Preserving Compliance with Security Requirements in Socio-Technical Systems

Mattia Salnitri, Elda Paja, and Paolo Giorgini

University of Trento, Trento, Italy,
{mattia.salnitri, elda.paja, paolo.giorgini}@unitn.it

Abstract. Socio-technical systems are an interplay of social (humans and organizations) and technical components interacting with one another to achieve their objectives. Security is a central issue in such complex systems, and it cannot be tackled only through technical mechanisms: the encryption of sensitive data while being transmitted, does not assure that the receiver will not disclose them to unauthorized parties. Therefore, dealing with security in socio-technical systems requires an analysis: (i) from a social and organizational perspective, to elicit the objectives and security requirements of each component; (ii) from a procedural perspective, to define how the actors behave and interact with each other. But, socio-technical systems need to adapt to changes of the external environment, making the need to deal with security a problem that has to be faced during all the systems' life-cycle. We propose an iterative and incremental process to elicit security requirements and verify the socio-technical system's compliance with such requirements throughout the systems' life cycle.

Keywords: Socio-Technical Systems, Security Requirements, Security Policies, Compliance, Business Processes

1 Introduction

Socio-technical systems are complex systems where social (human and organizational) and technical components interact with each other to achieve common objectives. Examples of socio-technical systems are healthcare systems, smart cities, air traffic management, etc. In a smart city citizens constantly exchange information with e-govermental systems such as tax-payment. The amount of information exchanged in such systems is considerable, and quite often part of such information is sensitive, i.e., should be protected. Apart from information, other types of assets are relevant when dealing with socio-technical systems. In a smart city, examples of other assets are the services being offered, such as the tax verification and monitoring service.

An analysis of security aspects is crucial to avoid severe consequences [7, 3, 13] such as loss of privacy and law infringement. Security is typically dealt with technical security mechanisms. For instance, encryption mechanisms are used to protect data confidentiality. However, such mechanisms cannot protect

information from misuse by authorized users. As a result, security analysis in socio-technical systems calls for an analysis of social and organizational aspects along technical ones.

An analysis of social and organizational aspects allows to capture the objectives of each stakeholder and their business policies, how stakeholders pursue their objectives, to then check if some of these business policies might threaten stakeholders' assets (or those of their interacting parties) with respect to security. For example, citizens might want the non-disclosure of their social security number, but the employees of the tax-payment system may need to use this information for statistical purposes. In this case, there is a conflict between stakeholders' need, which can be detected only through an analysis of social and organizational aspects. Starting from stakeholders' needs, and after dealing with possible conflicts one can obtain a consistent security requirements specification.

But an analysis of social and organizational aspects to security requirements engineering is not enough, verifying whether the socio-technical system is compliant with such requirements is crucial too. To perform such verification, we need to analyse the overall socio-technical system, the involved stakeholders, their behavior and interactions with others to check whether the procedures and activities underlying the system comply with the specified security requirements. Indeed, the analysis of procedural aspects allows to verify if the security requirements are satisfied by the socio-technical system. For instance, in a smart city, citizens require the non-disclosure of their social security numbers. An analysis of the flow of activities and the information flow, via business processes [12], allows verifying whether there is a flow of information containing the social security number, which does not start from the citizen, capturing in this way a breach with respect to non-disclosure of such information.

The need socio-technical systems have to adapt, has a high impact over their capability to remain compliant with security requirements. For example, the business process executed to coordinate the tax-payment system with the provision of the smart city services, is drastically changed because new technologies are employed to minimize the effort of the smart city employees. Before the deployment of the adapted business process, all security policies shall be verified, to avoid security breaches.

In this paper, we propose a process to guide security designers in capturing security requirements in socio-technical systems and preserving compliance with them. As far as our knowledge goes, no similar processes have been proposed in the literature to guide security designers in maintaining business processes running in a socio-technical system compliant with social and organizational security requirements. Specifically, we rely on the STS-ml [8, 18] (Socio-Technical Security modeling language), an actor and goal-oriented modeling language, for the modeling of social and organizational aspects of socio-technical systems, and SecBPMN [25] (Secure BPMN), an extension of Business Process Modeling and Notation (BPMN) for modeling procedural aspects of socio-technical systems. The process proposed in this paper guides security designers in the specification of SecBPMN security policies from STS-ml security requirements,

and in maintaining compliance with security policies, while preserving in this way compliance with security requirements.

The paper is structured as follows. Section 2 gives an overview of the proposed process, while Section 3 provides a detailed description of the steps of the process, with references to the chosen languages. Section 4 discusses related work, and Section 5 summarizes the paper and concludes.

2 Incremental Design Process for Socio-Technical Systems

We propose an iterative and incremental process to verify the continuous compliance of evolving and adaptable socio-technical systems with security requirements for the said system. The process is iterative, because it cycles various times, and incremental, because it allows security requirement engineers to refine and extend the model during its iterations. It receives in input the security specification of a socio-technical system and, during its iterations, it ensures compliance with security requirements.

The process, illustrated in Fig. 1, is divided in two phases, the first phase is executed by security requirement engineers and it regards the elicitation of security requirements considering social and organizational aspects, while the second phase is executed by security designers and it regards verifying compliance via procedural aspects of socio-technical systems. The process can be used: (i) before deployment, to guide the definition of the business processes (procedural) executed by the socio-technical system; (ii) after the deployment, to help preserving compliance of the socio-technical system during its life-cycle.

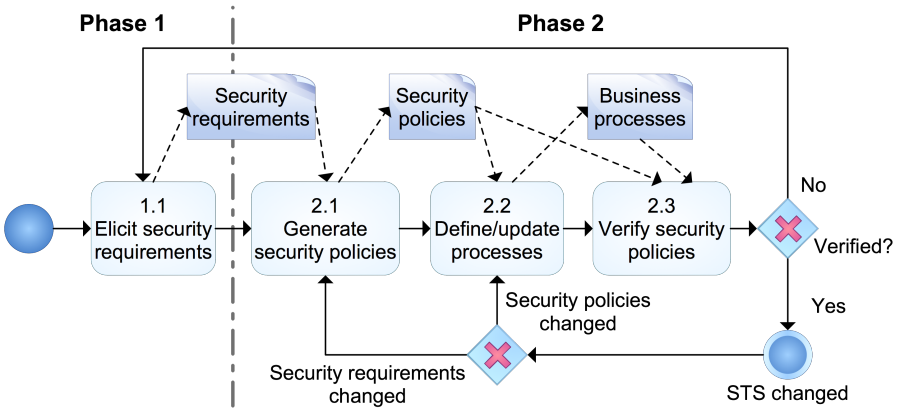

Fig. 1. Incremental Design Process

2.1 Phase 1

The first phase consists of only one activity, namely **1.1 Elicit security requirements**, and it is concerned with the extraction of security requirements for the considered socio-technical system. The set of security requirements is represented with the data object "Security requirements"; the dashed arrow from the activity to the data object means the activity creates or modifies this data object. Section 3.1 describes the elicitation activity in more detail.

2.2 Phase 2

The activities in the second phase are executed to verify if security requirements captured through Phase 1 are satisfied by the business processes of the socio-technical system. For this, security requirements are transformed in security policies, i.e., security constraints in terms of business process concepts.

The activity **2.1. Generate security policies** consists in generating security policies from security requirements. In this step, a set of transformation rules, presented in [27], is used to transform security requirements in security policies in a semi-automated fashion. This activity receives in input (see the incoming dashed arrow, Fig. 1) a set of security requirements and it results in a set of security policies (data object "Security policies") in output.

The activity **2.2. Define/update processes** consists in the definition of new business processes, or the modification of existing ones. This activity receives in input the security policies generated by the previous activity: the definition or modification of processes will be guided by the security policies they should comply with. The activity produces a set of business processes, represented with the data object "Business processes".

The activity **2.3. Verify security policies** consists in the verification of the security policies, generated by activity **2.2**, against the business processes generated by activity **2.3**. This step is necessary, although some business processes have been created using security policies as guidelines, given that verifying compliance with security policies requires the complete set of business processes.

If at least one business process does not comply with security policies, either the security specification or the procedural design shall be changed. This is represented with an arrow from the gateway to the beginning of the process. Otherwise, the process waits for a change in the socio-technical system: if the business processes change, then the the verification step is executed, otherwise, if the security requirements change, the process restart from the beginning.

The order of execution of the steps described in the process is not prescriptive, rather should be considered as a guideline. In particular, the order of the second and the third activity could be swapped: frequently processes are defined before the definition of security policies. In this case, the definition of the processes will not be guided by the security policies.

3 The Process in Action

We describe how the process is executed with the help of a motivating example. We use the SWIM[1] Air Traffic Management (ATM) socio-technical system [2] as a motivating example for our process. It consists of a large number of autonomous and heterogeneous components (stakeholders), such as pilots, airports personnel, national airspace managers, meteo services, radars, etc., which interact with each other to enable air traffic management operations. In such a complex system, ensuring security is critical, for security leaks may result in severe consequences on, for example, safety. For instance, a successful attack to the control tower, the core component of every airport, can paralyse an airport for hours or days, with severe consequences on managing flights and consequently on passengers.

3.1 Phase One: Eliciting Security Requirements

The elicitation of security requirements is concerned with the analysis of social and organizational aspects in the said socio-technical system to derive a consistent security requirements specification. To execute this activity we have adopted STS-ml [8, 18] (Socio-Technical Security- modeling language), an actor and goal-oriented security requirements modelling language for socio-technical systems. STS-ml was chosen because: (1) it is specifically thought for socio-technical systems, relating security to interaction, (2) it supports a rich set of security requirements, while providing a clearer ontological foundation than existing approaches [11, 16]. Moreover, STS-ml is fully supported by the STS-Tool [19] on modelling, analysing, and deriving a consistent set of security requirements.

In STS-ml, requirements models are created through three views: (i) the *social view*—represents the main stakeholders (in terms of actors) together with their objectives (via goals) and the interactions they enter in the socio-technical system; (ii) the *information view*—represents stakeholders' informational assets and their representation via documents; and (iii) the *authorization view*—represents the authorizations that actors grant to others over their information. Fig. 2 shows a partial STS-ml model of the motivating example.

Social view. Actors in STS-ml are modeled in terms of (i) agents—concrete entities that are already known at design-time (e.g., *Immigration office*), and (ii) roles—abstract entities representing a class of participants (e.g., *Web-Service*). Roles can be adopted (played) by different agents at runtime. An actor's rationale captures actors' goals, and how they are achieved via AND/OR goal decompositions (e.g., the root goal of the *Immigration Office* is *Immigration monitored*). Moreover, to achieve their goals, actors might need to *read* or *modify* documents, as well as create (*produce*) new documents (e.g., *Immigration Office* reads document *Visa* to achieve goal *Visa checked*). Most importantly, the social view captures actors' social interactions via two social relationships:

[1] The System Wide Information Management (SWIM) [2]

[2] This scenario is a variant of Case Study *B* of the FP7 EU Funded Project Aniketos http://www.aniketos.eu

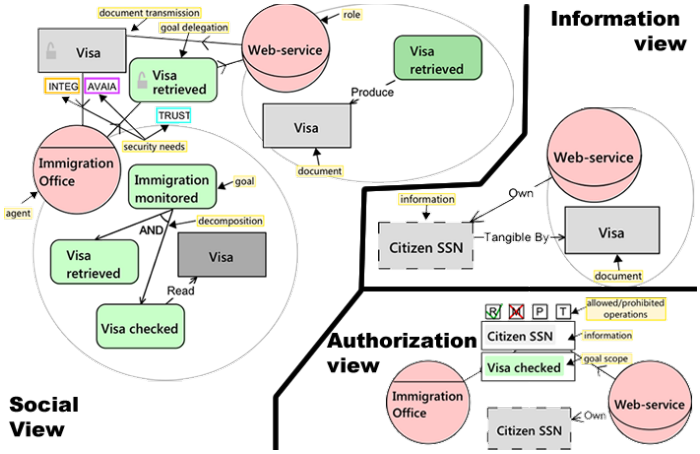

Fig. 2. STS-ml model of an ATM scenario

goal delegation and *document transmission*. STS-ml allows actors to express their concerns about security (security needs) over the interactions they enter to then derive security requirements with respect to confidentiality, integrity, availability, accountability, reliability, and authenticity.

Information view. STS-ml considers information a first class citizen, considering most security issues are concerned with the protection of information. Information owners are the ones concerned with the protection of information. Therefore, information ownership is a crucial aspect to model. In STS-ml, the relationship *own* relates an actor to the information that it owns.

But, information may be available in various forms. Thus, STS-ml distinguishes between information and its representation in form of *documents*. Documents become relevant from a security point of view because of the information they might represent. Thus, the purpose of the information view, apart from representing information entities and their respective owners, is to link together the documents actors use and exchange in the social view with their informational content. This link is drawn through "Tangible by" relationships, which indicate that an information entity is represented by a document. In Fig. 2, information *Citizen SSN* (Social Security Number) is made tangible by document *Visa*.

Authorization view. STS-ml allows capturing the permission and prohibition flow over information, on top of capturing information flow. An adequate representation of permissions and prohibitions is crucial to establishing whether information is used and exchanged in compliance with security requirements.

The authorisation view represents the permissions and/or prohibitions on information that actors grant one to another. An authorization relationship details: (i) the permissions/prohibitions on the operations actors can perform over information (Read, Modify, Produce, Transmit) while manipulating documents for the achievement of their goals; (ii) information entities for which permis-

sions/prohibitions are specified; (iii) the scope of authorisation, referring to the goal(s) for the fulfillment of which permission/prohibition is specified; and finally, (iv) transferrability, specifying whether permissions can be further granted to others (not applicable to prohibitions). In Fig. 2, the *Web-Service* authorizes the *Immigration Office* to use *Citizen SSN* in the scope of goal *Visa checked*.

3.2 Phase Two: Generating Secure Procedures

In order to verify compliance with security requirements, we generate security policies, define the process that will be executed in the socio-technical system, and verify compliance of security policies against the business processes. We have chosen the SecBPMN (Secure BPMN) [25] framework, for it offers support throughout these activities. Indeed, SecBPMN is aimed at modeling business processes with security aspects, modeling security policies, and verifying if one or more business processes are compliant with these security policies. The language is composed of: SecBPMN-ml (SecBPMN- modeling language), a modeling language for business processes; SecBPMN-Q (SecBPMN - Query), a graphical query language for specifying security policies in terms of SecBPMN-ml elements; and a software component, which verifies compliance of business processes with security policies. Each SecBPMN component is used in an activity of the second phase of the process in Fig. 1. The rest of the section describes how SecBPMN is used in each activity of **Phase 2** [3].

Activity 2.2. Define/update processes. In this activity business processes of a socio-technical system are defined using SecBPMN-ml [25], which extends BPMN with security concepts about information assurance and security defined in [6]. There are many proposals that extend BPMN with security concepts, e.g., [20, 28], but they are focused on a restricted set of security concepts. SecBPMN-ml, on the other hand, covers, as far as our knowledge goes, the most comprehensive set of security concepts.

The expressiveness of SecBPMN-ml permits security designers to define which are the security mechanisms that should be used in the implementation and execution of each activity. For example, it is possible to specify that the communication of a data object between two activities will be encrypted.

Fig. 3 shows part of a SecBPMN-ml model of a business process used in the ATM socio-technical system, where users can use different web-interfaces to select the best option for a flight, buy tickets, and perform most of the bureaucratic processes required to take the flight.

Activity 2.1. Generate security policies. This activity consists in generating security policies from STS-ml security requirements, in a semi-automated fashion [27], using SecBPMN-Q. For example, the security requirement of integrity attached to the "Visa" document in Fig. 2, can be transformed in the SecBPMN-Q security policy shown in Fig. 4.

[3] Note that we do not follow the flow of the process, but rather present the activities following a more natural description for SecBPMN, swapping activities 2.1. and 2.2.

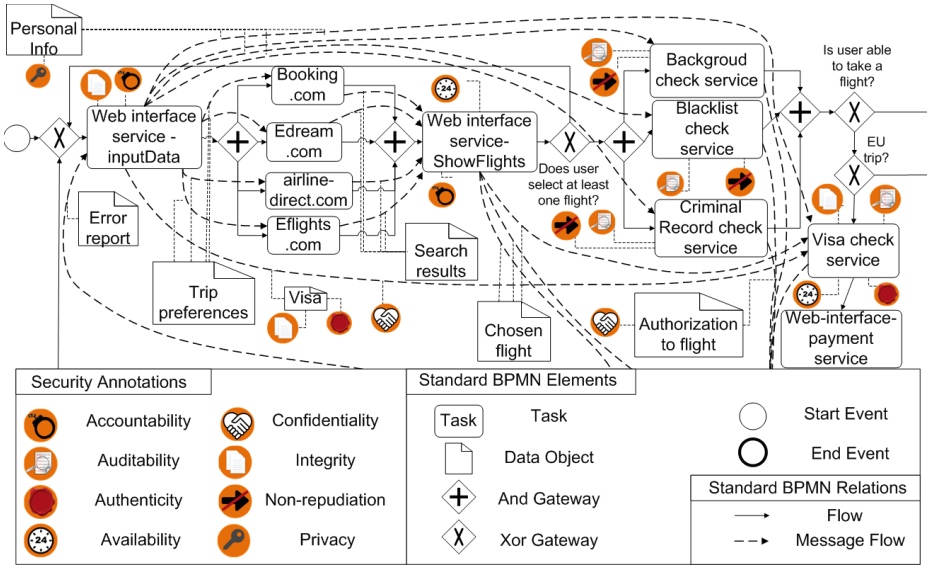

Fig. 3. Example of a business process modeled using SecBPMN-ml

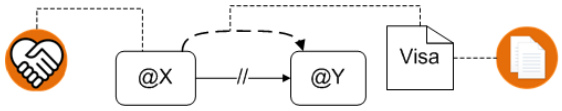

Fig. 4. Example of a security policy modeled using SecBPMN-Q

The graphical security policy in Fig. 4 is composed of two activities labeled with "@X" and "@Y", while the "@" symbol is used to match any activities. The two activities are linked with a path relation (the arrow with two slashes in the middle), which matches all the business processes where the first activity, marked with "@X" is executed before the second activity, marked with "@Y". The security policy is enriched with a message flow (represented as a dashed arrow), which exchanges a data object called "Visa". When executed, this security policy will match any message flow between two activities that exchange the "Visa" data object. The confidentiality annotation requires the communication channel to assure the data object will be received only by authorized users. Similarly, the integrity security annotation attached to the "Visa" data object, imposes the data object to be protected by unauthorized modifications.

SecBPMN-Q is essential for the transformation of security requirements in security policies. In previous work [26], we have demonstrated that it is possible to transform the most used security requirements in SecBPMN-Q security policies. This activity uses the transformation rules we provided in [26] to support the generation of security policies.

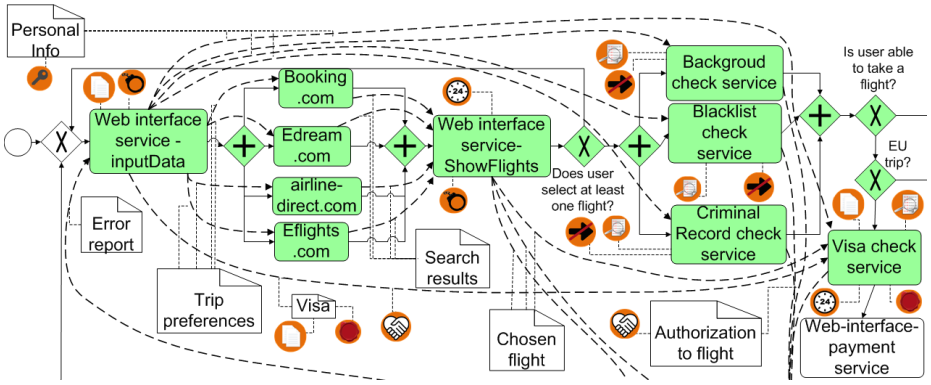

Fig. 5. Example of a path (highlighted in green) that satisfies the security policy showed in Fig. 4

Activity 2.3. Verify security policies. This activity consists in verifying if one or more business processes, modeled with SecBPMN-ml, comply with SecBPMN-Q security policies. This activity is straightforward for toy examples, as for the business process in Fig. 3 and the security policy in Fig. 4. However, in real-world scenarios, as the overall ATM case study [1], where business processes can contain hundreds of elements, it is infeasible to verify security policies manually. The software[4] provided with SecBPMN framework supports automated analysis to verify if a SecBPMN-Q security policy is satisfied by one or more SecBPMN-ml business processes.

Automated analysis allows to highlight the path that complies with the security policy, see Fig. 5 for the security policy in Fig. 4, where (i) the first activity of the path "Web interface service - inputData" is linked with a message flow to the last activity of the path"Visa check service"; (ii) the message flow is used to exchange the data object "Visa" and it assures confidentiality of the transferred data object; (iii) integrity and authenticity of the "Visa" data object are preserved. Assuming that the properties of the security annotations of the security policy are less restrictive than the properties of the business process, the path, and consequently the business process, complies with the security policy.

4 Related Work

In the years past, several approaches have been proposed to address the verification of requirements in business processes [5, 20, 28, 23, 15]. However, as far as our knowledge goes, there are no approaches that cover the overall security requirements engineering and verification process proposed in this paper.

[4] http://www.secbpmn.disi.unitn.it

In the following, we describe the most known approaches, while highlighting the differences with our approach considering the various phases.

Modeling BPMN with security concepts. As far as approaches dealing with security aspects are concerned, many graphical modeling languages extending BPMN [17] have been proposed. Ad-hoc notations are used in SecureBPMN [5] proposed by Brucker et al to capture security and compliance requirements. Other extensions of BPMN also rely on security annotated business process modelling [20, 28, 23, 15] similarly to our approach. However, differently from existing approaches, ours allows the definition of custom security policies. Instead, existing approaches employ software engines which use models created with the respective languages to check a fixed set of hard coded security policies. Examples of such engines include [28, 24, 22].

Modeling security policies. Graphical query languages have been proposed to check if a process satisfies a query, which can be interpreted as a policy. For instance, BP-QL (Business Process - Query Language) [4] and BPQL (Business Process Query Language) [9] allow to graphically define queries and check which business processes satisfy the queries. These two query languages are not based on BPMN, which makes their applicability and, most importantly, their learning process slower than that of, for example, SecBPMN-Q that is based on the well-known standard.

Other approaches are built on formal mathematical concepts (e.g. first order logic, temporal logic, etc.), and can be used to define business processes and/or the queries. These languages are expressive enough to include in the model security concepts. For instance, the approach of Rushby [21] proposes a language and a framework that checks if the code of the software diverges from specified behaviors (i.e., policies). These approaches have a main drawback: low usability, since they are quite complex and require lot of effort for the formalization of both business processes and security policies. In the eye of real scenarios, whose dimensions get larger and larger, it is nearly impossible to model business processes with such languages.

Verification of security policies. Liu et al. [14] describe how to check the compliance between a set of formally expressed regulatory requirements and business processes. The approach is accompanied by a software that allows verifying the business process against these compliance rules through the use of model-checking technologies. Their approach uses Business process Execution language (BPEL) instead of BPMN, and it is not focused on security, but rather focus on regulatory compliance.

Ghose and Koliadis [10] enrich BPMN with annotations, and they calculate how much a business process deviates from another business process. Differently from our approach, theirs focuses only on the structural difference between processes, again with no consideration of security requirements.

5 Conclusions

Security is quite a relevant aspect in the design of socio-technical systems, where a security leak in a single component may threaten the whole system, and security violations might have severe consequences. We have proposed a process intended to help security requirement engineers and security designers in verifying and maintaining the satisfaction of social and organizational security requirements through the procedural design of secure socio-technical systems. The proposed process is based on the STS-ml [8, 18] for the elicitation of a consistent security requirements specification, and on SecBPMN [25] for the verification of the satisfaction of security requirements in a procedural way.

The need to follow the proposed process becomes particularly important in adaptive socio-technical systems, where the design of the business processes changes to adapt to external changes. We have shown how to capture security requirements through STS-ml models, map them to security policies, and verify their satisfaction by business processes of the socio-technical system using an example from the air traffic management domain. The proposed process builds on the assumption that the tasks (defined in the business process), and the security aspects will be enforced rightly.

Our ongoing and future work includes: (i) developing a software that integrates STS-Tool with the SecBPMN component in order to offer an integrated framework for the management of security requirements in socio-technical systems; (ii) conducting empirical evaluation with security experts, to validate the overall process, as well as the integration of STS-ml with SecBPMN.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no. 257930 (Aniketos).

References

1. Final report on aniketos on industrial case studies. Technical report, 2014. http://aniketos.eu/sites/default/files/downloads/Aniketos%20D6.4%20-%20Final%20report%20on%20Aniketos%20%20applied%20to%20industrial%20case%20studies.pdf.
2. 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/.
3. R. Anderson. *Security engineering: A guide to building dependable distributed systems*. Wiley, April 2008.
4. C. Beeri, A. Eyal, S. Kamenkovich, and T. Milo. Querying business processes with BP-QL. *Information Systems*, 33(6):477–507, September 2008.
5. A. D. Brucker, I. Hang, G. Lückemeyer, and R. Ruparel. SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes. In *Proc. of SACMAT'12*, pages 123–126.

6. Y. Cherdantseva and J. Hilton. A reference model of information assurance and security. In *Proc. of ARES '13*, pages 546–555.
7. R. Crook, D. Ince, L. Lin, and B. Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proc. of RE'02*, pages 203–205. IEEE.
8. F. Dalpiaz, E. Paja, and P. Giorgini. Security Requirements Engineering via Commitments. In *Proc. of STAST'11*, pages 1–8.
9. D. Deutch and T. Milo. Querying structural and behavioral properties of business processes. In *Database Programming Languages*, LNCS, pages 169–185. 2007.
10. A. Ghose and G. Koliadis. Auditing business process compliance. In *Proc. ISOC'07*, pages 169–180.
11. P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Modeling Security Requirements through Ownership, Permission and Delegation. In *Proc. of RE'05*, pages 167–176.
12. H.J. Johansson, P. McHugh, A.J. Pendlebury, and W.A. Wheeler. *Business Process Reengineering: Breakpoint Strategies for Market Dominance*. Wiley & Sons, 1993.
13. M. N. Johnstone. Security requirements engineering-the reluctant oxymoron. In *Proc. of Australian Information Security Management Conference*, page 5, 2009.
14. Y. Liu, S. Müller, and K. Xu. A static compliance-checking framework for business process models. *IBM Syst. J.*, 46(2):335–361, April 2007.
15. M. Menzel, I. Thomas, and C. Meinel. Security Requirements Specification in Service-Oriented Business Process Management. In *Proc. ARES '09*, pages 41–48.
16. H. Mouratidis and P. Giorgini. Secure Tropos: A Security-Oriented Extension of the Tropos methodology. *IJSEKE'07*, pages 285–309.
17. OMG. BPMN 2.0. http://www.omg.org/spec/BPMN/2.0, Jan 2011.
18. E. Paja, F. Dalpiaz, and P. Giorgini. Managing security requirements conflicts in socio-technical systems. In *Proc. of ER'13*, pages 270–283.
19. E. Paja, F. Dalpiaz, M. Poggianella, P. Roberti, and P. Giorgini. Specifying and reasoning over socio-technical security requirements with sts-tool. In *Proc. of ER'13*, pages 504–507.
20. A. Rodríguez, E. Fernández-Medina, and M. Piattini. A BPMN extension for the modeling of security requirements in business processes. *IEICE Trans. on Information and Systems*, 90(4):745–752, 2007.
21. J. Rushby. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering and System Safety*, 75:167–177, 2002.
22. S. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In *Proc. of BPM '07*, pages 149–164.
23. 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.
24. M. Salnitri, F. Dalpiaz, and P. Giorgini. Aligning service-oriented architectures with security requirements. In *Proc. of OTM'12*, pages 232–249.
25. M. Salnitri, F. Dalpiaz, and P. Giorgini. Modeling and verifying security policies in business processes. *In Proc. of BPMDS'14*, pages 200–214.
26. M. Salnitri and P. Giorgini. Modeling and Verification of ATM Security Policies with SecBPMN. *In proc. of SHPCS'14*.
27. M. Salnitri and P. Giorgini. Transforming socio-technical security requirements in secbpmn security policies. *In Proc. of IStar'14*.
28. C. Wolter, M. Menzel, A. Schaad, P. Miseldine, and C. Meinel. Model-driven business process security requirement specification. *JSA '09*, 55(4):211 – 223.