Detecting Conflicts between Data-Minimization and Security Requirements in Business Process Models

Qusai Ramadan[1], Daniel Strüber[1], Mattia Salnitri[2], Volker Riediger[1] and Jan Jürjens[13]

[1] University of Koblenz-Landau, Koblenz, Germany
{qramadan, strueber, riediger, juerjens}@uni-koblenz.de
[2] Politecnico di Milano, Milano, Italy
{mattia.salnitri}@polimi.it
[3] Fraunhofer-Institute for Software and Systems Engineering ISST, Dortmund, Germany

Abstract. Detecting conflicts between security and data-minimization requirements is a challenging task. Since such conflicts arise in the specific context of how the technical and organizational components of the target system interact with each other, their detection requires a thorough understanding of the underlying business processes. For example, a process may require anonymous execution for a task that writes data to a secure data storage, where the identity of the writer is needed for the purpose of accountability. To address this challenge, we propose an extension of the BPMN 2.0 business process modeling language to enable: (i) the specification of process-oriented data-minimization and security requirements, (ii) the detection of conflicts between these requirements based on a catalog of domain-independent anti-patterns. The considered security requirements were reused from SecBPMN2, a security-oriented extension of BPMN 2.0, while the data-minimization part is new. SecBPMN2 also provides a graphical query language called SecBPMN2-Q, which we extended to formulate our anti-patterns. We report on feasibility and usability of our approach based on a case study featuring a healthcare management system, and an experimental user study.

Keywords: conflicts, security, data-minimization, BPMN

1 Introduction

Protecting the privacy of users has become a key activity in companies and governmental organizations. A important requirement for privacy is *data-minimization* [14, 34], that is, to minimize "*the possibility to collect personal data about others*" and *"within the remaining possibilities, [to minimize] collecting personal data*" (Pfitzmann et al. in [28], p.6). In addition to security concepts such as confidentiality, five data-minimization concepts, namely, *Anonymity*, *Pseudonymity*, *Unlinkability*, *Undetectability* and *Unobservability*, are considered as fundamental for protecting the users' privacy. These concepts were first defined by Pfitzmann et al. [28] and later included in the ISO 15408 standard of common criteria for information technology security evaluation [16].

Although privacy-enhancing technologies [35] address specific data-minimization needs, privacy breaches often do not come from loopholes in the applied protection technologies, but from conflicting requirements on the business level of the target system [6, 13–15, 25]. For example, in healthcare, users have strong privacy concerns about

how and for what purpose their health information is handled, which may interfere with an organization's documentation responsibilities for ensuring complete accountability. Various approaches have been proposed that deal with security and data-minimization requirements in a unified framework from the early stages of development [11, 17, 26, 8]. However, these approaches focus on the identification of security and data-minimization requirements in the elicitation phase without analyzing trade-offs or detecting conflicts between them. The final output is usually a set of textual requirements. Relying on textually-specified security and data-minimization requirements to manually uncover conflicts between them is a difficult and error-prone task. The main reasons for that are:

First, conflicts between the data-minimization and security requirements depend on the context of how the technical and organizational components of the target system interact with each other. Specifically, conflicts not only result from trade-offs between requirements related to the same asset in the system (e.g., anonymous vs. accountable execution of a task), but also from those related to different assets. For example, a task may be required to be executed anonymously, while writing data to a secure data storage where the identity of the writer must be known for accountability reasons. The detection of such conflicts requires an understanding of the underlying business processes and their included interactions between security and data-minimization requirements. However, no existing approach supports the modeling of data-minimization requirements in business process models in the first place.

Second, a single data-minimization concept may have varied meanings based on *what* (which of the system assets) and from *who* (i.e., adversary type) to protect. These variations make it hard to decide whether two specific requirements are conflicting. For example, providing fully anonymous execution of a specific task hinders the ability of the system to keep the task's executor accountable, leading to a conflict. In contrast, providing *partial* anonymity by means of using pseudonyms is not conflicting with accountability. So far, there exists no approach to detect such conflicts between security and data-minimization requirements in the design of a concrete system.

To address these challenges, we propose an extension of the Business Process Modeling Language (BPMN, [1]), supporting: (i) the specification of process-oriented data-minimization and security requirements in BPMN models, (ii) the detection of conflicts between security and data-minimization requirements based on a catalog of domain-independent anti-patterns. While the security annotations were reused from the security-oriented BPMN extension SecBPMN2 [33], our approach is the first to directly support modeling data-minimization requirements in BPMN models. It is also the first to support automatic conflict detection between specified security and data-minimization requirements in BPMN models. To express the anti-patterns, we extended SecBPMN2-Q, a graphical query language for BPMN models. We validate our approach using a case study based on a healthcare management system, and an experimental user evaluation.

The paper is organized as follows. Sec. 2 provides the necessary background. Sec. 3 introduces our BPMN extension. Sec. 4 presents the considered types of conflicts and our approach to detect them. Sec. 5 presents the tool support for our approach. Sec. 6 and 7 are devoted to the validation based on a case study and user evaluation. Sec. 8 and 9 discuss related work and conclude, respectively.

2 Background

We introduce the fundamental data-minimization concepts used in our work, and a BPMN-oriented security engineering approach whose security concepts we reused.

Data-minimization concepts. Pfitzmann et al. [28] define five data-minimization concepts that can be refined into privacy requirements for the target system [8, 11, 17, 26]. (i) *Anonymity* is the inability of an adversary to sufficiently identify a subject within a set of subjects, called the anonymity set. (ii) *Pseudonymity* is a special case of anonymity where a pseudonym is used as an identifier for a data subject other than one of the data subject's personal identifiable information. (iii) *Unlinkability* is the inability of an adversary to sufficiently distinguish whether two items of interests (IOIs, e.g., subjects, messages, actions, ...) within a system are related or not. (iv) *Undetectability* is the inability of an adversary to sufficiently distinguish whether an IOI is exist or not. By the definition [28], undetectability of an IOI can only hold against outsider adversary (i.e., neither being the system nor one of the participants in processing the IOI). (v) *Unobservability* is the undetectability of an IOI against all subjects uninvolved in it (i.e., outsider adversary) and the anonymity of the subject(s) involved in the IOI against other subject(s) involved in that IOI (i.e., insider adversaries).

Business process model-based security engineering [20] is a promising research direction in the field of security engineering. The key idea is to extend graphical business process modeling languages such as BPMN [1] to supports the modeling and analysis of procedural security requirements as early as during the design phase. Among various approaches [20], only the work proposed in [31] considers a data-minimization requirement, namely anonymity. However, further fundamental data-minimization requirements such as unlinkability and undetectability were not addressed yet.

To capture conflicts between security and data-minimization requirements, a unified framework for modeling both types of requirements is needed. Compared to other approaches, we found that the security concepts from SecBPMN2 [33] offer the following advantages: (i) In contrast to the work in [7, 10, 18, 22, 31, 32, 37] which support only a restricted set of security aspects, SecBPMN2 offers 10 security annotations, namely *accountability*, *auditability*, *authenticity*, *availability*, *confidentiality*, *non-repudiation*, *integrity*, *separation of duties*, *binding of duties*, and *non-delegation*. *Accountability* specifies that the system should hold the executors of the activities responsible for their actions. *Authenticity* imposes that the identity of a given activity's executor must be verified, or that it should be possible to prove a given data object as genuine, respectively. *Audibility* indicates that it should be possible to keep track of all actions performed by an executor or accessor of an activity, data object, or message flow. *Non-delegation* specifies that an activity shall be executed only by assigned users. *Binding of duties* and *Separation of duties* requires that the same person or different persons should be responsible for the completion of two related tasks, respectively. *Confidentiality* and *Integrity* indicate that only authorized users are allowed to read or modify data from a given activity, message flow, or data object, respectively. *Availability* indicates that it should be possible to ensure that an activity, data object, or message flow is available and operational when are required by authorized users.

Reusing these concepts allows us to study interactions between a comprehensive set of security and data-minimization requirements, enabling a powerful approach to conflict detection. (ii) While other works [27, 36] use textual stereotypes to enrich business process models with security requirements (e.g., «confidentiality»), SecBPMN2 represents security annotations using graphical icons [33]. The example in Fig. 2, explained in detail later, illustrates the specification of confidentiality, accountability, non-repudiation and binding-of-duty requirements in a BPMN model (icons in orange). Graphical annotations have the potential to increase the complexity of the resulting business process models less than textual ones would do [24], and as consequence, may contribute to the usability of our approach.

In addition, SecBPMN2 provides a query language for specifying queries that can be matched against a given SecBPMN model, called SecBPMN2-Q [33]. We reuse and extend this query language in our approach for specifying conflicts as anti-patterns.

3 Modeling Data-Minimization and Security Requirements

We propose a BPMN extension for specifying data-minimization and security requirements. Our support for data-minimization requirements is new, while the security-specific elements are reused from the security-oriented BPMN extension SecBPMN2. We first present a running example and then a complete description of our extension.

Running Example. Fig. 1 represents a business process in the context of healthcare management. A patient makes use of a telemedicine device to receive an over-distance healthcare service. A patient can also evaluate the service through an online evaluation portal. Executors of a business process are represented by *pools* and *swimlanes* such as Tele-medicine Device and System Portal respectively. Communication between pools is represented by *message flows*; the content of such communications are *messages*: for example, Tele-medicine Device sends the message measures to System Portal. Atomic activities are represented with *tasks*, for example Send alert. *Data Objects* provide information about what activities require to be performed and/or what they produce, for example electronic healthcare record (EHR). A *data association* is a directional association used to model how data is written to or read from a data object. For instance, the Check the case task needs the EHR data object to be read. Events are represented with circles. *Start* events and *End* events mark the initial and terminal points. *Catch events* represent points in a business process where an event needs to happen, for example at this time. Gateways specify deviations of the execution sequence: the gateway measures>=config? allows either the right or left branch to be executed.

Security concepts are represented with orange icons. *Confidentiality* is associated to message flows, meaning that the content of the message is to be preserved and not to be accessed by unauthorized users, respectively. *Accountability* is associated to Submit evaluation meaning that the task's executor must be monitored. Our new data-minimization concepts, discussed below, are represented with yellow icons.

Data-minimization annotations. To allow users to enrich business process models with data-minimization requirements, we extended BPMN's *artifact* class with four concrete data-minimization concepts namely, *anonymity*, *undetectability*, *unlinkability* and *unobservability*. The meta-model of these concepts is shown in Fig. 2: gray parts

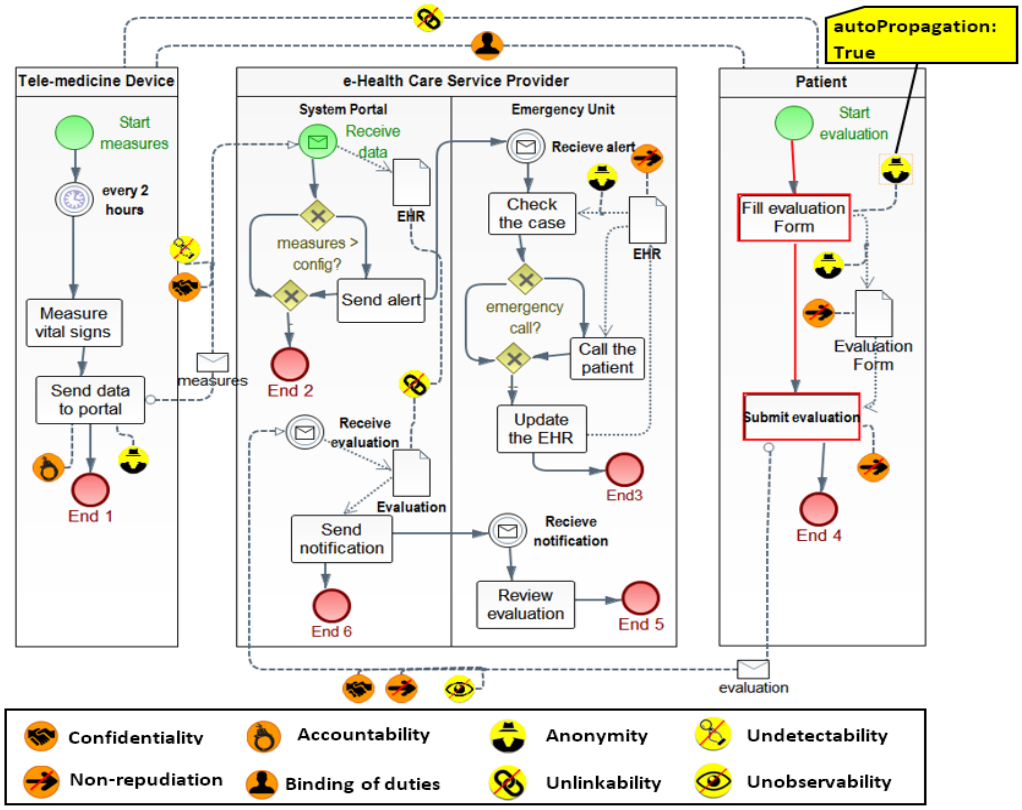

Fig. 1. Running example: Specifying data-minimization and security requirements in a healthcare business process.

represent SecBPMN2 elements; white parts are new elements. Since an additional concept described by Pfitzman et al., *pseudonymity*, is a special case of anonymity, we use one annotation for both concepts. An attribute called *level* captures the required anonymity level (i.e., full anonymous vs pseudonymous). Using one annotation to represent related concepts is recommended to reduce graphical complexity [20]. A special type of association called *SecurityAssociation* is used to link security annotations with elements in the business process model. Additional details are captured using attributes and references, describing in particular the items of interests (IOIs) and adversary perspectives, as introduced in Sect. 2. In this section, we focus on the meta-model elements being relevant for conflict detection, leaving the discussion of others (in particular, the specification of *Mechanisms* and *DataSubjectRoles*) outside the scope of this paper.

To reduce specification overhead, data-minimization annotations have an attribute *autoPropagated* which supports the propagation of the requirement to selected other elements in the model. Four cases are possible, depending on the type of the element the annotation is linked with: (1) For an activity, the requirement is propagated to all following tasks in the same lane. (2) For a message flow, the requirement is propagated

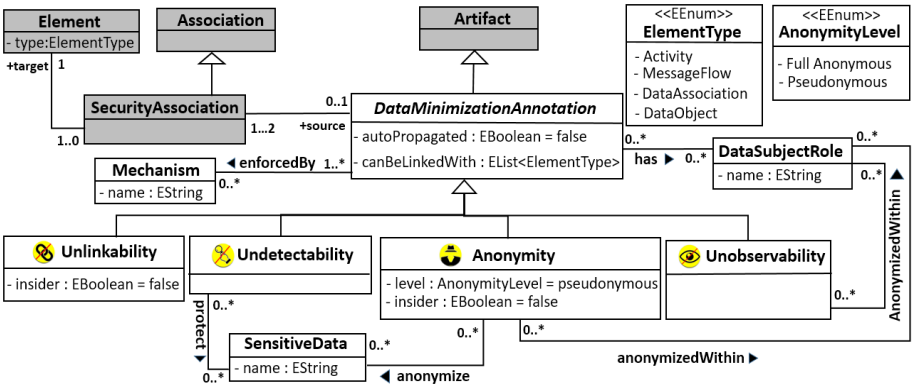

Fig. 2. Meta-model of our BPMN extension. Attributes show their default values.

to all message flows that goes from the source-Pool of the considered messageFlow to its target Pool. (3) For a data input association, the requirement is propagated to all data input associations that read data from that data object in the same lane. (4) For a data output association, the requirement is propagated to all data output associations that write data to that data object in the same lane.

We designed the graphical syntax of the data-minimization annotations following Moody's guidelines for increasing the usability of modeling languages [24]. The data-minimization annotations share two common visual aspects with security annotations in SecBPMN2: they all have a solid texture, and a circular shape; they differ in their fill color, using yellow instead of orange. We believe that having different colors for security and data-minimization annotations contributes to usability.

In the rest of this section, all data-minimization annotations are defined. Each of them is defined in terms of one or more variants, one for every type of BPMN element it can be linked with. We mapped each annotation to a restricted list of element types to avoid overlapping meaning of different data-minimization annotations. For example, two messages cannot be linked to each other as related (i.e., unlinkability) if they are sent anonymously (i.e., anonymity). Therefore, having both unlinkability and anonymity annotations for message flows would be redundant.

Anonymity comes in four variants for the different BPMN elements it can be linked to: (i) *Anonymity-Activity* specifies that the executor of the task should be anonymous with respect to a given adversary perspective. (ii) *Anonymity-MessageFlow* specifies that the sender of the message should be anonymous with respect to a given adversary perspective. (iii) *Anonymity-DataOutputAssociation* specifies that the task should not write personal identifiable information to the data object. (iv) *Anonymity-DataInputAssociation* specifies that the task should only retrieve an anonymized variant of the data object.

The exact meaning of this annotation can be shaped by two attributes: The attribute *level* specifies the required anonymity level (i.e., fully anonymous or pseudonymous). In some scenarios, the system requires that the executor of an activity should be accountable, and thus, pseudonyms should be used to de-identify the executor of the activity. The attribute *insider* specifies against who to protect. The considered adversary type is either just outsider (*false*) or both outsider and insider (*true*). We define the outsider

adversary as any entity being part of the surrounding of the system considered. The insider is any entity being part of the system considered, including the system itself.

The example model Fig. 1 shows three anonymity annotations associated to different BPMN elements. Consider, for example, the one associated to the Fill evaluation form activity. This annotation specifies that a patient shall be able to execute the Fill evaluation form task anonymously within the set of all patients without being identifiable by either outsider or insider adversaries. Since the requirement is propagated, the same requirement applies to the Submit evaluation task.

Unlinkability comes in two variants, depending on which BPMN elements the annotation linked with. (i) *Unlinkability-Process* can be linked with two pools/lanes to specify that an adversary of the given type shall not be able to link two executed processes as related. In other words, if linked to two pools, this annotation imposes that a subject may make use of multiple services without allowing others to link these uses together as related [28]. (ii) *Unlinkability-DataObject* can be linked with two data objects to specify that, from the given adversary perspective, it should not be possible to link the two data objects as related. Since unlinkability can only be applied to two specific processes or data objects, it cannot be propagated to other elements. The attacker type is specified using the *insider* attribute, in the same way as in the *anonymity* case.

The example model includes two unlinkability annotations. Consider, for example, the unlinkability annotation associated with the two data objects namely, EHR and Evaluation. This annotation specifies that both outsider and insider adversaries must not be able to link an EHR and an Evaluation data objects as related.

Undetectability has three variants, depending on the BPMN elements it is linked with. (i) *Undetectability-Activity* specifies that an adversary should not be able to detect whether an activity is executed or not. (ii) *Undetectability-MessageFlow* specifies that an adversary cannot sufficiently distinguish a true messages from a false ones (e.g., random noise). (iii) *Undetectability-DataInputAssociation* specifies that a task should not be able to distinguish whether a piece of data is exists in a data object or not.

The example model shows an undetectability annotation linked with the message flow between the Send data to portal task and the Receive data start event. The annotation specifies that outsider adversaries must not be able to distinguish true messages sent over the message flow between the Send data to portal task and Receive data event from a false ones. In other words, at a specific time, an outsider adversary cannot detect whether the Tele-midicine device is sending data or not.

Unobservability can only be applied to message flows, leading to precisely one variant called *Unobservability-MessageFlow*: the sender of the message should be anonymous with respect to insider adversaries and the message itself should not be detectable by outsider adversaries.

The example model includes an unobservability annotation linked with the message flow between the Submit evaluation task and the Receive evaluation catch event. This annotation specifies that an outsider adversaries should not be able detect true messages being sent over the message flow from false ones, and the patient who sent messages over the message flow must be anonymous to the insider adversary.

4 Conflict Detection

Uncovering conflicts during the design of business processes is of vital importance to avoid privacy violations and expensive fixes in the later development phases. Detecting conflicts manually in annotated business process models is a challenging task, especially in real cases where business process frequently are composed of many tasks. We present an automated conflict detection techniques which takes as input a BPMN model with data-minimization and security annotations, and reports a list of *conflicts* and *potential conflicts*. The former represent definitive mismatches between two requirements; the latter may result in conflicts under certain circumstances. Consequently, our tool shows conflicts as *errors*, and potential conflicts as *warnings* to the user.

Conflicts between security and data-minimization requirements occur in two flavors: First, requirements related to the same asset in the system may be conflicting. For example, consider the *accountability* and *anonymity* annotations linked with the Send data to portal task in Fig. 1. For accountability, the system needs to track the executor of this task's responsibility, while the anonymity annotation specifies that the executor should be fully anonymous against insider adversaries. Second, requirements related to different, dependent assets may be conflicting. For example, in Fig. 1, consider the *anonymity* and *non-repudiation* annotations linked with the Fill evaluation form task and the Evaluation form data object, respectively. The former imposes that an executor to the Fill evaluation form task should be fully anonymous against insider adversaries; the latter indicates that an accessor to the Evaluation form data object should not be able to deny that she accessed the Evaluation form. Since the Fill evaluation form task writes data to the Evaluation form, a conflict is reported.

Potential conflicts as considered in our work result from control flows between activities with specified requirements. For example, Fig. 1 includes a path between the *anonymity*-annotated fill evaluation form task and the *non-repudiation*-annotated Submit evaluation task. Such situations not necessarily give rise to an actual conflict. For instance, imagine a flow between two tasks where the first task allows a customer to anonymously use a service and the second task allows the service provider to prevent a customer from being able to deny his payment for receiving a service. In this situation, it may be sufficient for a service provider to prove that a customer performed the payment task without uncovering which service a customer is paying for, and as a consequence, preserve the customer anonymity. Such potential conflicts should be reported and discussed during the design of the business process models.

Automated conflict detection using anti-patterns. We propose an automated conflict detection technique that relies on encoded knowledge about conflicts and potential conflicts between pairs of requirements. Specifically, we propose a catalog of *conflict anti-patterns* which are matched against the given business process model in order to detect conflicts and potential conflicts. Our patterns are formulated in a specialized query language, which extends an existing query language called SecBPMN2-Q [33]. SecBPMN2-Q supports custom graphical queries enriched with security requirements that can be matched to SecBPMN2 models, usually for verification purposes. We extended SecBPMN2-Q so that it supports our new data-minimization annotations as well, allowing us to specify conflicts as anti-patterns that can automatically detected.

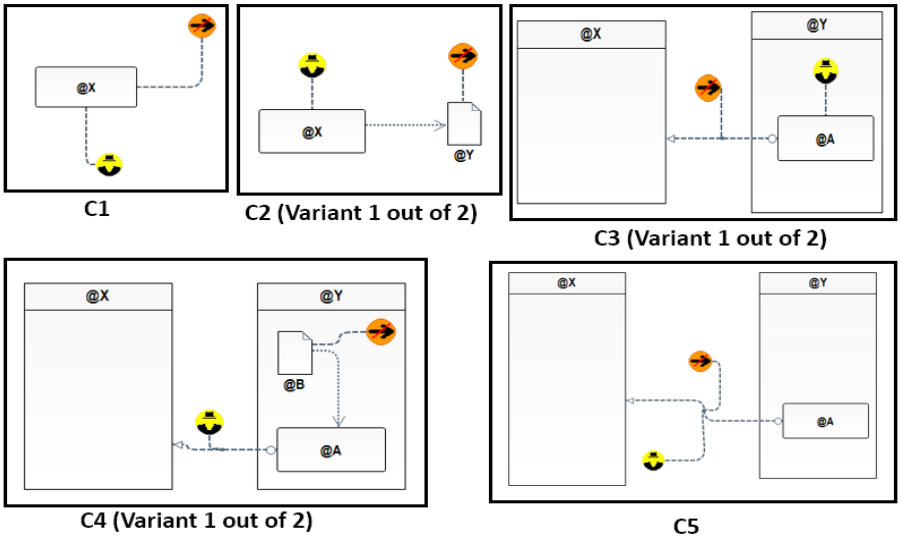

Fig. 3. Conflicts C1–C5 between *non-repudiation* and *anonymity* as anti-patterns.

Fig. 3 shows a selection of anti-patterns defined using our SecBPMN2-Q extension. Together, the depicted anti-patterns represent all conflicts that can happen between *non-repudiation* and *anonymity*. The patterns include labels of the form "@X", which act as placeholders for element names, allowing us to formulate the anti-patterns in a domain-independent way. All anonymity annotations in Fig. 3 are specified with the following attributes: *{anonymity level=full anonymous, insider=true}*.

Consider, for example, conflicts C1 and C5 in Fig. 3. These conflicts arise when *non-repudiation* and *anonymity* annotations are linked to the same task or message flow, respectively. C1 can be matched to one place in the example model, which is highlighted in Fig. 1: The *anonymity* annotation of the Fill evaluation form is propagated to the Submit evaluation task, which is annotated with a *non-repudiation* annotation. In contrast, C5 does not occur in the example model, since the model does not have an *anonymity*- and *non-repudiation*-annotated message flow. C2, C3 and C5 each come in two variants, resulting from duality: the direction of the data object call (read or write) can be inverted, and the assignment of requirements to elements can be swapped.

Fig. 4 shows three anti-patterns specifying potential conflicts between *anonymity* and *non-repudiation*. In these patterns, we use a *walk* relation (illustrated using an edge with double arrowhead), which is defined for pairs of activities, events or gateways. It allows to match all pairs of elements in the input model for which there is a path between the source and the target element. Note that 3 out of 8 overall potential conflicts for the considered pair of requirements are shown. Two additional cases called PC3 and PC5 are formed in analogy to C3 and C5 in Fig. 3; three additional variants arise from duality like in the discussion of the conflicts. Again, all anonymity annotations are specified with the following attributes: *{anonymity level=full anonymous, insider=true}*.

The potential conflict PC1 in Fig. 4 includes a *non-repudiation*-annotated and an *anonymity*-annotated task between which a path exists. PC4 specifies a path between an

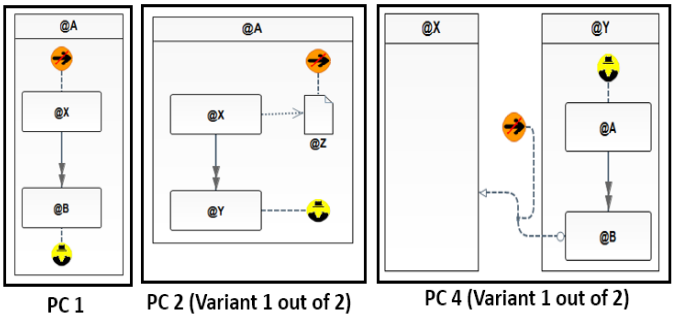

Fig. 4. Potential conflicts between *non-repudiation* and *anonymity* as anti-patterns.

anonymity-annotated task and another task that sends messages over a *non-repudiation*-annotated message flow. Both situations may lead to a conflict, depending on the actual circumstances in the system.

From matching potential conflicts PC1 and PC4 to the example model in Fig. 1, two warnings will be reported: (i) There is a path between the *anonymity*-annotated Fill evaluation form task and the *non-repudiation*-annotated Submit evaluation task, thus violating PC1. (ii) There is a path between the *anonymity*-annotated Fill evaluation form task and the Submit evaluation task which sends messages over a *non-repudiation*-annotated message flow, leading to a violation of PC4.

Catalog of a domain-independent conflicts. As mentioned in Sect. 2, SecBPMN2 in fact supports 10 different security requirements. Similar to our data-minimization annotations, the same security annotation can be linked to different BPMN elements. We analyzed all possible situations where conflicts or potential conflicts between a security and a data-minimization annotation may happen. For each identified situation, we specified an anti-pattern using our extension of SecBPMN2-Q.

We now give an overview of the resulting catalog of anti-patterns. A more detailed account is found in [30]. Table 1 shows all pairs of requirements for which we identified a (potential) conflict. Each cell shows the number of conflicts, plus the number of potential conflicts between the considered pair of requirements. For example, there are 8 conflicts and 8 potential conflicts for non-repudiation and anonymity. The origin of these numbers is explained in the previous descriptions of Fig. 3 and Fig. 4. The other numbers arise from the various possibilities of linking a data-minimization or a security annotation with other BPMN elements. In total our catalog contains 140 anti-patterns.

Considering conflicts and potential conflicts, *Accountability*, *Authenticity*, *Audibility*, and *Non-delegation* represent different requirements to keep insider users accountable for their actions. To preserve them, the identity of an action's executor must be verified. Therefore, similarly to *Non-repudiation*, all of these security concepts may have conflicts or potential conflicts with *Anonymity* (where required against insiders) and *Unobservability*, since part of its definition implies full anonymity against insiders.

Binding and *Separation of duties* can conflict with *Anonymity* if any of the activities to which they are applied also require to be executed anonymously. For instance, it will be hard, in case of *Binding of Duties*, to prove that two fully-anonymously executed

Table 1. Overview of conflict + potential conflict anti-patterns per pair of requirements.

Requirement pair	*Accountability*	*Authenticity*	*Auditability*	*Non-repudiation*	*Non-delegation*	*Binding of Duties*	*Separation of Duties*	*Anonymity*	*Confidentiality*	*Integrity*	*Availability*
Anonymity	2+2	6+6	8+8	8+8	2+2	1+0	1+0	4+5	0+8	0+11	0+11
Unlinkability	0+0	0+0	0+0	0+0	0+0	0+1	0+0	0+0	0+0	0+0	0+0
Unobservability	1+1	3+3	4+4	4+4	1+1	0+0	0+0	2+2	0+4	0+6	0+6

activities are executed by the same person or not. A potential conflict between the *Binding of duties* and *Unlinkability* is also possible: *Unlinkability* is linked to two pools and indicates that the two process executions should not be linked to each other as related. Therefore, it may conflict with *Binding of Duty*.

Confidentiality, *Integrity*, and *Availability* represent different requirements to allow authorized users to read, modify, or access a system asset, respectively. The satisfaction of these requirements relies on authorization, which, however, does not necessarily imply identification: The literature provides many techniques for performing authorization without uncovering the real identity of an action executor, for example, *zero-knowledge protocols* [23]. However, the system developers may choose to implement these requirements by a mechanism that relies on identification, such as access control, which may lead to conflicts with data-minimization requirements. Hence, a decision about whether a conflict arises cannot be made on the abstraction level of process models. Therefore, as shown in Table 1, we classified the interactions between these security requirements and the data-minimization requirements as potential conflicts.

In some cases, *Confidentiality* and *Integrity* are considered as supplementing requirements to *Anonymity* [15]. For instance, anonymity against outsider adversaries implies that the outsider adversaries should not be able to trace a message back to its sender. However, if the sent message contains personal identifiable information and it is sent in clear (i.e., without encryption), an outsider attacker can easily link the messages to its sender. Such kind of interactions can not be considered as conflicts or potential conflicts, and thus, they are omitted from Table 1.

Conversely, conflicts may not only occur between security and data-minimization requirements. Table 1 indicates that a particular *Anonymity* annotation might conflict with other *Anonymity* or *Unobservability* annotations. For instance, requiring full anonymous execution for an activity is in conflict with requiring the users to execute the same activity anonymously using their *pseudonyms*. *Undetectability*, by definition [28], only shields against outsider attackers. Therefore, it is omitted from the Table 1 since it does not give rise to conflicts with security or data-minimization requirements.

5 Tool Support

We developed a prototypical implementation of our work on top of STS [2], the supporting tool for the BPMN extension SecBPMN2 [33]. Our implementation supports the

two main contributions of this work: First, the modeling of data-minimization and security requirements in BPMN models, using a suitable model editor. Second, automated conflict analysis in data-minimization- and security-annotated BPMN models, based on our catalog of anti-patterns. The examples shown Fig. 1, Fig. 3 and Fig. 4 come from screenshots of our implementation. Our conflict detection approach takes as an input a security- and data-minimization-annotated BPMN model. The output is a set of textual messages that describe the detected conflicts. On demand, the conflict can be highlighted in the model. For example, the highlighted path in Fig. 1 is the result of selecting the conflict message that describes the PC1 anti-pattern in Fig. 4. Our implementation is available online at `http://www.sts-tool.eu/downloads/secbpmn-dm/`.

6 Case Study

To study the feasibility our approach, we applied it in a healthcare scenario. We extended a teleconsultations healthcare management case study from the Ospedale Pediatrico Bambino Gesù, a pediatric Italian hospital. The case study was part of the *VisiOn* research project [3]. The main objective of VisiOn consists in increasing the citizens awareness on privacy. The final outcome of the project was a platform that can be used by public administrations and companies to design their systems, using privacy as a first-class requirement.

The teleconsultations case study described a situation where a patient EHR can be transfered from the OPBG system to specialists in another hospital for a teleconsultations purposes. In this scenario many security requirements are considered (e.g., confidentiality, accountability) but the privacy preferences were more related to data anonymization. In this paper, we extended this scenario to to cover situations where data-minimization plays an important role not only protecting the users data but also their activities and communications. To this end, we modeled a process featuring an over distance healthcare service, an excerpt being shown in Fig. 1. Using our approach, as explained in Sect. 3, we were able to enrich the model with data-minimization requirements that represent privacy preferences for patients.

For conflict detection, we annotated the model with security requirements that represent security needs from the system point of view. Assessing the accuracy of conflict detection based on this model required a ground truth. To this end, we manually analyzed the model and identified 8 conflicts and 20 potential conflicts, a subset being discussed in Sect. 4. Applied to the model, our conflicts detection technique precisely detected these expected 8 conflicts and 20 potential conflicts. The used version of the model with all data-minimization and security requirements can be found in `https://github.com/QRamadan/conflictsDetection/`.

7 Usability validation

Our main contribution is twofold: we provide support for the enrichment of the BPMN models with data-minimization requirements, and the detection of conflicts between data-minimization and security requirements. As a preliminary evaluation for both contributions, we performed a user experiment that focused on two research questions:

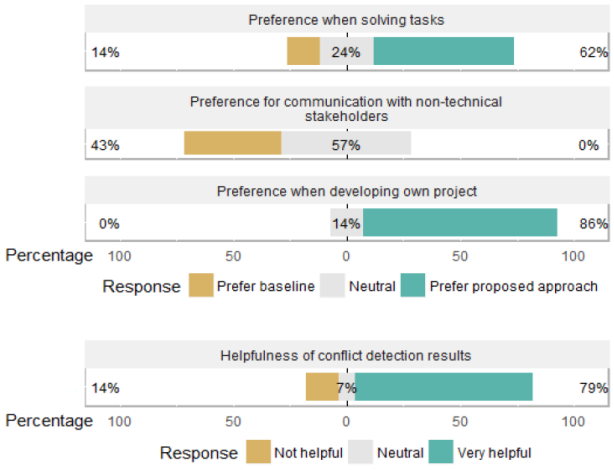

Fig. 5. Results: Preference scores for notation (RQ1) and helpfulness of output (RQ2).

(RQ1) *How usable are our data-minimization annotations, compared to textual requirement specifications?* **(RQ2)** *How useful is the conflict detection output?* The focus of RQ1 is on our new data-minimization annotations; the security annotations from SecBPMN2 were already evaluated in an earlier work [33]. As participants, we recruited 6 doctoral students and 1 post-doc from three institutions. We asked the participants to rate their experience in process modeling (in particular, BPMN), privacy, and security using 5-point Likert scales. Six participants rated their BPMN experience as 3 or 4. Six participants rated their security expertise as 3 or 4. The self-assessed privacy experience was 4 for one participant, 3 for four participants and 2 for two participants. In total, this distribution approximates the knowledge of the intended user group.

The set-up of our experiment involved a questionnaire with embedded model excerpts based on the model from our case study. Models were included in two versions: with visual data-minimization and security annotations (*proposed approach*), and with textual data-minimization requirement description and visual security annotations (*baseline*). The participants received a description sheet of all used annotations. For RQ1, participants were asked to complete comprehension tasks and, afterward, to state their subjective notation preference for solving our tasks, for communication with non-technical stakeholders, and for developing their own projects. For RQ2, participants were asked to identify conflicts manually. Afterwards, we showed them the output of our tool and asked them to rate its helpfulness to be used it for this task. We also asked the participants for informal feedback using a free-form input field.

The summarized results are shown in Fig 5; the shown scores accumulate the answers to multiple related questions. Regarding RQ1, we found that the participants mostly preferred our proposed approach for solving the tasks (62%), and for developing their own project (86%). One participant stated that *"with the extension, it's a lot easier to detect the model elements that are affected by a requirement than with the text version (have to find the relevant elements and correlate text and model)"*. However, for

communication with non-technical stakeholders, all participants gave a neutral (57%) or negative (43%) answer. An explanation offered by a participant was that to "*fully understand the effect of the annotations is very hard. This is the reason why I rated both variants equally usable for non-technical audience*"'. Regarding RQ2, the majority of participants rated the conflict detection results as very helpful for the identification of conflicts (79%). Two of the participants stated that the detection results pointed them to conflicts that they had not noticed when inspecting the models manually. In summary, our results give a promising outlook for the usability of our approach.

Threats to Validity. Owing to the limited sample size, we relied on descriptive statistics, leaving a comprehensive user study with a more rigorous statistical analysis to future work. The actual usefulness in practice may significantly depend on the considered model, of which we only considered one in our experiment. Usability was assessed through a subjective questionnaire rather than objective performance measures. This threat can be addressed using a different kind of experiment, as we plan to conduct in the future. Finally, we only considered the comprehension, rather than the editing of models. On the other hand, understanding is a necessary part of any editing process.

8 Related Work

Conflicts between security and data minimization. To the best of our knowledge, no existing approach supports conflict detection between security and data-minimization requirements. Hansen et al. in [15] defined six privacy and security goals for supporting the privacy needs of users. The authors considered a subset of the data-minimization concepts in [28], namely *anonymity* and *unlinkability*, and discuss their relationships. However, conflicts are discussed on the conceptual level, while in our work, we argue that the specific conflicts arising in a system can be identified by analyzing the data-system's minimization and security requirements. The perspective papers of Ganji et al. [13] and Alkubaisy [6] highlight the importance of detecting conflicts between security and privacy requirements, for data-minimization requirements in particular. Both papers discuss the components required for a potential approach, however, without providing a complete solution. Ganji et al. [13] envision a realization based on the SecureTropos framework as future work.

Data-minimization-aware approaches. Various works in security requirements engineering aim to specify privacy requirements using the data-minimization concepts proposed in [28]. In Deng et al.'s LINDDUN framework [11], both misuse cases and data-minimization requirements can be identified by mapping predefined threat-tree patterns to the elements of a data-flow diagram. Kalloniatis et al. [17] propose the Pris methodology, which maps data-minimization and other security concepts to a system's organizational goals to identify privacy requirements. Pris introduces privacy-process patterns that describe the effect of privacy requirements to organizational processes. Mouratidis et al. [26] present a conceptual framework that combines security and data minimization concepts, and show its use to specify details about privacy goals such as the involved actors and threats. Beckers et al. [8] propose a privacy-threats analysis approach called ProPAn that uses functional requirements modeled in the problem-frame approach to check if insiders can gain information about specific stakeholders. Ahma-

dian et al. [4] support a privacy analysis for system design models, based on the four privacy key elements of purpose, retention, visibility and granularity. Since none of these approaches considers conflicts between data-minimization and security requirements, our approach can be seen as complementary: Their output can be used as input for our approach to allow the enrichment of the business process models with data-minimization and security requirements and then to perform conflict detection.

Diamantopoulou et al. [12] provide a set of privacy process patterns for data-minimization and security concepts, aiming to provide predefined solutions for different types of privacy concerns in the implementation phase. In addition to textual description of the patterns, BPMN design patterns were provided to guide operationalization at the business process level. This work is complementary to ours, as it focuses on the implementation of data-minimization requirements, rather than on the detection of conflicts.

9 Conclusions and Future Work

We proposed an extension of the BPMN modeling language to enable the specification of data-minimization and security requirements in a unified framework. Based on this extension, we introduce a technique for conflict detection between the specified requirements. Our technique analyses data-minimization- and security-enriched models based on a catalog of a domain-independent anti-patterns, which we formulated in an extension of a graphical query language called SecBPMN2-Q. We validated our approach in a case study based on a healthcare management system, and an experimental user study.

In the future, we aim to formally validate the completeness of our technique. Encoding the semantics of data-minimization and security requirements using graph transformations would allow us to apply formal conflict detection [9, 19] for that purpose. We also aim to extend our approach to support the resolution of conflicts. As a first step, we aim to extend existing work in [33] to ensure that the enriched model is aligned with the collected security and data-minimization requirements. This will allow us to identify and fix unintentional conflicts (e.g., errors during the enrichment of the model with security and data-minimization requirements). Although a fully automated process would be appreciated, we believe that the resolution of actual conflicts (e.g., between two requirements related to different views of system stakeholders) is a sensitive issue that requires human intervention, a further challenging task that involves reasoning on the privacy impact of different solution strategies [5, 21]. Once that all conflicts are resolved, the system design typically needs to be aligned with the specified privacy and security requirements, a challenge that can benefit from the use of model transformation technology [29]. Finally, we intend to perform a comprehensive user experiment to study the usability and validity of our approach in a broader setting.

Acknowledgment

We wish to thank Paolo Giorgini and the STS tool development team in the University of Trento for providing us with access to the source code of the STS tool. We also wish to thank the participants of our study. This research was partially supported by: (I) The Deutsche Akademische Austauschdienst (DAAD), (II) the project "Engineering Responsible Information Systems" financed by the University of Koblenz-Landau.

References

1. BPMN 2.0. http://www.omg.org/spec/BPMN/2.0/.
2. STS. http://www.sts-tool.eu/downloads/secbpmn-dm/.
3. VisiOn. http://www.visioneuproject.eu/.
4. Amir Shayan Ahmadian, Daniel Strüber, Volker Riediger, and Jan Jürjens. Model-based privacy analysis in industrial ecosystems. In *European Conference on Modelling Foundations and Applications*, pages 215–231. Springer, 2017.
5. Amir Shayan Ahmadian, Daniel Strüber, Volker Riediger, and Jan Jürjens. Supporting Privacy Impact Assessment by Model-Based Privacy Analysis. In *ACM Symposium on Applied Computing*. ACM, 2018. To appear.
6. Duaa Alkubaisy. A framework managing conflicts between security and privacy requirements. In *International Conference on Research Challenges in Information Science*, pages 427–432. IEEE, 2017.
7. Wihem Arsac, Luca Compagna, Giancarlo Pellegrino, and Serena Elisa Ponta. Security Validation of Business Processes via Model-Checking. *ESSoS*, 6542:29–42, 2011.
8. Kristian Beckers, Stephan Faßbender, Maritta Heisel, and Rene Meis. A problem-based approach for computer-aided privacy threat identification. In *Annual Privacy Forum*, pages 1–16. Springer, 2012.
9. Kristopher Born, Leen Lambers, Daniel Strüber, and Gabriele Taentzer. Granularity of conflicts and dependencies in graph transformation systems. In *International Conference on Graph Transformation*, pages 125–141. Springer, 2017.
10. Achim D Brucker, Isabelle Hang, Gero Lückemeyer, and Raj Ruparel. SecureBPMN: Modeling and enforcing access control requirements in business processes. In *ACM Symposium on Access Control Models and Technologies*, pages 123–126. ACM, 2012.
11. Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.
12. Vasiliki Diamantopoulou, Nikolaos Argyropoulos, Christos Kalloniatis, and Stefanos Gritzalis. Supporting the design of privacy-aware business processes via privacy process patterns. In *International Conference on Research Challenges in Information Science*, pages 187–198. IEEE, 2017.
13. Daniel Ganji, Haralambos Mouratidis, Saeed Malekshahi Gheytassi, and Miltos Petridis. Conflicts Between Security and Privacy Measures in Software Requirements Engineering. In *International Conference on Global Security, Safety, and Sustainability*, pages 323–334. Springer, 2015.
14. Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14(3), 2011.
15. Marit Hansen, Meiko Jensen, and Martin Rost. Protection goals for privacy engineering. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 159–166. IEEE, 2015.
16. ISO and IEC. Common Criteria for Information Technology Security Evaluation - Part 2 Security functional components. In *ISO/IEC 15408, International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC)*, 2012.
17. Christos Kalloniatis, Evangelia Kavakli, and Stefanos Gritzalis. Addressing privacy requirements in system design: the PriS method. *Requirements Engineering*, 13(3):241–255, 2008.
18. Wadha Labda, Nikolay Mehandjiev, and Pedro Sampaio. Modeling of privacy-aware business processes in BPMN to protect personal data. In *ACM Symposium on Applied Computing*, pages 1399–1405. ACM, 2014.
19. Leen Lambers, Daniel Strüber, Gabriele Taentzer, Kristopher Born, and Jevgenij Huebert. Multi-Granular Conflict and Dependency Analysis in Software Engineering based on Graph

Transformation. In *International Conference on Software Engineering*. IEEE/ACM, 2018. to appear.
20. Curtis L Maines, David Llewellyn-Jones, Stephen Tang, and Bo Zhou. A cyber security ontology for BPMN-security extensions. In *International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing*, pages 1756–1763. IEEE, 2015.
21. Rene Meis and Maritta Heisel. Systematic identification of information flows from requirements to support privacy impact assessments. In *International Joint Conference on Software Technologies*, volume 2, pages 1–10. IEEE, 2015.
22. Michael Menzel, Ivonne Thomas, and Christoph Meinel. Security requirements specification in service-oriented business process management. In *International Conference on Availability, Reliability and Security*, pages 41–48. IEEE, 2009.
23. Austin Mohr. A survey of zero-knowledge proofs with applications to cryptography. *Southern Illinois University, Carbondale*, pages 1–12, 2007.
24. Daniel Moody. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6):756–779, 2009.
25. Anthony Morton and M Angela Sasse. Privacy is a process, not a PET: A theory for effective privacy practice. In *Proceedings of the 2012 workshop on New security paradigms*, pages 87–104. ACM, 2012.
26. Haralambos Mouratidis, Christos Kalloniatis, Shareeful Islam, Marc-Philippe Huget, and Stefanos Gritzalis. Aligning Security and Privacy to Support the Development of Secure Information Systems. *J. UCS*, 18(12):1608–1627, 2012.
27. Jutta Mülle, Silvia von Stackelberg, and Klemens Böhm. *A security language for BPMN process models*. KIT, Fakultät für Informatik, 2011.
28. Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, unobservability, pseudonymity, and identity management. *TU Dresden and ULD Kiel, Tech. Rep*, 2011.
29. Qusai Ramadan, Mattia Salnitri, Daniel Strüber, Jan Jürjens, and Paolo Giorgini. From secure business process modeling to design-level security verification. In *International Conference on Model Driven Engineering Languages and Systems*, pages 123–133. IEEE, 2017.
30. Qusai Ramadan, Daniel Strüber, Mattia Salnitri, Volker Riediger, and Jan Jürjens. Detecting Conflicts between Data-Minimization and Security Requirements in Business Process Models (Long Version). https://figshare.com/s/664b1c79c55130a44e79, 2018.
31. Alfonso Rodríguez, Eduardo Fernández-Medina, Juan Trujillo, and Mario Piattini. Secure business process model specification through a UML 2.0 activity diagram profile. *Decision Support Systems*, 51(3):446–465, 2011.
32. M Saleem, J Jaafar, and M Hassan. A domain-specific language for modelling security objectives in a business process models of soa applications. *AISS*, 4(1):353–362, 2012.
33. Mattia Salnitri, Fabiano Dalpiaz, and Paolo Giorgini. Modeling and verifying security policies in business processes. In *Enterprise, Business-Process and Information Systems Modeling*, pages 200–214. Springer, 2014.
34. Sarah Spiekermann and Lorrie Faith Cranor. Engineering privacy. *IEEE Transactions on software engineering*, 35(1):67–82, 2009.
35. GW Van Blarkom, JJ Borking, and JGE Olk. Handbook of privacy and privacy-enhancing technologies. *Privacy Incorporated Software Agent (PISA) Consortium, The Hague*, 2003.
36. José L Vivas, José A Montenegro, and Javier López. Towards a business process-driven framework for security engineering with the UML. In *International Conference on Information Security*, pages 381–395. Springer, 2003.
37. Christian Wolter and Andreas Schaad. Modeling of task-based authorization constraints in BPMN. *Business process management*, pages 64–79, 2007.