjects can be conditioned by the results they are providing. We mitigated this threat by carefully formulating questions as much impartially as possible, and by clearly stating the purpose of the questionnaire.

Threats to external validity. External validity is threatened by the <u>interaction</u> <u>of setting and treatment</u>. In our case, this would occur with business process diagrams that are not an accurate representation of the real process. Due to time constraints, the first two proposed business processes were relatively small; the third one, however, is medium-sized, and constitutes therefore a fair representation of a real-world business process for the chosen domain.

6.1.3 Experiment results

The survey was completed by 30 subjects; the large majority (96%) were familiar with at least one business process modeling language, 60 % where familiar with BPMN standard. The majority of the subjects (60%, $N = 18$) declared to have good or wide knowledge of security, while 40% of the subjects ($N = 12$) are not security experts ($\bar{x} = 3.13$ on a scale from 1 to 5, $\sigma = 1.41$). The original results are publicly available on [52]. Let us review some key results:

– For small diagrams (respondents $N = 30$), SecBPMN-ml is largely considered more understandable and less complex than BPMNts: 80% preferred it to BPMNts, 13% rated both diagrams understandable, 7% found none of them understandable, no-one preferred BPMNts.

– For slightly larger diagrams ($N = 27$), the preference for SecBPMN-ml is confirmed, even though a smaller percentage (67%); 11% of the respondents opted for BPMNts diagrams, 18% of the respondents found both of them understandable, and 4% of the sample found no notation understandable.

– For medium-size diagrams ($N = 25$), the majority of the subjects found SecBPMN-ml more useful to define secure business processes (80%), 8% preferred BPMNts, 8% of the subjects would choose either, and 4% of the sample would use none. When asked about which language would be more effective to communicate with stakeholders, 60% chose SecBPMN-ml, 12% chose BPMNts, 20% chose both of them and, 8% wouldn't use neither SecBPMN-ml nor BPMNts.

An interesting result is about the perceived level of security of business process modeled with SecBPMN-ml, that we tested with the second couple of diagrams. While both diagrams expressed the same security information, the majority of the subjects (74%, $N = 27$) thought that the SecBPMN diagram represents a more secure business process, 15% chose BPMNts diagram, and 11% thought that both diagram represent a business process with the same level of security. This seems to indicate that SecBPMN is more understandable than BPMNts, in the sense that it is easier to identify the security choices.

For what concerns the comparison between SecBPMN-Q and CTL, the former is preferred for communication with customers (88%), none of the subjects chose the CTL formula, 4% would use either of them, and 8% none of them. Regarding the use for verifying compliance, a CTL formula was chosen by the majority of the subjects (54%), 21% would use SecBPMN-Q, 21% would use either of them and 4% would use neither SecBPMN-Q nor CTL. This result shows that the subjects think that SecBPMN-Q is not expressive enough for the verification of security policies. However, SecBPMN-Q is based on temporal logics, and our verification engine fully supports it. It is probably the case that the respondents' opinion is due to the common knowledge about the expressiveness of temporal logic formulas, and had no knowledge on the expressiveness of SecBPMN-Q.

The last question investigated the usefulness of highlighting those paths in a model that satisfy a given security policy; this is a key feature of our modeling and verification toolset. We asked whether this could help security experts to find causes of non-compliance; the respondents ($N = 23$) rated this feature positively, even though not extremely positively: on as scale from 1 (not useful at all) to 5 (extremely useful), we obtained $\bar{x} = 3.78$ and $\sigma = 1.00$.

With the results collected in the survey is possible to refute H0-1.1, H0-1.2, and H0-1.3 and hence confirm H1-1.1, H1-1.2, and H1-1.3. For the set of hypotheses we defined for SecBPMN-Q, it is possible to refute H0-2.1 and to confirm H1-2.1. In other words, the results of the survey constitute a preliminary evidence of the fact that SecBPMN-ml is more understandable and has a

lower perceived complexity than BPMNts, and that SecBPMN-Q is preferred to CTL for communicating security policies with stakeholders.
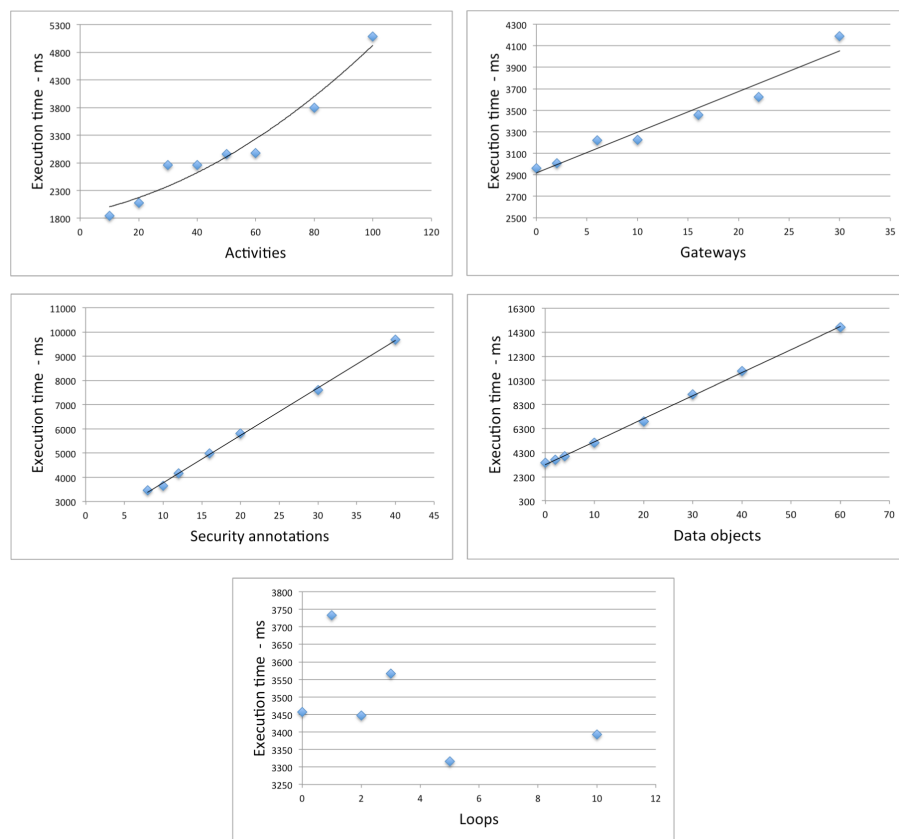
6.2 Scalability analysis

To assess the adequacy of the SecBPMN verification engine for real-world scenarios, we conducted a scalability analysis concerning the engine's performance in terms of *execution time*. We defined two sets of experiments: the former checks how the performance is affected by increasing the complexity of the business process model in SecBPMN-ml, while the latter analyzes the performance trend by increasing the complexity of the security policies in SecBPMN-Q. Each of the experiments was repeated three times, and we took the average time of execution to perform the verification. The models used for the tests can be found online on [52]. We ran our experiments on a virtual machine with a 1Ghz processor, 1GB of RAM memory, and equipped with Microsoft Windows XP.

6.2.1 Scalability with increasingly complex SecBPMN-ml business process

Our first set of experiments aimed at checking which are the factors that affect the scalability of the verification of SecBPMN-ml processes, and to what extent these factors play a role. We chose to test artificially generated models of growing complexity in terms of the number of activities, gateways, loops, data objects, and security annotations. These are all the relevant factors that may affect the verification of a SecBPMN-ml business process. In this experiment, we kept the security policy unaltered, in order to properly test only the factors of the business processes. We used a SecBPMN-Q security policy with 8 activities, 8 security annotations, and 7 paths.

Figure 6 shows the results of the experiments. The dots represent the values for each of the conducted tests, and the continuous line, whenever shown, is a heuristic plot of the scalability curve. The raw results from our results can

Fig. 6 Scalability analysis with increasingly complex SecBPMN-ml business processes

be found online [52]. The experiments show positive results—linear growth—when increasing complexity in the number of gateways, security annotations, and data objects. The increase in the number of activities leads to a polynomial trend. The number of loops does not influence at all the complexity of the compliance verification problem, thanks to the heuristics and optimizations of the engine implementation.

The absolute execution time of the tests is relatively low, also considering the low computational power of the chosen virtual machine. The longest execution took 14.71 seconds on a business process which consisted of 50 activities, 8 security annotations, 60 data objects and 16 gateways. The execution time for the business process with the highest number of activities (100) took

5.09 seconds. We can confidently claim that these are acceptable times. We repeated the tests on a similar machine, with more RAM memory (4 GB). The results followed the same trend as the ones reported in Figure 6, but 0.5 seconds faster. The execution time can be reduced with optimizations such as the adoption of an in-memory object-oriented database, or the usage of an ah-hoc graphical interface.

6.2.2 Scalability with increasingly complex SecBPMN-Q security policies

The objective of the second set of experiments was to discover what factors in a SecBPMN-Q policy affect most the complexity of the verification problem. Just like for the scalability of SecBPMN-ml models, we chose to analyze the number of activities, paths, data objects, security annotations, and gateways. In this experiment we kept the business process unaltered: we used a SecBPMN-ml business process with 50 activities, 30 security annotations, 10 data objects, and 16 gateways.

Figure 7 shows the results of our experiment. Again, the raw results can be consulted online [52]. The results are encouraging, showing that all the considered factors of complexity have a linear impact on the verification execution time.

The absolute execution time is positive. For example, the policy which required the longest execution time, only 10.06 seconds, contains 14 activities, 6 paths, 10 data objects, and 30 security annotations. The execution time of the security policy with the highest number of activities (30) is 1.64 seconds. The execution time for multiple policies is the sum of the execution time of each security policy.

6.3 Application to a case study

We applied the SecBPMN framework to a case study about the SWIM [15] ATM system, part of the Aniketos[3] European FP7 project. The ATM system

---

[3] www.aniketos.eu
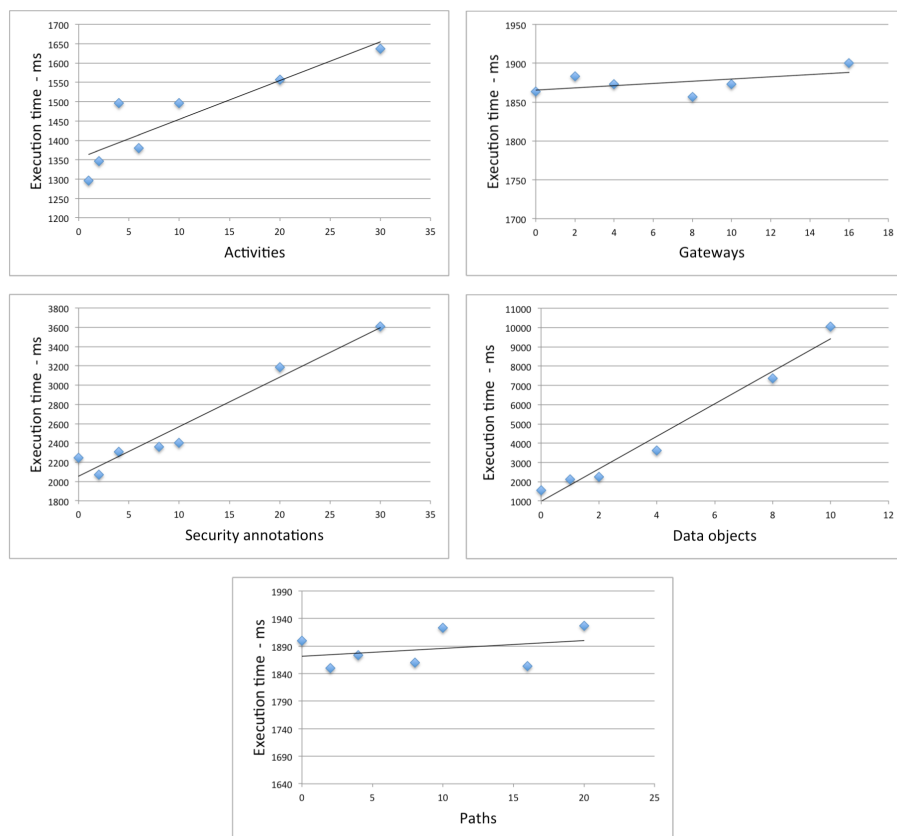
Fig. 7 Scalability analysis with increasingly complex SecBPMN-Q security policies

consists of a large number of autonomous and heterogeneous components, which interact with each other to enable air traffic management operations: pilots, airport personnel, national airspace managers, meteo services, radars, etc. In such a complex socio-technical system, ensuring security is critical, for security leaks may result in severe consequences on safety and confidentiality.

Security experts involved in Aniketos project analyzed the security requirement documents of the SWIM ATM information system and identified 27 active participants and 60 textual security policies. We analyzed those textual security policies and transformed all of them in SecBPMN-Q security policies. In certain cases we transformed a single textual policy into multiple SecBPMN-Q security policies; for example, we transformed a non-disclosure

security policy into a SecBPMN-Q policy concerning the disclosure of electronic documents, and another one about the print of documents. In general, the transformation process was easy, and critical points were about the interpretation of the textual security policy, and never due to limitations of SecBPMN-Q language.

SecBPMN-Q enabled us model all the security policies elicited by the experts except for two specific cases:

- security policies concerning redundancy, which we could represent only at a high-level of abstraction, without managing to express if the fallback activities have to be performed by the same or a different executor. This limitation was inherited by BPMN-Q, which does not support verifying policies that concern swim-lanes and pools.
- security policies about the non-delegation of an activity, i.e., preventing that third parties execute one activity or parts of it. Even in this case, our future work includes introducing additional elements to the meta-model to support this type of policy.

The dimensions of SWIM ATM information system are considerably wide but with similar, redundant, sub-parts. Therefore, when we used SecBPMN-ml to model the business processes, we opted to focus on three representative aspects of the information system: the management of external services, the landing and the taking-off. We also chose to minimize the number of diagrams, aggregating, where possible, the processes in a single, comprehensive, model. This led to the creation of four SecBPMN-ml diagrams: two for the service management, one for the taking-off and one for the landing. With this choice we used SecBPMN-ml language with medium-size business processes instead of small-size ones. The SecBPMN-ml diagrams are not included because their dimension prevent their readability, but they can be found online on [52].

We modeled one business process for the taking-off procedure. In such procedure the RBT is negotiated between the tower control and the pilots, after that, some services are called to check the consistencies of the RBT then,

if the FO (Flight Object) is authorized, the takeoff can start. This procedure is executed for each airspace, an area controlled by a control tower, crossed by the RBT of the FO. This business process is composed by 48 elements that are executed by 3 different participants. It contains 13 message flows and 17 data objects and 31 security annotations.

For the landing process, we modeled a business process in which the FO negotiates, with the tower control, a RBT to the landing point. When the FO reaches the landing position (i.e., the airspace above the airport to land) it waits flying on the RBT defined by the tower control that is controlling the landing point. When its turn arrives, it starts the RBT to land. This business process is composed by 59 elements executed by 4 participants. It contains 14 message flows and 14 data objects and 31 security annotations.

In a SWIM ATM information system external services can be used, for example, to retrieve weather forecasts. But once external services can access to the internal network of the ATM, they may threat a number of assets, therefore special procedures are executed to permit internal components to use the external services and to evaluate the quality of services offered by such services.

A SWIM ATM information system grants to internal users the reliability and trustworthiness of external services with a trust-basted mechanism. When an unknown service is allowed to access the internal network, the ATM system assigns a predefined, low, trust value. Every time a functionality of a external service is used, internal components evaluate that functionality: if the evaluation is positive the trust value of the service provider is increased, otherwise is decreased. This mechanism is used to evaluate external services and to filter the virtuous services from the ones that do not offer a good-enough quality of service. This business process contains 55 elements that are executed by 5 participants. It contains 15 message flows and 16 data objects and 18 security annotations.

Before to be allowed to access to the internal network, an external service and the SWIM ATM information system negotiate the quality of service that

will be offered to SWIM ATM users. The business process we modeled for such procedure is composed by 28 elements that are performed by 4 participants. It contains 5 message flows and 7 data objects. In this business process 14 security annotations are used to represent, on the SecBPMN model, the security aspects of each activity.

During the analysis of the security policy of the case study we realized how different the interpretations of the same security policy can be. This confirms the usefulness of the flexibility of SecBPMN. We found that the understandability of SecBPMN-ml is a key point, because when business processes grow in size and complexity, the security choices can still be easily recognized, giving a good first impression of how the security is handled. On the other side the engine performed very well: during the design of the business processes we check the security policies many times and all the executions took seconds, allowing to check the compliance at each incremental step of the design. For more information on how we used SecBPMN framework to model the procedural aspects of this case study please refer to [49].

7 Related works

The literature offers a number of approaches for expressing and verifying security in business process models. We analyze the most relevant and prominent works. We review extensions of business process modeling languages for security (Section 7.1), methods and guidelines for the design of secure business processes (Section 7.2), approaches for business process compliance verification, i.e., query languages for business processes (Section 7.3) and verification techniques that employ formal languages (Section 7.4).

7.1 Security extensions of business process modeling languages

A natural solution to represent the security aspects of business process is to create or extend a modeling language. Such languages are easy to learn and

to use [36], thereby requiring a moderately low effort for security designers to specify a secure business process.

Menzel et al. [34] propose security extension of BPMN that enables generating security specifications for service-oriented applications. They introduce two security annotations and a set of security properties. Their proposed transformation rules generate machine-readable specification of such security properties in Rampart [56]. The major limitation of this approach consists in the fixed set of security properties that are checked, which disallows for the creation of custom and domain-specific security properties.

Rodriguez et al. [44] take a subset of BPMN and introduce extensions to express a predefined set of security requirement types. However, this approach has limited expressiveness, as it does not take into account the information flow of business processes, and it does not decouple the specification of the policy from the modeling of the security solutions that the process implements.

Saleem et al. [46] extend BPMN with security objectives for Service-oriented Architecture (SoA) applications. They include a set of security concepts in BPMN: confidentiality, integrity, availability, traceability, and auditing. Like the previous ones, the language does not decouple policies from security solutions in the processes. Moreover, their work is specific for the SoA domain.

Wolter et al. [59] propose a modeling language for business processes and business security goals, to be used to graphically define security specifications. They also develop a framework which transforms security goals in security policies specified in XACML [38] and Rampart [56]. The framework automatically extracts specifications of security mechanisms which enforce the security goals, but it does not permit security experts to compose security goals and, therefore, to create complex security policies.

Wolter and Schaad [60] propose an extension of BPMN for specifying task-based authorization constraints. Their approach includes a graphical extension of BPMN as well as a formalization of task-based authorization constraints. Their approach permits to specify dynamic resource allocation such as dynamic separation of duty and role-based resource allocation. Their approach

is focused on authorization constraints of executors of tasks, and it is not possible to use it to specify other security aspects, such as availability or integrity.

Salnitri et al. [47] propose a verification engine for verifying whether a business process is compliant with a given set of security requirements. They use SecureBPMN [7] to represent business processes. The main drawback consists in the fixed set of security requirements that the approach supports.

Schmidt et al. [51] propose two ontologies for defining quality constraints and for defining service processes, respectively. Such ontologies are used to check if a service process complies with the imposed quality constraints. The main drawback of this approach is in the fixed set of constraints that can be specified and checked.

SecureBPMN [7] extends BPMN with access control and information flow constraints. It uses the hierarchic structure of the organization, in which the business process will be executed, to help security designers to define security properties such as, for example, binding of duty [30] and separation of duty [30, 53]. However, SecureBPMN is limited in that it is not possible to specify other central security aspects such as, for instance, confidentiality or availability.

UMLSec [24] is a security-oriented extension of the Unified Modeling Language (UML) [19]. In particular, the extension of UML activity diagrams can be used to define business process with security choices. However, UMLSec does not allow security designers to define security policies and verify them against a process.

7.2 Methods and guidelines

Some approaches provide methods and guidelines that help security experts in the process of constructing sound business process models.

Gruhn and Laue [20] propose a heuristic approach for finding common design errors in business process models, represented using Event Process Chains (EPCs) [57]. They defined a set of rules to check if a business process is not sound or it matches some bad design patterns. Security experts could adopt

this approach to verify the compliance of the business process model against a fixed set of rules. However, using a fix set of rules is a major limitation when dealing with security policies, as it forces security experts to adopt an interpretation of security policies which may not fit the original policy.

Blanc et al. [6] propose an incremental inconsistency checker. Such framework is based on the hypothesis that the definition of a business process is an incremental task, and, thus, inconsistency checking shall be done incrementally. They offer a software tool, based on Prolog [9], which checks if a fixed set of well-formedness rules are satisfied by a business process model. The framework can be applied to any modeling language that can be translated in Prolog. The fixed set of queries is a major limitation, as it inhibits custom security policies.

7.3 Query languages for business processes

Security policies can be seen as patterns, and the their verification against business processes corresponds to the problem of checking if a pattern holds in a business process. Query languages and their software tooling can be used to solve this type of problems, as they allow the creation of queries (patterns), and compliance verification against a business process model.

Dolman et al. [11] propose a pattern matching approach for conceptual models. Such approach consists in algorithms for solving the relaxed graph isomorphism problem, i.e., verifying if the nodes of a labeled graph match with a given pattern (isomorphism problem), and the existence of a path among the graph nodes as indicated in the pattern (homeomorphism problem). They created a tool that implements their algorithms to verify the compliance of a graph with a pattern. The approach is not specific to any modeling language, being rooted in labeled graph theory. However, such approach has to be extended to support the verification of security aspects in a business process.

Beeri et al. [5] propose BP-QL (Business Process Query Language), a pattern-based graphical query language for business processes. They also pro-

vide software tooling to determine the compliance of a business process—defined using WS-BPEL [37]—with a pattern. The decision of using WS-BPEL, a machine-readable standard, hinders the readability of the business process, especially with real case scenarios, where business process easily reach hundreds of elements.

APQL (A Process model Query Language), proposed by Hofstede at al. [21], is a textual query language, based on 20 predicates that can be composed to create complex queries. This approach suffers of scalability issues: the definition of complex queries is a challenging task that will lead to errors due to the complexity of the task. Moreover, as far as our knowledge goes, this approach is not supported by a software framework.

VMQL (Visual Model Query Language) [55] is a graphical query language based on UML activity diagrams [13]. It permits to define custom properties, which are evaluated when the compliance of a query is verified against a business process. But VMQL was not created for security purposes: even if the custom properties can be used to represent security concepts, the VMQL software engine can not interpret them limiting their usage only as a representation of security concepts.

The Business Process Query Language (BPQL) [12] permits to graphically define both queries and business process models using the same language. Unfortunately, BPQL is not based on BPMN, hence the learning process is likely to be slower than that with by BPMN-Q. Moreover, BPQL (just like BPMN-Q), does not include security concepts.

7.4 Verification of properties using formal languages

Some approaches build on logic languages (e.g., first-order, temporal, etc.) for determining compliance. These works are characterized by high expressiveness, but poor usability, for they require a substantial effort for formalizing business processes and security policies.

Sadiq et al. [45] propose to use a Formal Contract Language (FCL) to express normative specifications. Their approach includes a modeling language to visualize business processes as well as normative constraints. They also define a compliance distance, which denotes the extent to which the process model has to be changed to become compliant with the declared constraints. The limitation of this approach is the complexity of the language, despite the provision of a tool to graphical represent normative requirements and business processes. In future work, it would be interesting to compare the usability of the SecBPMN framework with the FCL-based approach.

Liu et al. [31] propose a language and a framework which statically verifies a business process against a formally expressed regulatory requirements. The framework accepts as input a business process specified in WS-BPEL [37] and a set of regulatory requirements, expressed with a temporal logic language called "Business Process Specification Language". While powerful, this approach is hardly usable for large scenarios, due to the complexity of expressing regulatory requirements.

The approaches for business process compliance verification we analyzed do not take in consideration security aspects, therefore is not possible to use them for the purposes of this paper. Moreover, only few of them provide a graphical modeling language for the definition of the patterns to verify, even if it is essential for the usability of such approaches [25, 31].

Other research works [1, 43] use extensions of Petri nets to define business processes with security choices of stakeholders. Petri net modeling language is simple and easy to use but it does not include all the graphical constructs of BPMN. This influence negatively the understandability of models about medium-size or large business processes, limiting the applicability to only small-size business processes.

8 Conclusions and future work

This paper has introduced SecBPMN, a framework for establishing and maintaining compliance between security-annotated business processes and security policies. It is composed by (i) SecBPMN-ml, a modeling language for representing security-annotated business processes; (ii) SecBPMN-Q, a query language for specifying security policies; and (iii) a software toolset that supports both modeling and checking queries against processes. Furthermore, we presented a process that guides analysts while using the SecBPMN framework, and presented an evaluation of our approach.

Our approach overcomes some limits of existing approaches, which either suffer from a low expressiveness—being graphical languages that support only a predefined set of security annotations—, or are hard to use—begin reliant on temporal logics, which are hardly usable by most analysts.

Our approach opens the doors to several future directions, including: (1) applying the languages to different domains; (2) creating a catalog of patterns representing common security policies; (3) including our engine in a workflow system to support security policy-compliant runtime reconfiguration;(4) extending SecBPMN to specify inter-organizational processes; and (5) extending SecBPMN to specify constraints on roles.

References

1. Atluri, V., Huang, W.: An Extended Petri Net Model for Supporting Workflows in a Multilevel Secure Environment. In: Database Security X: Status and Prospects, pp. 199–216 (1996)
2. Awad, A.: Bpmn-q: A language to query business processes. In: EMISA, vol. P-119, pp. 115–128 (2007)

3. Awad, A.: A compliance management framework for business process models. Ph.D. thesis (2010)

4. Basili, V.R., Caldiera, G., Rombach, D.H.: The Goal Question Metric Approach. John Wiley & Sons (1994)

5. Beeri, C., Eyal, A., Kamenkovich, S., Milo, T.: Querying business processes with BP-QL. Information Systems **33**(6), 477–507 (2008)

6. Blanc, X., Mougenot, A., Mounier, I., Mens, T.: Incremental Detection of Model Inconsistencies Based on Model Operations. In: Proc. of CAiSE, pp. 32–46 (2009)

7. Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes. In: Proc. of SACMAT, pp. 123–126 (2012)

8. Cherdantseva, Y., Hilton, J.: A Reference Model of Information Assurance and Security. In: Proc. of ARES, pp. 546–555 (2013)

9. Clocksin, W., Mellish, C.: Programming in PROLOG. Springer Science & Business Media (2003)

10. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Adaptive Socio-Technical Systems: a Requirements-driven Approach. Requirements Engineering **18**(1), 1–24 (2013)

11. Delfmann, P., Dietrich, H., Havel, J., Steinhorst, M.: A Language-independent Model Query Tool. In: Proc. of DESRIST, pp. 453–457 (2014)

12. Deutch, D., Milo, T.: Querying Structural and Behavioral Properties of Business Processes. In: Proc of DPL, pp. 169–185 (2007)

13. Dumas, M., Hofstede, A.H.M.: UML Activity Diagrams As a Workflow Specification Language. In: Proceedings of UML, pp. 76–90 (2001)

14. Emerson, E.A., Halpern, J.Y.: Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. In: Proc. of STOC, pp. 169–180 (1982)

15. Federal Aviation Administration: SWIM ATM case study, last visited March 2014. http://www.faa.gov/about/office_ org/headquarters_ offices/ato/service_ units/techops/atc_ comms_ services/swim/ (2014)

16. Ferraiolo, D., Cugini, J., Richard Kuhn, D.: Role-Based Access Control (RBAC): Features and Motivations (1995)

17. Firesmith, D.: Specifying Reusable Security Requirements. JOT **3**(1), 61–75 (2004)

18. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: Proc. ISOC, pp. 169–180 (2007)

19. Group, O.M.: OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. Tech. rep. (2007). URL `http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF`

20. Gruhn, V., Laue, R.: A heuristic method for detecting problems in business process models. Business Process Management Journal pp. 806–821 (2010)

21. Hofstede, A., Ouyang, C., La Rosa, M., Song, L., Wang, J., Polyvyanyy, A.: APQL: A Process-Model Query Language. In: Proc. of AP-BPM, vol. 159, pp. 23–38 (2013)

22. ISACA: An Introduction to the Business Model for Information Security. Tech. rep. (2009). http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/An-Introduction-to-the-Business-Model-for-Information-Security.aspx

23. Josang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems **43**(2), 618 – 644 (2007)

24. Jurjens, J.: UMLsec: Extending UML for Secure Systems Development. In: Proc. of UML, pp. 412–425 (2002)

25. Kharbili, M.E., de Medeiros, A.K.A., Stein, S., van der Aalst, W.M.P.: Business process compliance checking: Current state and future challenges. In: P. Loos, M. Nttgens, K. Turowski, D. Werth (eds.) MobIS, *LNI*, vol. 141, pp. 107–113. GI (2008)

26. Leitner, M., Miller, M., Rinderle-Ma, S.: An Analysis and Evaluation of Security Aspects in the Business Process Model and Notation. In: Proc. of ARES, pp. 262–267 (2013)

27. Leitner, M., Rinderle-Ma, S.: A Systematic Review on Security in Process-Aware Information Systems- Constitution, Challenges, and Future Directions. Information Software Technology **56**(3), 273–293 (2014)

28. Leitner, M., Schefer-Wenzl, S., Rinderle-Ma, S., Strembeck, M.: An Experimental Study on the Design and Modeling of Security Concepts in Business Processes. In: Proc. of PoEM, pp. 236–250 (2013)

29. Li, J., Mirkovic, J., Wang, M., Reiher, P., Zhang, L.: SAVE: Source address validity enforcement protocol. In: Proc. of INFOCOM, vol. 3, pp. 1557–1566 (2002)

30. Li, N., Tripunitara, M.V., Bizri, Z.: On mutually exclusive roles and separation-of-duty. TISSEC **10**(2), 5 (2007)

31. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. IBM System Journal **46**(2), 335–361 (2007)

32. Mason, M.: Sample size and saturation in PhD studies using qualitative interviews. Forum: Qualitative Social Research **11**(3) (2010)

33. McCumber, J.: Information Systems Security: A Comprehensive Model. Proc. of NCSC (1991)

34. Menzel, M., Thomas, I., Meinel, C.: Security Requirements Specification in Service-Oriented Business Process Management. In: Proc. ARES, pp. 41–48 (2009)

35. Monakova, G., Brucker, A.D., Schaad, A.: Security and safety of assets in business processes. In: Applied Computing 27, pp. 1667–1673 (2012)

36. Moody, D.: The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transaction on Software Engineering **35**, 756–779 (2009)

37. OASIS: Web Services Business Process Execution Language. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html (2007). URL `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html`

38. OASIS: eXtensible Access Control Markup Language (XACML)Version 3.0. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (2013)

39. OMG: BPMN 2.0 (2011). URL `http://www.omg.org/spec/BPMN/2.0`

40. Parker, D.: Our Excessively Simplistic Information Security Model and How to Fix It. ISSA pp. 12–21 (2010)

41. Parker, D.B.: Fighting computer crime - a new framework for protecting information. Wiley (1998)

42. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. J. Manage. Inf. Syst. **24**(3), 45–77 (2007)

43. Rasmussen, J.L., Singh, M.: Designing a Security System by Means of Coloured Petri Nets. In: Proc. of ICATPN, pp. 400–419 (1996)

44. Rodríguez, A., Fernández-Medina, E., Piattini, M.: A BPMN extension for the modeling of security requirements in business processes. IEICE Transaction on Information and Systems **90**(4), 745–752 (2007)

45. Sadiq, S., Governatori, G., Namiri, K.: Modeling Control Objectives for Business Process Compliance. In: Proc. of BPM, pp. 149–164 (2007)

46. Saleem, M., Jaafar, J., Hassan, M.: A Domain- Specific Language for Modelling Security Objectives in a Business Process Models of SOA Applications. AISS **4(1)**, 353–362 (2012)

47. Salnitri, M., Dalpiaz, F., Giorgini, P.: Aligning Service-Oriented Architectures with Security Requirements. In: Proc. of OTM, pp. 232–249 (2012)

48. Salnitri, M., Dalpiaz, F., Giorgini, P.: Modeling and Verifying Security Policies in Business Processes. In Proc. of BPMDS pp. 200–214 (2014)

49. Salnitri, M., Giorgini, P.: Modeling and Verification of ATM Security Policies with SecBPMN. In Proc. of SHPCS (2014)

50. Samarati, P., Vimercati, S.: Access Control: Policies, Models, and Mechanisms. In: In FOSAD, vol. 2171, pp. 137–196 (2001)

51. Schmidt, R., Bartsch, C., Oberhauser, R.: Ontology-based Representation of Compliance Requirements for Service Processes. In: Proc. of CEUR (2007)

52. SecBPMN Website: SecBPMN website, last visited September 2014. http://www.secbpmn.disi.unitn.it (2014)

53. Simon, R., Zurko, M.: Separation of duty in role-based environments. In: Proc. of CSFW, pp. 183–194 (1997)

54. Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., Mcdermid, J., Paige, R.: Large-scale Complex IT Systems. Communication of the ACM **55**(7), 71–77 (2012)

55. Störrle, H.: VMQL: A Visual Language for Ad-hoc Model Querying. J. Vis. Lang. Comput. **22**, 3–29 (2011)

56. The Apache Software Foundation: Apache Rampart website, last visited August 2014. http://axis.apache.org/axis2/java/rampart/ (2014)

57. W.M.P. van der Aalst: Formalization and Verification of Event-Driven Process Chains. Information and Software Technology **41**(10), 639 – 650 (1999)

58. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction (2000)

59. Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. JSA **55**(4), 211 – 223 (2009)

60. Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in bpmn. In: G. Alonso, P. Dadam, M. Rosemann (eds.) Business Process Management, *Lecture Notes in Computer Science*, vol. 4714, pp. 64–79. Springer Berlin Heidelberg (2007)