Compliance Validation of Secure Service Compositions

Achim D. Brucker[1], Luca Compagna[2], and Pierre Guilleminot[2]

[1] SAP SE, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
achim.brucker@sap.com
[2] SAP SE, Sophia-Antipolis, Mougins, France
luca.compagna@sap.com

Abstract. The Aniketos Secure Composition Framework supports the specification of secure and trustworthy composition plans in term of BPMN. The diversity of security and trust properties that is supported by the Aniketos framework allows, on the one hand, for expressing a large number of security and compliance requirements. On the other hand, the resulting expressiveness results in the risk that high-level compliance requirements (e. g., separation of duty) are not implemented by low-level security means (e. g., role-based access control configurations).
In this chapter, we present the Composition Security Validation Module (CSVM). The CSVM provides a service for checking the compliance of secure and trustworthy composition plans to the service designer. As proof-of-concept we created a prototype in which the CSVM module is deployed on the SAP NetWeaver Cloud and two CSVM Connectors are built supporting two well-known BPMN tools: SAP NetWeaver BPM and Activiti Designer.
Keywords: Validation, Security, BPMN, SecureBPMN, Compliance.

1 Introduction

The Aniketos Secure Composition Framework (see Chapter 4 and Chapter 9) supports the specification of secure and trustworthy composition plans in term of BPMN (see Chapter 8). The diversity of security and trust properties that is supported by the Aniketos Secure Composition Framework allows, on the one hand, for expressing a large number of security and compliance requirements. On the other hand, the resulting expressiveness results in the risk that high-level compliance requirements (e. g., separation of duty) are not implemented by low-level security controls (e. g., role-based access control configurations).

To ensure the compliance of service composition to the specified security requirements, we are integrating a model-checking based validation approach (see [3, 4, 8] for details) into the Aniketos platform. This integration into the Aniketos Secure Composition Framework [7] consists out of a server component (the Composition Security Validation Module) and an integration into the Activiti Designer which is also used as front-end the Aniketos Secure Composition Framework. Moreover, we provide an alternative integration into SAP

A.D. Brucker et al. (Eds.): Secure Service Composition, LNCS 8900, pp. 136–149, 2014.

NetWeaver BPM. Both modelling tools provide a provides accessible user interfaces that support the modelling of security requirements as well as the graphical rendering of the validation results.

2 The Composition Security Validation Module (CSVM)

Figure 1 provides a high-level overview of the architecture of the Composition Security Validation Module (CSVM). The overall architecture comprises two main elements: the CSVM itself and the CSVM Connector. The service designer uses a CSVM-enabled BPM client to validate the compliance of her business processes. The CSVM-enabled BPM client is a BPM client for which a CSVM connector has been developed and integrated.

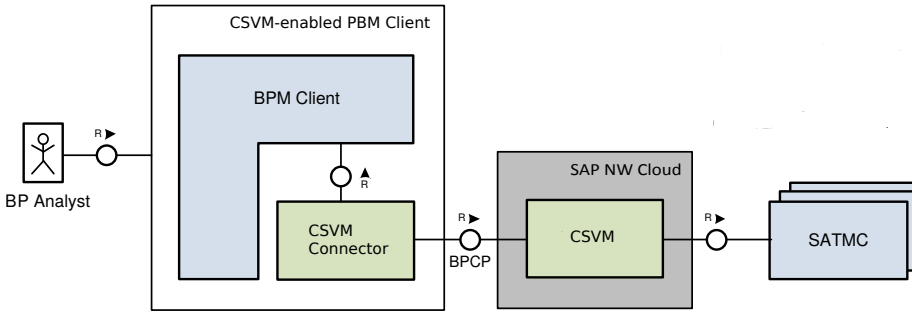

Fig. 1: High-level View of the CSVM Architecture

The security validation activity is triggered by the service designer. The CSVM connector retrieves all the security-relevant information necessary for the validation and creates an instance of the Business Process Compliance Problem (BPCP). The BPCP is send to the CSVM for validation. The BPCP is an XML specification that we devised to make our approach as much as possible independent from the targeted BPM client. It relies on the established BPMN2 standard [11] and extends it with a BPMN2-SEC schema that we defined to capture the security-relevant aspects of business processes.

The validation itself is handled by the CSVM that transforms the BPCP resource into a formal specification suitable for SATMC [2], a SAT-based model checker. As soon as the model checker completes its formal analysis the raw result is provided back to the CSVM that converts it into an XML format. The CSVM connector can now access and render the validation result. Alternatively the results can be consulted on the cloud, e. g., using a Web application.

2.1 Business Process Compliance Problem (BPCP)

The Business Process Compliance Problem (BPCP) is a client-independent data format that bundles the property that should be validated together with all

information necessary for its validation. As such, the CSVM only needs the BPCP to validate the compliance of a given business process. In more detail, the BPCP is an XML specification that contains two elements:

- the composition plan (business process model) in standard BPMN2 format, optionally augmented details on Data Objects and their task input/output,
- the security-relevant aspects of the business process and corresponding validation results both specified in BPMN2-SEC.

In this section, we will base our examples on a simple business process for requesting travel approvals (see Figure 2). The security relevant aspects of the

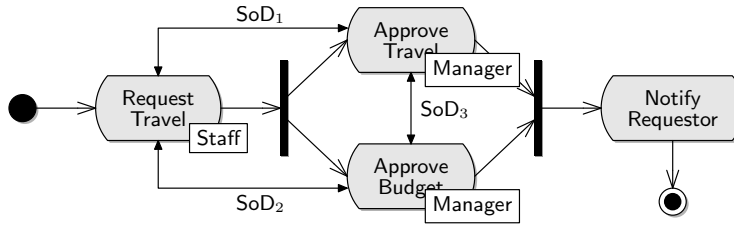

Fig. 2: A simple travel approval process with annotated security requirements

BPCP are described in BPMN2-SEC to ensure independent form a specific BPM client. For example both SecureBPMN (see Chapter 8), as used by the Aniketos platform, as well as the proprietary format used by SAP Netweaver BPM can easily be mapped to BPMN2-SEC.

The BPMN2-SEC language (see Figure 3) allows mainly to express three aspects: 1. the *policy* underlying the targeted business process, 2. the *security properties* the business process is supposed to satisfy, and 3. the validation *result* (if any already obtained).

The Policy. The `Policy` element comprises both the role-based access control (RBAC) [14] relevant for the business process and the delegation policy the BPM client is subject to. The RBAC element allows for specifying the roles and users involved in the business process, the permissions, and the assignment of these permissions to users and roles. Listing 1.1 shows a simplified example of an RBAC section within a BPCP specification.

- `manager`, `staff`, and `reception` are roles (lines 2–7)
- `mickael` is a user (lines 8-11)
- the role `manager` is assigned to the user `mickael` (line 12)
- Two permissions are defined: one allows for executing the Approve Travel (`approvetravel`) activity (lines 15-20) and the other one prohibits, via the `negate` construct, the execution of the Request Travel (`requesttravel`) activity (lines 21-25)
- The permission to execute the Approve Travel activity is assigned to role `manager` (lines 29-30) while the prohibition is assigned to role `reception` (lines 31-32)

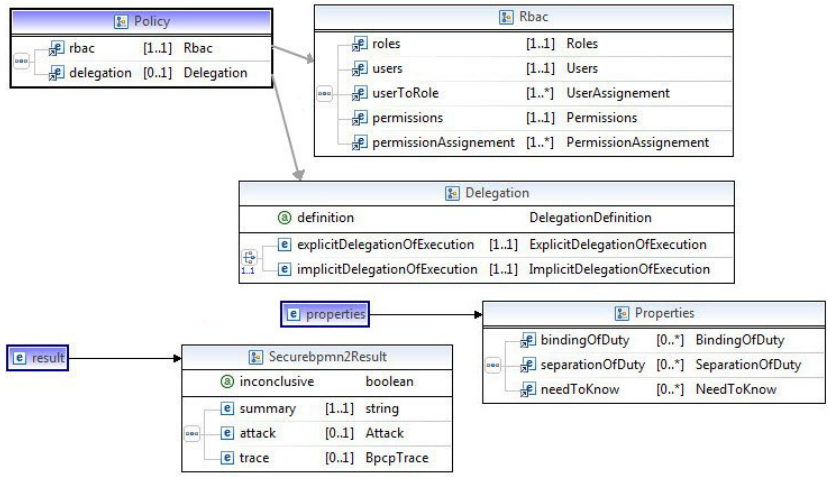

Fig. 3: The meta-model of BPMN2-SEC

```
<rbac>
   <roles>
       <role id="manager"><name>Manager</name></role>
       <role id="staff"><name>Staff</name></role>
       <role id="reception"><name>Reception</name></role>
       ...
   </roles>
   <users>
       <user id="mickael"><name>Mickael</name></user>
       ...
   </users>
   <userToRole roleRef="manager" userRef="mickael" />
   ...
   <permissions>
       <permission id="exe_approveTravel">
           <action>execute</action>
           <resource>bpmn2:main#approvetravel</resource>
       </permission>
       <permission id="noexe_requestTravel" negate="true">
           <action>execute</action>
           <resource>bpmn2:main#requesttravel</resource>
       </permission>
       ...
   </permissions>
   <permissionAssignement principalRef="manager"
       permissionRef="exe_approveTravel" />
   <permissionAssignement principalRef="reception"
       permissionRef="noexe_requestTravel"/>
   ...
</rbac>
```

Listing 1.1: BPMN2-SEC: RBAC example (simplified)

Moreover, BPMN2-SEC supports the specification of delegation (see [8] for details): the `delegation` element allows for specifying the intended delegation policy employed by the BPM client during the execution of the service composition. Basically the delegation policy defines under which conditions (if any) a

user involved in a certain task of the business process can delegate to a colleague such a task.

Security Properties. The `Properties` element lists the security properties that the business process is required to achieve. These are the properties that our security validation approach will evaluate. Properties can be created on top of an enumeration of security property templates. Our approach can be easily extended to support other property templates provided they can be recast as an LTL (Linear Temporal Logic) formula which is a quite powerful and expressive logic. The properties currently defined and supported by CSVM are

Data confidentiality: The access to sensitive data should be restricted to certain users.

Separation of duty (SoD): Separation of duty aims to mitigate the risk of fraud by dividing the responsibility in executing critical parts of business processes.

Binding of Duty: In some cases, it is necessary for a group of business process activities to be performed by only one user so as to ensure the integrity of the data.

Need-to-know (NtK): Users shall only be able to access only the information that is strictly necessary to accomplish their tasks, i. e., the tasks should be performed in an objective manner. For a critical task, data can be defined that should not be known by the principal executing the task.

Access Control Over Automated Tasks: Automated tasks are usually implemented by calling a service. Such services are often provided by external organisations should adhere to the access restrictions required by the service providers.

Listing 1.2 presents two simple example properties, namely SoD and Ntk, for a travel approval process: the first one captures a SoD between travel request and travel approval (lines 2-6) and the second one model a NtK stating that the manager that will execute the travel budget approval does not need to know the trip business reason (lines 7-11).

```
<properties>
   <separationOfDuty id="sod1" maxUserActions="1" minUsers="2">
       <activityRef>bpmn2:main#requesttravel</activityRef>
       <activityRef>bpmn2:main#approvetravel</activityRef>
   </separationOfDuty>
   <needToKnow id="needtoknow1">
       <activityRef>bpmn2:main#approvetravel</activityRef>
       <dataObjectRef>bpmn2:main#traveldata</dataObjectRef>
       <privatefield>reason</privatefield>
   </needToKnow>
   ...
</properties>
```

Listing 1.2: BPMN2-SEC: property example

Results. The `Result` element describes the validation result. Listing 1.3 shows an example validation result of our simple travel approval process. The validation result is not inconclusive (line 1) meaning that the model checker was able to determine whether there is an attack (i. e., a possible system trace that results in a system state violating the compliance or security requirements) or not (this is normally the case when the business process does not feature complex loops).

```
<securebpmn2:result inconclusive="false">
   <securebpmn2:summary>
       Separation of Duty between Request Travel and
       Approve Travel
   </securebpmn2:summary>
   <securebpmn2:attacks>
     <securebpmn2:attack name="Separation Of Duty"
             propertyRef="securebpmn2:main#sod1" type="SoD">
       <securebpmn2:par>karl</securebpmn2:par>
       <securebpmn2:par>requesttravel</securebpmn2:par>
       <securebpmn2:par>approvetravel</securebpmn2:par>
     </securebpmn2:attack>
  </securebpmn2:attacks>
  <securebpmn2:trace>
     <securebpmn2:step
         flowElementRef="bpmn2:main#requesttravel"
         name="Request Travel">
       <securebpmn2:subStep type="claimed">
           <securebpmn2:par>staff</securebpmn2:par>
           <securebpmn2:par>karl</securebpmn2:par>
           <securebpmn2:par>requesttravel</securebpmn2:par>
       </securebpmn2:subStep>
       <securebpmn2:subStep type="executed">
           <securebpmn2:par>staff</securebpmn2:par>
           <securebpmn2:par>karl</securebpmn2:par>
           <securebpmn2:par>requesttravel</securebpmn2:par>
       </securebpmn2:subStep>
     </securebpmn2:step>
     ...
     <securebpmn2:step flowElementRef="bpmn2:main#approvetravel"
          name="Approve Travel">
       <securebpmn2:subStep type="delegationOfpermission">
           <securebpmn2:par>mickael</securebpmn2:par>
           <securebpmn2:par>manager</securebpmn2:par>
                   <securebpmn2:par>karl</securebpmn2:par>
           <securebpmn2:par>approvetravel</securebpmn2:par>
       </securebpmn2:subStep>
     </securebpmn2:step>
     ...
     <securebpmn2:step flowElementRef="bpmn2:main#approvetravel"
                  name="Approve Travel">
       <securebpmn2:subStep type="claimed">
           <securebpmn2:par>manager</securebpmn2:par>
           <securebpmn2:par>karl</securebpmn2:par>
           <securebpmn2:par>approvetravel</securebpmn2:par>
       </securebpmn2:subStep>
       <securebpmn2:subStep type="executed">
           <securebpmn2:par>manager</securebpmn2:par>
           <securebpmn2:par>karl</securebpmn2:par>
           <securebpmn2:par>approvetravel</securebpmn2:par>
       </securebpmn2:subStep>
     </securebpmn2:step>
   </securebpmn2:trace>
</securebpmn2:result>
```

Listing 1.3: BPMN2-SEC: Validation results

More specifically, an attack has been found on one of the SoD properties (see lines 6-13). The counter-example trace is also reported (lines 14-55). In there, Karl claims and executes a Travel Request for himself (lines 15-28). Sometime in the future Karl got delegated by the manager Mickael to handle Mickael' managerial activities (delegation of permission, lines 30-39). We could imagine that Mickael got suddenly sick. Karl has now all the permissions associated with the manager role and, among other things, can claim and execute the approval of his own travel request (lines 41-54) violating the SoD requirement.

2.2 The CSVM Architecture

Figure 4 provides a detailed overview of the CSVM architecture. The CSVM Connector includes a loader component to load from the BPM client all data necessary to create the BPCP resource. It is often the case that not all the data that is necessary for a complete definition of a BPCP can be loaded from the BPM client (e. g., the security properties to be validated). The UI component provides graphical controls to collect the missing data, to configure the CSVM connector, to trigger the security validation process overall, to render the validation results, etc. The REST client takes care of preparing and sending the REST requests to the REST API of the CSVM. The controller component coordinates the interaction among all the components of the CSVM connector. The persistency component can be optionally implemented to enrich the CSVM connector in keeping track of all the validations that were carried out by business analysts within this specific CSVM connector.

The CSVM Server exposes a REST API whose incoming requests are handled by the Request handler component. BPCPs are REST resources that are stored with their validation status into the persistency layer (Persistency manager component). The BPC broker queues the pending BPCPs that are then pulled by BPC workers in order to be validated. The BPC worker first translates the BPCP into its formal representation in ASLan (see Chapter 9) that is fed in input to one external SATMC instance. The model checking task can be quite costly in terms of time and resource consumption. For the sake of performances one SATMC process should run on one virtual machine with 100% CPU and reasonable RAM allocation. The BPC workers manager component starts on-the-fly a new BPC worker thread and a corresponding external virtual machine with a new SATMC instance as soon as certain work-load customer-dependent criteria are reached. As soon as the model checker finishes the analysis, the BPC worker translates this outcome into the proper XML structure that is filled into the result element of the BPCP. The validation result is now ready to be consulted. The CSVM Portal provide a single web-based entry-point for the end-users that could for instance monitor the status of all their BPCP resources. The CSVM Portal also offers a full-fledged security validation environment available for those BPM Clients that wants to opt for a light integration with CSVM. Indeed, even customers employing BPM Clients that are not augmented with CSVM Connectors could get advantage of CSVM by just accessing the CSVM Portal and managing the entire security validation life-cycle there. Of course the

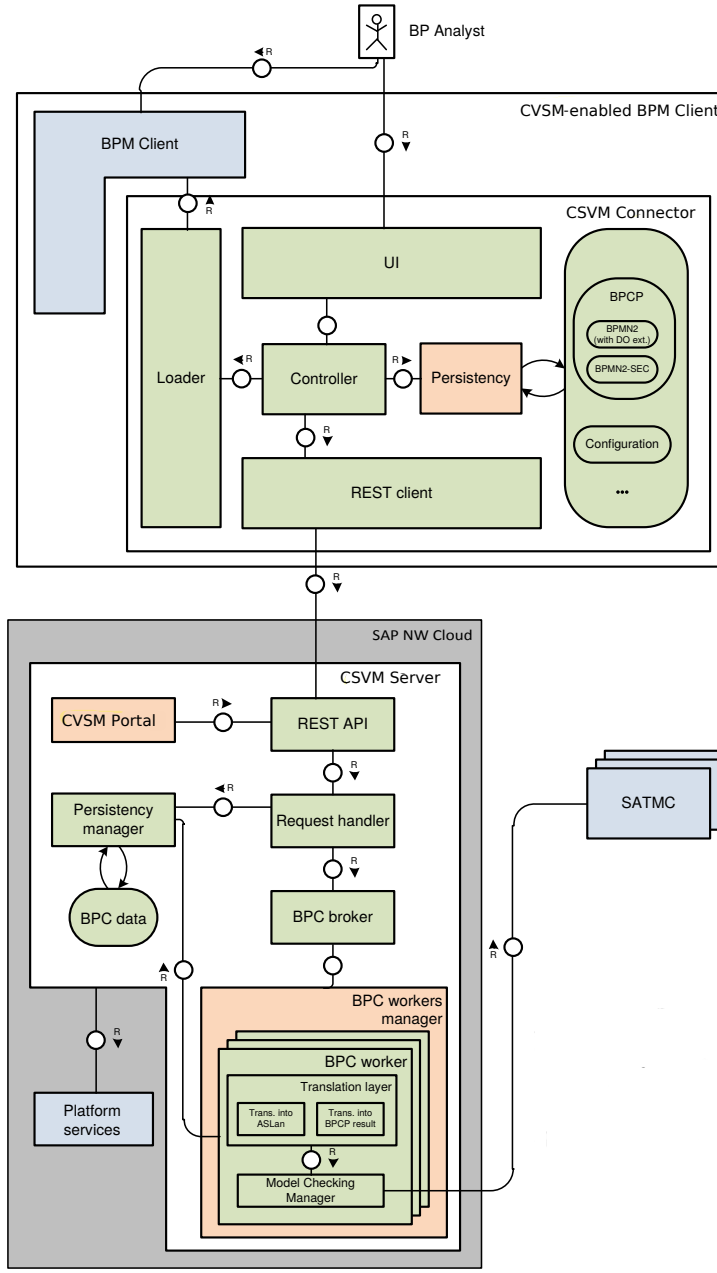

Fig. 4: The CSVM Architecture

level of interactivity and usability would definitely be not comparable to those BPM Clients featuring customized CSVM Connectors. This is why we consider more promising those business scenarios in which BPM Clients are enriched with CSVM Connectors.

2.3 The REST-Based Interaction Protocol

As mentioned the CSVM Connectors and the CSVM Server interact through a REST API that features methods for managing BPCPs and in particular their creation and reading of the validation results. In order to instantiate a new BPCP, the CSVM Connector exports a set of information from its BPM Client. This set of information will be necessary to create a BPCP meta-model. To allow BPM Client to export different files in parallel, the CSVM REST API is defined as a multiple-step resource creation. First, a new resource associated to the validation is created. This resource is unique and will remain accessible to the end-user at any time at a specific location. The only action required is to send a POST at the CSVM Server URI `/validation/`. After this, the client can export the set of information required to feed the newly created resource. To do so, the client sends PUT requests associated with data, on specific nested elements of its validation resource location.

After the creation of the validation resource, the client can start the validation process by asking for the result of a specific BPCP resource with a GET. As mentioned the security validation process may take some time and this is why it is treated asynchronously. Therefore the client may not get the result immediately.

3 Lessons Learned

In [3] and [5] we presented validation approaches for secure business processes that integrate the validation into the BPM Client. Our discussion with the product groups within SAP revealed that this approach has, in particular in an industrial environment, certain limitations ranging from technical issues like scalability to licensing and maintenance issues. For instance, some customers use both on-demand and on-premise BPM Clients while designing their business processes.

While both the on-demand and the on-premise BPM Clients could have been augmented with an implementation of the original security validation approach, the required effort for this was a clear obstacle. Additional commercialisation obstacles were also perceived on the BPM Client software producer side: while the Security Validation approach provides a nice-to-have differentiating feature, the long-term maintenance contracts for enterprise software does not go very well with the idea of a research proof-of-concept depending on (academic) third-party modules. All in all, the following requirements motivated us to switch for a cloud-based solution:

- CSVM shall be flexible enough to match the heterogeneous BPM customer landscapes including those in which multiple instances of different (on-demand or on-premise) BPM Clients are operated by multiple business analysts;
- CSVM shall be scalable with respect to multiple customer landscapes;

- CSVM shall be flexible enough to offer various degrees of integration within BPM Clients ranging from the most customisable one up to the lightest/simplest one:
  a) the BPM Client is augmented with its own customised CSVM UIs for e. g., specifying the security requirements of the business process under-design, rendering the results of the validation, etc;
  a) the BPM Client is just augmented with a button that outsources the overall security validation activity on the Cloud including e. g., security requirement specification, result rendering, etc;
- CSVM shall be expressive and flexible enough to be consumable by most of the commercial, state-of-the-art BPM Clients despite of their peculiarities and differences;
- CSVM shall be extensible enough to easily integrate new security properties within the validation life-cycle;
- CSVM shall be extensible enough to integrate novel, efficient techniques for validating BPCP; e. g., it shall be possible to add a novel model checker or different automated reasoning tool;

In our CSVM solution we decouple the security validation business logic from the rest of the approach and we take advantage of the Cloud paradigm as a vehicle to overcome some of the challenges that the original security validation approach faced, especially with respect to commercialization. To assess and

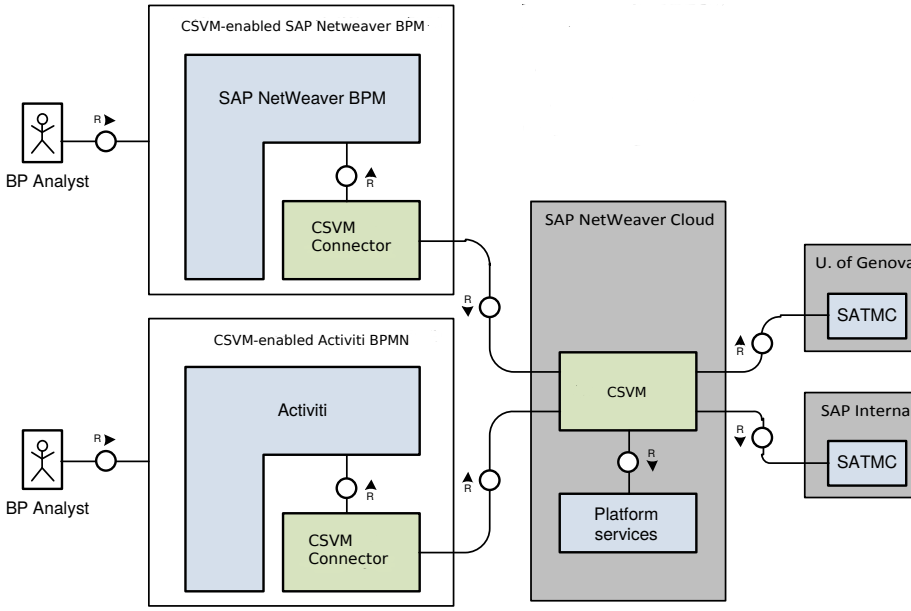

Fig. 5: Proof-of-Concept

demonstrate our overall CSVM approach we focused on the proof-of-concept shown in Figure 5 in which the CSVM Server is deployed on the SAP NetWeaver

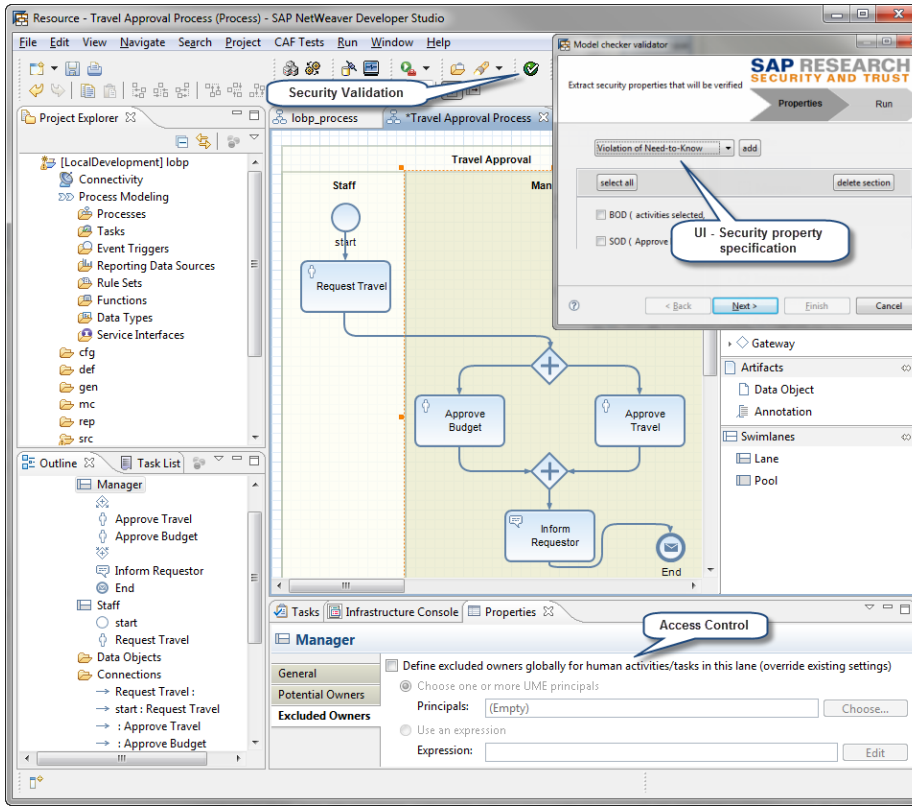

Fig. 6: Security Validation within SAP NetWeaver BPM

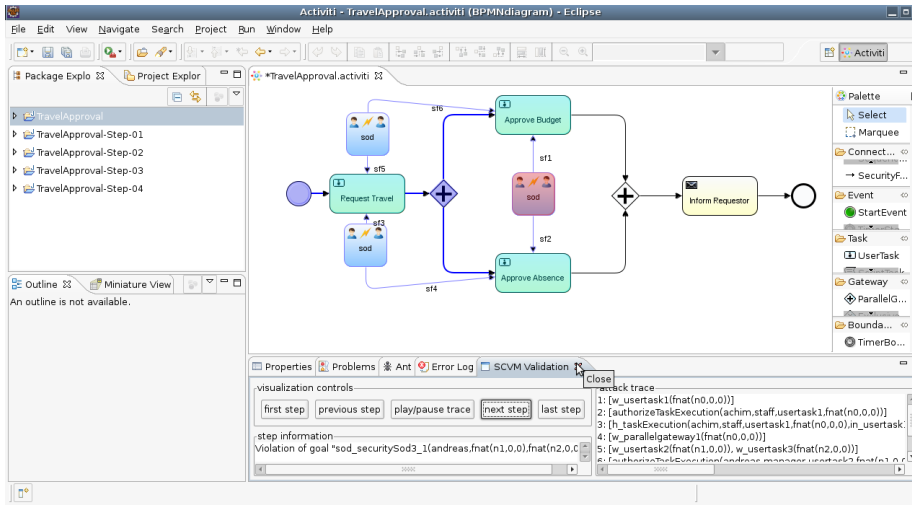

Fig. 7: Security Validation within the Activiti BPMN Editor

Cloud and its REST API is consumed by two BPM clients, SAP NetWeaver BPM (see Figure 6) and Activiti (see Figure 7). Both these BPM Clients use an Eclipse-based business process design environment. This is why we first developed a generic CSVM Connector for the Eclipse environment and then we customised it for our BPM clients.

The performance of the security validation activity, below 1 second, improved with respect to the experiments run and described in [3]. This is simply due to more powerful machines hosting the SATMC model checker. More interestingly, the CSVM architecture allows to efficiently handle parallel requests for security validation. In our proof-of-concept we only considered two machines hosting SATMC and still we were able to smoothly serve two business analysts designing medium-size business processes (around 50 tasks, 5 users and roles involved, and 5 data objects) and requesting for a security validation every 15 minutes. These are promising preliminary results that we aim to extend by setting up pilots with customers so to run more intensive experiments in real landscapes.

4 Conclusion and Related Work

4.1 Related work

While there is a large body of literature extending business process modelling languages with means for expressing security and regulatory compliance properties, e. g., [6, 10, 12, 13, 16] only a few approaches support validation or testing of the specified properties. The closest related works are [15] and [13]. Wolter and Meinel [15] use SPIN for checking that if an access control specification enforces binary static separation of duty and binding of duty constraints. Salnitri et al. [13] use specialised algorithms implemented in a query engine for validating generic compliance requirements. Additionally, [5] presents an approach that allows to statically check that service implementations, e. g., in Java, conform to the process-level security and regulatory compliance specification.

Besides security properties, there is also strong need for checking the consistency of business processes itself, e. g., the absence of deadlocks. There are several works, e. g., [1, 9] that integrate these kind of process internal consistency validation checks locally into the business process modelling environment.

4.2 Conclusion and Further Work

In this paper we presented CSVM, a promising approach and research prototype to test business process compliance. CSVM takes advantage of the Cloud paradigm to provide on-demand security validation services to BPM systems. Once properly interfaced via a CSVM Connector, the BPM Client is enabled to consume CSVM services, allowing its business analysts to determine, in a push-button fashion, whether the business processes under-design are respectful of critical compliance and security properties. Moreover, the CSVM architecture meets core business requirements collected during internal projects run at SAP

with the ultimate goal of increasing its chances to reach industrial commercialisation. We developed and deployed a proof-of-concept on top of our CSVM approach and demonstrated through preliminary results that it can serve multiple business analysts using heterogeneous BPM Clients even belonging to the same customer landscape.

Potential further steps include piloting with real customers, more intensive testing and assessment of CSVM scalability (e. g., using the elastic Amazon Cloud as hosting platform for SATMC instances to benchmark the BPC worker manager component), and integration of the implementation validation discussed in [5]. Last, but not least, we would like to explore if the availability of a common security validation technique like CSVM could pave the way for 1. establishment of domain-specific repositories of compliance requirements accessible for any BPM system, and 2. a systematic certification of business processes underdesign that could be then compared in this regards and, e. g., sold at different prices depending also on the security and compliance they offer.

References

[1] 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, *Lecture Notes in Computer Science*, vol. 4961, pp. 46–61. Springer-Verlag, Heidelberg (2008). doi: 10.1007/978-3-540-78743-3_4

[2] Armando, A., Carbone, R., Compagna, L.: LTL Model Checking for Security Protocols. Journal of Applied Non-Classical Logics **19**(4), 403–429 (2009)

[3] Arsac, W., Compagna, L., Kaluvuri, S.P., Ponta, S.E.: Security validation tool for business processes. In: Breu, R., Crampton, J., Lobo, J. (eds.) SACMAT, pp. 143–144. ACM (2011)

[4] Arsac, W., Compagna, L., Pellegrino, G., Ponta, S.E.: Security Validation of Business Processes via Model-checking. In: International Symposium on Engineering Secure Software and Systems (ESSoS 2011). Lecture Notes in Computer Science, Springer-Verlag (2011)

[5] Brucker, A.D., Hang, I.: Secure and compliant implementation of business process-driven systems. In: Joint Workshop on Security in Business Processes (SBP), *Lecture Notes in Business Information Processing (LNBIP)*, vol. 132. Springer-Verlag (2012) doi: 10.1007/978-3-642-36285-9_66

[6] Brucker, A.D., Hang, I., Lückemeyer, G., Ruparel, R.: SecureBPMN: Modeling and enforcing access control requirements in business processes. In: ACM symposium on access control models and technologies (SACMAT), pp. 123–126. ACM Press (2012). doi: 10.1145/2295136.2295160

[7] Brucker, A.D., Malmignati, F., Merabti, M., Shi, Q., Zhou, B.: A framework for secure service composition. In: International Conference on Information Privacy, Security, Risk and Trust (PASSAT), pp. 647–652. IEEE Computer Society (2013). doi: 10.1109/SocialCom.2013.97.

[8] Compagna, L., Guilleminot, P., Brucker, A.D.: Business process compliance via security validation as a service. In: Oriol, M., Penix, J. (eds.) IEEE Sixth International Conference on Software Testing, Verification and Validation (ICST), pp. 455–462. IEEE Computer Society (2013). doi: 978-1-4673-5961-0

[9] 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

[10] Mülle, J., von Stackelberg, S., Böhm, K.: A security language for BPMN process models. Tech. rep., University Karlsruhe (KIT) (2011)

[11] OMG: Business Process Modeling Notation (BPMN). http://www.omg.org/spec/BPMN/2.0 (2011)

[12] 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

[13] Salnitri, M., Dalpiaz, F., Giorgini, P.: Modeling and verifying security policies in business processes. In: Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P. (eds.) BMMDS/EMMSAD, *Lecture Notes in Business Information Processing*, vol. 175, pp. 200–214. Springer (2014). doi: 10.1007/978-3-662-43745-2_14

[14] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer **29**(2), 38–47 (1996).

[15] 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

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