Secure and Compliant Implementation of Business Process-Driven Systems

Achim D. Brucker and Isabelle Hang

SAP AG, SAP Research, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
{achim.brucker, isabelle.hang}@sap.com

Abstract. Today's businesses are inherently process-driven. Consequently, the use of business-process driven systems, usually implemented on top of *service-oriented* or *cloud-based* infrastructures, is increasing. At the same time, the demand on the security, privacy, and compliance of such systems is increasing as well. As a result, the costs—with respect to computational effort at runtime as well as financial costs—for operating business-process driven systems increase steadily.
In this paper, we present a method for statically checking the security and conformance of the system implementation, e. g., on the source code level, to requirements specified on the business process level. As the compliance is statically guaranteed—already at design-time—this method reduces the number of run-time checks for ensuring the security and compliance and, thus, improves the runtime performances. Moreover, it reduces the costs of system audits, as there is no need for analyzing the generated log files for validating the compliance to the properties that are already statically guaranteed.

Keywords: Business Process Security, Secure Service Tasks, BPMN, Static Program Analysis

1 Introduction

Business-process driven systems form the backbone of most modern enterprises. As a consequence, process-models as such and Business Process Modeling (BPM) as a methodology are becoming more and more important, not only as a documentation artifact but also for controlling and steering the execution of business processes. Moreover, the number of businesses that operate in regulated markets, i. e., that need to comply to regulations such as Health Insurance Portability and Accountability Act (HIPAA) [15] in the health care sector or Basel II [6] in the financial sector, is increasing. Such compliance regulations along with the increased awareness of IT security result in the need for modeling, analyzing, and execution techniques for business processes that treat security, privacy, and compliance properties in business processes as first class citizen.

To meet these requirements, several approaches, e. g., [12, 20, 23, 27], have been suggested to integrate the security specification into process models. Only a few of these approaches use the security models for more than documentation

M. La Rosa and P. Soffer (Eds.): BPM 2012 Workshops, LNBIP 132, pp., pp. 662–674, 2012.

purposes. Usually, the approaches that use the security models besides documentations only provide means for generating access control configurations for monolithic workflow management systems and, therefore, are not adequate for modern service-oriented or cloud-based infrastructures.

Modern service-oriented or cloud-based infrastructures are usually operated by multiple parties and, moreover, comprise many technical layers (see Fig. 1):

- The *User Interface Layer* managing the interaction with end users, i. e., allowing them to claim new tasks, querying input, or displaying results.
- The *Business Object Layer* or *Service Layer* comprises all backend systems as well as (external) services providing the functionality required for implementing the service tasks.
- The *Business Process Layer* comprising a business process execution engine that links the user interface layer and the business object layer, i. e., for human tasks, the necessary interaction with users is triggered and for automated tasks, the calls to backend systems or, e. g., services.

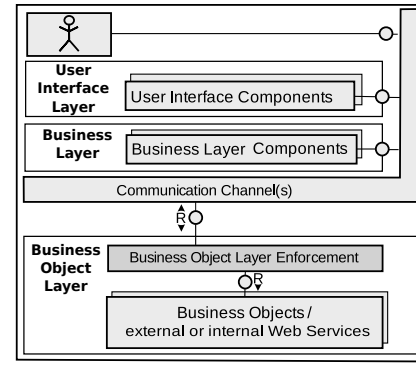

Fig. 1. System architecture or, potentially distributed or cloud-based, workflow management systems.

Each of these layers has to comply to the various compliance and security requirements to ensure that the overall systems complies to the security and compliance properties expressed at the business process level. Most works on integrating security aspects into business process models concentrate on the modeling as well as the process execution in the business process layer. In contrast, we concentrate on ensuring that the accompanying implementations (and system configurations) in the user interface layer as well as the business process layer comply to the process level security and compliance requirements.

In this paper, we present a novel approach that allows for statically checking the conformance of service implementations, i. e., on the source code level, to requirements specified on the business process level. As the compliance is statically guaranteed—already at design-time—this method reduces the number of run-time checks necessary and, thus, improves the runtime performances. Moreover, it reduces the costs of system audits, as there is no need for analyzing the generated log files for validating the compliance to the properties that are already statically guaranteed.

The rest of the paper is structured as follows: After introducing secure business process models using SecureBPMN in Sect. 2, we present a mapping from process level security and compliance requirements to the implementation level

in Sect. 3. In Sect. 4 we describe our prototype and in Sect. 5 we discuss related work and draw our conclusions.

2 Secure Business Processes: An Example

Modeling security properties, as a first class citizen of business processes, requires an integrated language for both security and business requirements. One option is the extension of a process modeling language with security concepts. In our work, we follow the meta-modeling approach for extending the meta-model of BPMN [22] with a security language, called SecureBPMN, that allows for specifying hierarchical role-based access control (RBAC) [1] as well as further security and compliance properties. The decision for a meta-model based approach is based on our previous experience in extending UML with RBAC [10].

Overall, SecureBPMN [12] enables the specification of security properties at a fine granular level. For example, separation of duty and binding of duty can restrict individual permissions (e. g., completing a task requires two *clerks* or one *manager*) rather than restricting the whole task.

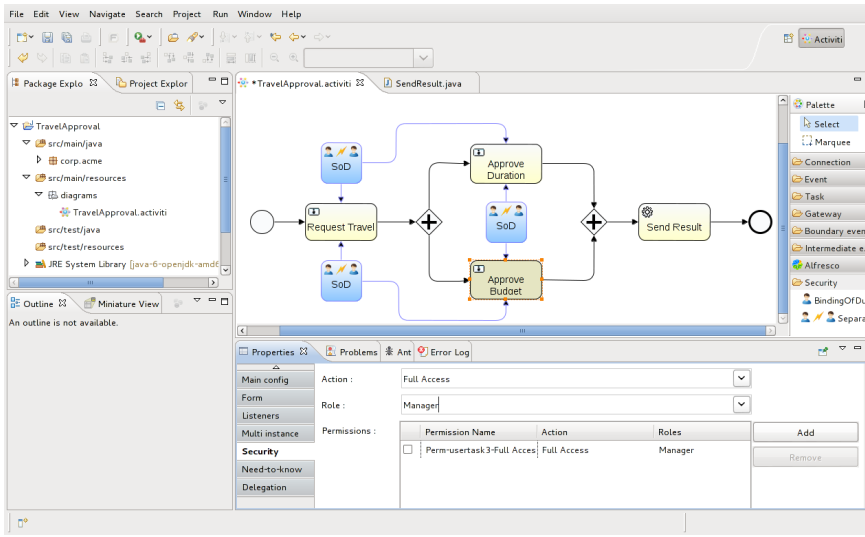

Fig. 2. The SecureBPMN modeling environment allows for specifying security requirements diagrammatically as well as using specialized user interfaces.

For example, consider a travel approval process in which the budget and the travel duration need to be approved by different managers. The main window in Fig. 2 illustrates such a process. This simple process requires already the following compliance and security requirements for a more detailed discussion of security requirements for process models):

```
<userTask id="Approve Duration">
  <extensionElements>
    <activiti:formProperty id="user_lastname" writable="false"/>
    <activiti:formProperty id="user_firstname" writeable="false"/>
    <activiti:formProperty id="travel_destination" writeable="false"/>
    <activiti:formProperty id="travel_duration" writeable="false"/>
    <activiti:formProperty id="travel_budget" writeable="false"/>
  </extensionElements>
</userTask>
```

Listing 1. User interface implementation for Request Travel using form properties.

- *Access Control:* Access to resources as well as actions need to be restricted to certain roles (e. g., clerks, managers) or subjects.
  In our example, we assume a simple role hierarchy containing the roles staff and manager where every member of the role manager is also a member or role staff. Moreover, the role staff has *full access* (e. g., is allowed to claim, execute, or cancel) for task Request Travel; members of the role manager have full access for the tasks Approve Duration and Approve Budget.
- *Separation of Duty:* More than one subject is required to successfully complete the process. Similarly, one can define *Binding of Duty* as the requirement that certain tasks need to be handled by the same subject.
  In our example, we use separation of duty to ensure that the subject requesting a travel is not allowed to approve the absence or the budget–even though he or she might be a member of the role manager.

While, in this example, most likely not necessary, we also assume the strict application of the need to know principle. In more detail:

- *Need to Know:* User should only be able to access the information that is required for their work.
  In our example, we apply the need to know principle to ensure that the manager approving the absence has only access to the duration of the travel and the manager approving the budget has only access to the travel costs.

Executing this process in the context of an enterprise system requires more than deploying the process model in a business process execution engine (recall Fig. 1): among others, user interfaces for the manual or human tasks, e. g., Request Travel or Approve Duration, need to be implemented. Listing 1 shows such an implementation of the user interface for Request Travel using an HTML like formalism, called *form properties*. Moreover, the internal flow of the service tasks, e. g., Send Result, needs to be implemented. Listing 2 shows such an implementation in Java that, e. g., can be used as the business logic of a web service.

These examples show that the implementation of business process-driven systems requires much more than only the business process model itself. Moreover, these artifacts are, compared to the high-level process models, quite low-level. Consequently, specifying security and compliance properties on the process level is not enough to ensure the secure and compliant operation of business process-driven enterprise systems.

```
package corp.acme;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import org.activiti.engine.delegate.JavaDelegate;
import org.activiti.engine.delegate.DelegateExecution;

public class SendResult implements JavaDelegate {
    @Override
    public void execute(DelegateExecution execution) throws Exception {
    String lastname    = (String) execution.getVariable("user_lastname");
    String firstname   = (String) execution.getVariable("user_firstname");
    String email       = (String) execution.getVariable("user_email");
    String destination = (String) execution.getVariable("travel_destination");
    String duration    = (String) execution.getVariable("travel_duration");

    if (firstname.equals("eve"))
      execution.setVariable("travel_budget",
          (new Integer(execution.getVariable("travel_budget")*2)).toString());

    sendEmail(firstname, lastname, email, destination, duration);

    }
}
```

Listing 2. Excerpt of the Java implementation of Send Result.

While there are works, e. g., [12, 28], that use process level security specifications for generating configurations for access control infrastructures such as XACML [21], we are not aware of any works that allow for checking process level security and compliance properties on the actual implementations of user interfaces or services. In the following, we present an approach that allows for checking the conformance of source code artifacts to process level requirement specifications. For many properties, such a conformance check can be done statically at implementation time. Thus, such checks help to improve the overall runtime of business process-driven systems as they reduce the number of runtime security checks required.

3 Mapping Secure Business Processes to Implementations

Checking process-level security and compliance specifications on the implementation level requires to link the implementation artifacts to process-level concepts such as tasks, data objects or process variables as well as the translation of the high-level to security and compliance requirements to low level concepts on the implementation level.

Tab. 1 summarizes the mapping from process-level security concepts to checks on the implementation level for three different implementations aspects. Namely, the implementation of service tasks and the two forms of implementing user interfaces: using domain specific languages such as the form properties provided by the Activiti BPM Platform as well as user interfaces implemented in generic programming languages such as Java or JavaScript. In more detail:

Table 1. Mapping process level requirements to the implementation level.

	Service Impl.	UI (Form Prop.)	UI (Java)
Access Control	AC check impl.	–	AC check impl.
Separation of Duty	AC check impl.	–	AC check impl.
Binding of Duty	AC check impl.	–	AC check impl.
Need to Know	proc. var. access	proc. var. access	proc. var. access
Confidentiality	dataflow	proc. var. access	dataflow

- For service task implementations, e. g., in Java, we can statically ensure that *access control* checks for enforcing access control in general and *separation of duty* (*binding of duty*) in particular are implemented. If we assume that the actual enforcement is based on a standard architecture for distributed systems, e. g., using an XACML policy decision point, we can only that the policy enforcement point is implemented correctly, but we still need to rely on a correctly implemented or generated policy.
  To ensure the conformance to the *need to know principle*, we can check the (read or write) access to process variables representing data objects. Similarly, we can check if a task implementation accesses confidential information. Still, checking *confidentiality* requirements requires a data flow analysis on the source code level. This allows, for example, also to ensure the compliance in situation in which a confidential information can be processed locally but it is not allowed to persist the data or transmit it to a third party. Even fine-grained requirements such as "this data object needs to be encrypted with a specific encryption algorithm and key length" can be checked on the implementation level.
- For user interface implementations using form properties, only a very limited set of properties can be checked. Namely, we can check if such a form accesses certain process variables (either read-only, write-only, or read and write) which allows us to detect violations against the *need to know* principle or data *confidentiality*. For all other requirements, in particular the access control, we need to rely either on checks by the business layer components or on the business object layer enforcement.
- For user interfaces written in a generic programming language (e. g., Java, JavaScript), we can apply the same checks as for service task implementation. Note that from a security perspective, one cannot rely on security checks in the user interface layer. Thus, these checks are only an addition to the checks in the other layers.

Note that this mapping is, in most cases, not complete, i. e., while the checks on the implementation level can detect violations to process level security policies, they do not, in general, guarantee the conformance to all aspects of the process-level security requirements.

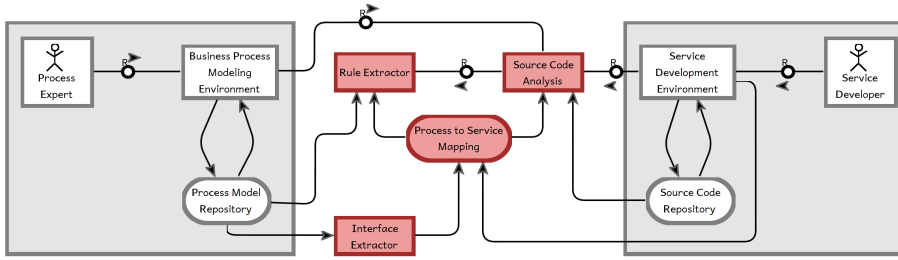

Fig. 3. Architecture of the Analysis Engine for Checking Process level Requirements on the (Service-) Implementation-level.

4 Design and Implementation

We implemented our method in a prototype using the Activiti BPM Platform (`http://www.activiti.org`). Fig. 3 illustrated the overall architecture including the environments for modeling business processes (e. g., Activiti Designer) and for developing services and user interfaces (e. g., Eclipse). The architecture comprises in particular:

- *Interface extractor:* This components extracts the (initial) interfaces of service implementations and user interface implementations from the process models (stored in the process model repository). These interfaces are used for managing the mapping between the business process layer and the implementation layer. The actual mapping is stored in the Process to Service Mapping Database.
- *Rule extractor:* This component extracts the business-process level security and compliance specification from the process model repository and translates them to the implementation level. This includes a de-composition of high-level requirements into a set of technical requirements that can be checked on the implementation level.
- *Process to Service Mapping (Database):* This database stores the mapping between the business process layer and the implementation layer.
- *Source Code Analyzer:* The source code analyzer checks the service implementations as well as the user interface implementations based on the extracted rules. The results of the analysis are displayed to the developer in the service/user-interface development environment. If required, the results can be displayed to the process expert as well.

This system interacts, on the one hand, with one or more business process modeling environments (left hand side of Fig. 3) that stores the process model in a dedicated Process Model Repository. This component allows for modeling business processes and is also used for documenting the business-process level security and compliance requirements. On the other hand, the system interacts with one or more software or user interface development environments (right

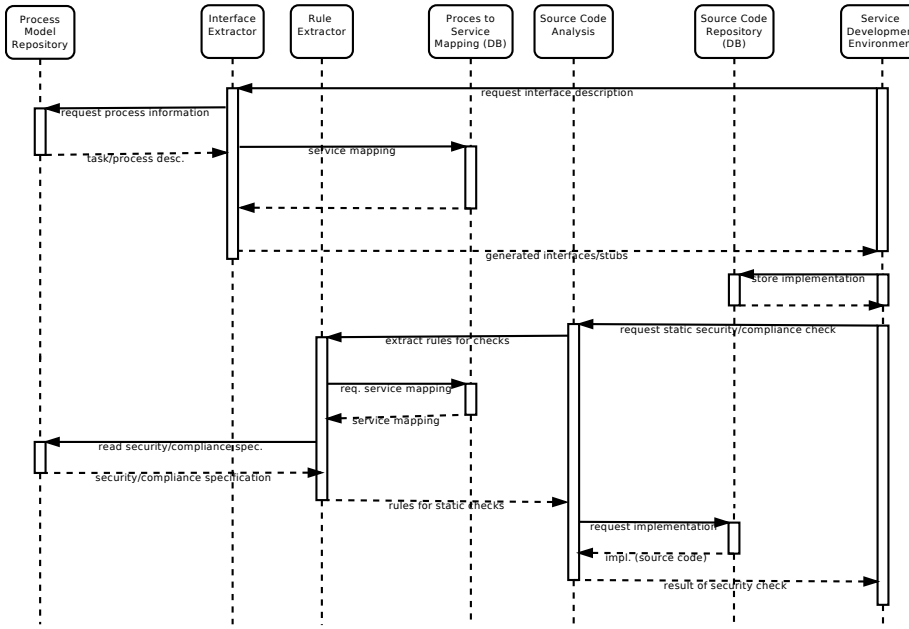

Fig. 4. Sequence diagram illustrating the use of our approach.

hand side of Fig. 3). Fig. 4 illustrates the interaction of the different components in more detail:

1. Generating the mapping from business processes to (service) implementations, i. e., the service developer initiates the creation of the service interfaces based on the process model.
   (a) The developer initiates, via the Service Development Environment, the interface generation using the Interface Extractor.
   (b) The Interface Extractor queries the Process Model Repository and generates the mapping from tasks and processes to interfaces (e. g., implemented in Java). This mapping is stored in the Process to Service Mapping database.
2. Service development:
   (a) Using the infrastructure provided by Service Development Environment, interfaces stubs are generated and stored in the source code repository.
   (b) The service developer creates or modifies the source of the services and stores the result in the source code repository.
3. Checking the correctness, security, and compliance on the implementation level:
   (a) From the Service Development Environment, the developer can check that her/his implementation fulfills the specified requirements. For this she/he uses the Source Code Analysis Module.
   (b) The Source Code Analysis module uses the Rule Extractor for generating a process specific (i. e., based on the information stored in the Process to

Service Mapping database as well as the Process Model Repository) setup for the analysis of the service implementations.

(c) The Source Code Analysis module executes a static source code analysis and reports the results (e. g., service fulfills the requirements, required property violated in module x on line y to the service developer.

Recall our example from Sect. 2: from the high-level specification of our example (see Fig. 2), we derive automatically requirements that need to be fulfilled on the source code level, i. e., by the programs implementing the user-interfaces for human tasks (e. g., Approve Budget) as well as for the service implementation that implement the automated tasks (e. g., Send Result). Here, for example, our method can ensure, at design-time, and, thus, without additional costs at run-time or during the audit, that

- the user interface for Approve Duration does not show/access the budget information,
- the user interface for Approve Budget does not show/access travel details (duration, destination, etc.),
- the necessary runtime-checks for enforcing the access control for all tasks are actually implemented,
- neither a service nor an user interface implementation accesses information in the back-end that is not needed for executing this process.

In particular, we detected the following two violations in our toy example:

- The user interface of task Request Travel access both the travel budget and the travel destination which violates the need to know principle.
- The service implementation contains a backdoor granting users with first name eve twice the amount of travel budget that was visible to the manager executing the task Approve Budget.

To review and fix the detected violation in the source code, we offer a direct jump to the line of the implementation where the violations occur.

Our prototype extends the Activiti BPM Platform and its modeling component, called Activiti Designer, which is based on Eclipse. For the static code analysis, we use our own analysis tool based on Wala (`http://wala.sf.net`). As an alternative, we are also experimenting with generating configurations for a commercially available static code analysis tools.

5 Related Work and Conclusion

5.1 Related Work

We see three areas of related work: modeling of security requirements for business processes, analyzing security properties of business processes, and runtime enforcement of security properties for business-process driven systems.

There is a large body of literature extending graphical modeling languages with means for specifying security or privacy requirements. One of the first approaches is SecureUML [18], which is conceptually very close to your BPMN

extension. SecureUML is a meta-model-based extension of UML that allows for specifying RBAC-requirements for UML class models and state charts. There are also various techniques for analyzing SecureUML models, e. g., [7, 11]. While based on the same motivation, UMLsec [16] is not defined using a meta-model. Instead, the security specifications are written, in an ad-hoc manner, in UML profiles. In contrast, integrating security properties in to business processes is a quite recent development, e. g., motivated by Wolter and Schaad [27]. In the same year, Rodríguez et al. [23] presented a meta-model based approach introduction a secure business process type supporting global security goals. In contrast, our approach allows the fine-grained specification of security requirements for single tasks or data objects. Similar to UMLsec, Mülle et al. [20] present an attribute-based approach (i. e., the conceptual equivalent of UML profiles) of specifying security constraints in BPMN models without actually extending BPMN.

With respect to the validation of security requirements on the business process level, the closed related work is the work of Wolter and Meinel [25] and Arsac et al. [5] both support the checking if an access control specification enforces binary static of duty and binding of duty constraints. Besides security properties, there is also the strong need for checking the consistency of business processes itself, e. g., the absence of deadlocks. There are several works that concentrate on these kind of process internal consistency validation, e. g., [3, 14]. Moreover, there are several approaches for analyzing access control constraints over UML models, e. g., [11, 16, 24]. These approaches are limited to simple access control modes, as it UML models are usually quite distant from business process descriptions comprising high level security and compliance goals.

Wolter et al. [26] present an approach for generating XACML policies for RBAC models in the context of process models as well as generating configurations for Apache Rampart (`http://axis.apache.org/axis2/java/rampart/`). Moreover, model-based security approaches for UML, e. g., [8, 13, 16], support, usually, the generation of security configurations as well.

5.2 Conclusion and Future Work

We presented an approach for checking process level security and conformance requirements on the implementation level. Our approach is integrated framework for modeling, analyzing, and enforcing security, privacy, and trust requirements in business process-driven systems. Checking process-level security and compliance requirements on the implementation level has several advantages. For example, all security requirements that can be guaranteed based on a static checked not need to be checked (or enforced) and runtime. As the conformance to those requirements is guaranteed "by design," these requirements are not subject to manual audits. Thus, our approach helps to improve the overall system performance as well as reduce costs for manual compliance audits.

There are several lines of related work, among them the development of support for system audits, e. g., by integrating analysis techniques such as [2, 4]. In particular, process mining approaches appear to be particularly interesting:

combing process mining with our business process animation, i. e., the visualization of attack traces, allows for interactively investigation deviations of the actual process execution with the intended one. Moreover, we see the integration analysis techniques checking the internal consistency of processes, e. g., [14], as well as their reconfiguration, e. g., [3]. To improve the run-time of the enforcement architecture, the generation of security artifacts can extended with support for advanced caching architectures, e. g., [17] or optimization techniques, e. g., [19]. Finally, we intend to integrate security testing approaches, e. g., [9], for validating the compliance of services and (legacy) back-end systems in a black-box scenario.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no. 257930. We thank Jan Alexander and Matthias Klink for valuable discussions on earlier versions of this paper.

References

[1] American National Standard for Information Technology – Role Based Access Control. ANSI, New York (2004). ANSI INCITS 359-2004

[2] van der Aalst, W., de Medeiros, A.: Process mining and security: Detecting anomalous process executions and checking process conformance. ENTCS **121**(0), 3–21 (2005). doi: 10.1016/j.entcs.2004.10.013

[3] van der Aalst, W.M.P., Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., Rosa, M.L., Mendling, J.: Correctness-preserving configuration of business process models. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE, *LNCS*, vol. 4961, pp. 46–61. Springer (2008). doi: 10.1007/978-3-540-78743-34

[4] Accorsi, R., Wonnemann, C.: Indico: Information flow analysis of business processes for confidentiality requirements. In: Cuéllar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM, *LNCS*, vol. 6710, pp. 194–209. Springer (2010). doi: 10.1007/978-3-642-22444-713

[5] Arsac, W., Compagna, L., Pellegrino, G., Ponta, S.E.: Security validation of business processes via model-checking. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS, *LNCS*, vol. 6542, pp. 29–42. Springer (2011). doi: 10.1007/978-3-642-19125-13

[6] Basel Committee on Banking Supervision: Basel III: A global regulatory framework for more resilient banks and banking systems. Tech. rep., Bank for International Settlements, Basel, Switzerland (2010). `http://www.bis.org/publ/bcbs189.pdf`

[7] Basin, D., Clavel, M., Doser, J., Egea, M.: Automated analysis of security-design models. Information and Software Technology **51**(5), 815–831 (2009). doi: 10.1016/j.infsof.2008.05.011. Special Issue on Model-Driven Development for Secure Information Systems

[8] Basin, D.A., Doser, J., Lodderstedt, T.: Model driven security: From UML models to access control infrastructures. ACM Transactions on Software Engineering and Methodology **15**(1), 39–91 (2006). doi: 10.1145/1125808.1125810

[9] Brucker, A.D., Brügger, L., Kearney, P., Wolff, B.: An approach to modular and testable security models of real-world health-care applications. In: ACM SACMAT, pp. 133–142. ACM Press (2011). doi: 10.1145/1998441.1998461.

[10] Brucker, A.D., Doser, J.: Metamodel-based UML notations for domain-specific languages. In: Favre, J.M., Gasevic, D., Lämmel, R., Winter, A. (eds.) 4th International Workshop on Software Language Engineering (ATEM 2007) (2007).
[11] Brucker, A.D., Doser, J., Wolff, B.: A model transformation semantics and analysis methodology for SecureUML. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006: Model Driven Engineering Languages and Systems, no. 4199 in LNCS, pp. 306–320. Springer (2006). doi: 10.1007/1188024022. An extended version of this paper is available as ETH Technical Report, no. 524.
[12] Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: SecureBPMN: Modeling and enforcing access control requirements in business processes. In: ACM SACMAT. ACM Press (2012). doi: 10.1145/2295136.2295160
[13] Brucker, A.D., Petritsch, H.: Extending access control models with break-glass. In: Carminati, B., Joshi, J. (eds.) ACM SACMAT, pp. 197–206. ACM Press (2009). doi: 10.1145/1542207.1542239.
[14] Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology **50**(12), 1281–1294 (2008). doi: 10.1016/j.infsof.2008.02.006
[15] HIPAA: Health Insurance Portability and Accountability Act of 1996. `http://www.cms.hhs.gov/HIPAAGenInfo/` (1996)
[16] Jürjens, J., Rumm, R.: Model-based security analysis of the german health card architecture. Methods Inf Med **47**(5), 409–416 (2008)
[17] Kohler, M., Brucker, A.D., Schaad, A.: ProActive Caching: Generating caching heuristics for business process environments. In: International Conference on Computational Science and Engineering (CSE), vol. 3, pp. 207–304. IEEE Computer Society (2009). doi: 10.1109/CSE.2009.177.
[18] Lodderstedt, T., Basin, D.A., Doser, J.: SecureUML: a uml-based modeling language for model-driven security. In: Jézéquel, J.M., Hussmann, H., Cook, S. (eds.) UML, no. 2460 in LNCS, pp. 426–441. Springer (2002)
[19] Miseldine, P.: Automated XACML policy reconfiguration for evaluation optimisation. In: Win, B.D., Lee, S.W., Monga, M. (eds.) SESS, pp. 1–8. ACM (2008). doi: 10.1145/1370905.1370906
[20] Mülle, J., von Stackelberg, S., Böhm, K.: A security language for BPMN process models. Tech. rep., University Karlsruhe (KIT) (2011).
[21] OASIS: eXtensible Access Control Markup Language (XACML), version 2.0 (2005). `http://docs.oasis-open.org/xacml/2.0/XACML-2.0-OS-NORMATIVE.zip`
[22] Object Management Group: Business process model and notation (BPMN), version 2.0 (2011). Available as OMG document formal/2011-01-03
[23] Rodríguez, A., Fernández-Medina, E., Piattini, M.: A BPMN extension for the modeling of security requirements in business processes. IEICE - Trans. Inf. Syst. **E90-D**, 745–752 (2007). doi: 10.1093/ietisy/e90-d.4.745
[24] Sohr, K., Ahn, G.J., Gogolla, M., Migge, L.: Specification and validation of authorisation constraints using UML and OCL. In: di Vimercati, S.D.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS, *LNCS*, vol. 3679, pp. 64–79. Springer (2005)
[25] Wolter, C., Meinel, C.: An approach to capture authorisation requirements in business processes. Requir. Eng. **15**(4), 359–373 (2010). doi: 10.1007/s00766-010-0103-y
[26] Wolter, C., Menzel, M., Schaad, A., Miseldine, P., Meinel, C.: Model-driven business process security requirement specification. Journal of Systems Architecture **55**(4), 211–223 (2009). doi: 10.1016/j.sysarc.2008.10.002. Secure Service-Oriented Architectures (Special Issue on Secure SOA)

[27] Wolter, C., Schaad, A.: Modeling of task-based authorization constraints in bpmn. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM, *LNCS*, vol. 4714, pp. 64–79. Springer (2007)

[28] Wolter, C., Schaad, A., Meinel, C.: Deriving XACML policies from business process models. In: Weske, M., Hacid, M.S., Godart, C. (eds.) WISE Workshops, *LNCS*, vol. 4832, pp. 142–153. Springer (2007)