Business Process Compliance via Security Validation as a Service

Luca Compagna, Pierre Guilleminot
SAP Research Sophia-Antipolis, Mougins, France
luca.compagna@sap.com

Achim D. Brucker
SAP AG, Karlsruhe, Germany
achim.brucker@sap.com

***Abstract*—Modern enterprise systems are often process-based, i. e., they allow for the direct execution of business processes that are specified in a high-level language such as BPMN. In this paper, we present a service, called Security Validation as a Service (SVaaS) for validating the compliance of the business processes during design-time. Basically, while modeling a business process the business analyst specifies as well the security and compliance requirements the business process should comply to. By pressing a button, these requirements are validated and the results are presented in a graphical format to the business analysis.**

At the core of SVaaS lies a rigorous and industrially viable approach in which the security validation business logic is handled server-side (SVaaS Server) in the Cloud, while the client-side user interface that business analysts use is handled by a light-weight SVaaS Connector. As proof-of-concept we created a SVaaS prototype in which the SVaaS Server is deployed on the SAP NetWeaver Cloud and two SVaaS Connectors are built to enable two well-known BPMN tools, SAP NetWeaver BPM and Activiti, to consume SVaaS against industrial relevant business processes.

***Keywords*-Validation, Security, Business Process Management**

I. Introduction

More and more organizational activities are captured and executed via business processes. While this increases business agility, flexibility and efficiency, security and regulatory compliance requirements, including fraud prevention require the business processes to be designed and executed with care.Industrial Business Process Management systems (BPM Clients) aim to enforce security and compliance, but are of little help in guaranteeing business analysts that the business process they have designed fulfills the expected security requirements.

Figure 1 illustrates a simple example of a process for approving travel requests: a staff member may issue a travel request. Both the travel reason as well as the travel budget needs to be approved by a manager. Afterwards, the requesting user is notified if her request is granted or not. The execution of a task may be user-centric or automatic. In the latter case, the system executes a service that implements the task without user interaction; in the former case, a human is responsible for executing the task, i. e., the user has to claim the task to work it off. Regulatory compliance and company policy requirements apply also for this simple business process. Besides the role-based access control that ensures that only managers approve the travel reason and budget, we use *separation of duty* (SoD) constraints to ensure that the persons requesting a travel as well as approving the reason or budget are mutually exclusive. Already in this simple example, we need to validate that the access control specification and the SoD constraints do not contradict each other to ensure that the process is actually executable respectful of its regulatory compliance and security requirements. We call this validation problem, the *Business Process Compliance Problem* (BPCP).

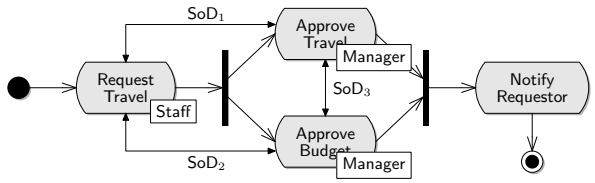

Figure 1: A simple travel approval process with annotated security requirements

Besides using standard testing techniques towards both the BPM Client platform and the software services employed by the business process to probe they are behaving as they should be, the BPCP needs to check that the business process execution within the BPM Client does not violate any of the expected security requirements. This testing activity can be recast into a validation problem that is performed at design time by considering information of the BPM Client runtime environment (e. g., users, roles, delegation policy). This recast finds its reasons in one of the key model-driven development principle underlying the BPM paradigm: you run exactly the business process that you design. Security Validation of business processes, based on a combination of model-checking, accessible user interfaces and graphical rendering of the outcomes, has been proven a successful and usable technique to detect business process compliance issues at design time [1], [2].

In this paper, we describe our business process compliance validation platform SVaaS (Security Validation as a Service) that follows up on this rigorous approach. The ultimate goal of SVaaS is to provide a scalable and extensible validation solution for the large BPM community as a whole. Moreover, providing formal analysis approaches as services clears barriers in commercializing formal methods.

The paper is organized as follows. In Section II we detail the overall SVaaS concepts (e.g., the BPCP), architecture, and operations. Section III presents what we learned in promoting security validation of business process within the SAP industrial environment including the assessment of the proof-of-concept we deployed there. Finally, in Section IV we discuss the related work and we provide some final remarks as well as future promising directions.

II. SVaaS

Figure 2 depicts a high-level overview of the SVaaS architecture. SVaaS comprises two main elements: the SVaaS Server and the SVaaS Connector. The business analyst uses a SVaaS-enabled BPM Client to validate the compliance of his/her business processes. The SVaaS-enabled BPM Client is just a BPM Client for which a SVaaS Connector has been developed and integrated. The security validation activity

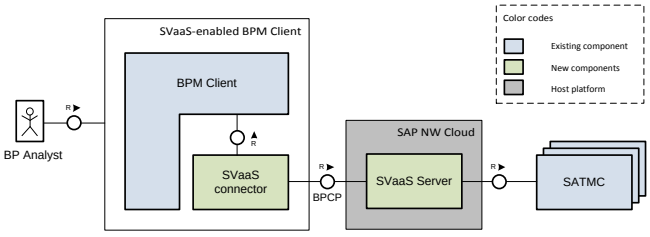

Figure 2: SVaaS architecture - high level view

is triggered by the business analyst. The SVaaS Connector retrieves all the security-relevant information necessary for the validation, wraps them in a Business Process Compliance Problem (BPCP), and initiates the validation by invoking the SVaaS Server. 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 [3] and extends it with a BPMN2-SEC schema defined by us to capture the security-relevant aspect of business processes. The BPCP is handled as a REST resource. The validation itself is handled by the SVaaS Server that transforms the BPCP resource into a formal specification suitable for formal analysis via the SAT-based model checker (SATMC, [4]). As soon as the model checker completes its formal analysis the raw result is provided back to the SVaaS Server that converts it into a proper XML result output format that is added to the BPCP resource. The SVaaS Connector can now access and render the result of solving the BPCP so that the business analyst can finally process the outcomes and fix his/her business process in case issues were reported. Alternatively the results can be consulted on the Cloud.

A. Business Process Compliance Problem (BPCP)

BPCP is an XML specification capturing all the relevant data that defines a Business Process Compliance Problem

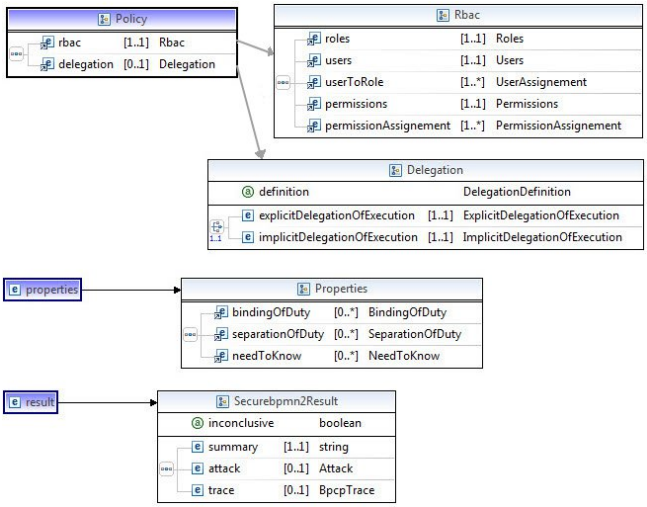

Figure 3: BPMN2-SEC: overview

within SVaaS. It is what the SVaaS Connector normally has to create in order to trigger the validation on the SVaaS Server. We handle BPCPs as REST resources each one comprising the following two elements:

- the business process workflow (in standard BPMN2 format) optionally augmented with more details on Data Objects and their task input/output
- the security-relevant aspects of the business process and corresponding validation results both specified in our own BPMN2 extension for security, referred to as BPMN2-SEC

For the following discussion of the main aspects of our BPMN2-SEC, while helpful, a deep understanding of the BPMN2 standard is not necessary.

Figure 3 depicts the three elements of BPMN2-SEC: *1)* the *policy* underlying the targeted business process and BPM Client, *2)* the *security properties* the business process is supposed to satisfy, and *3)* the validation *result* (if any already obtained). These elements are detailed hereafter.

1) Policy: The Policy element comprises both the role-based access control (RBAC) 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 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)
- Manager is assigned to Mickael (line 12)
- Two permissions are defined: one allows for executing the Approve Travel activity (lines 15-20) and the other one prohibits, via the `negate` construct, the execution of Request Travel (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)

Listing 1: BPMN2-SEC: RBAC example

```
1  <rbac>
2    <roles>
3       <role id="manager"><name>Manager</name></role>
4       <role id="staff"><name>Staff</name></role>
5       <role id="reception"><name>Reception</name></role>
6       ...
7    </roles>
8    <users>
9       <user id="mickael"><name>Mickael</name></user>
10      ...
11   </users>
12   <userToRole roleRef="manager" userRef="mickael" />
13   ...
14   <permissions>
15      <permission id="exe_approveTravel">
16         <action>execute</action>
17         <resource>
18            bpmn2:main#approvetravel
19         </resource>
20      </permission>
21      <permission id="noexe_requestTravel" negate="true">
22         <action>execute</action>
23         <resource>
24            bpmn2:main#requesttravel
25         </resource>
26      </permission>
27      ...
28   </permissions>
29   <permissionAssignement principalRef="manager"
30      permissionRef="exe_approveTravel" />
31   <permissionAssignement principalRef="reception"
32      permissionRef="noexe_requestTravel"/>
33   ...
34 </rbac>
```

The Delegation element allows for specifying the intended delegation policy employed by the BPM Client during the execution of the business process. 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. The BPMN2-SEC schema support the following standard delegation concepts:

- Delegation of execution: a user that was ready to execute a certain task, suddenly cannot do that anymore and delegates the execution of that task to another user. This delegation is only limited for the period of time necessary to execute the task.
- Delegation of permission: a user can delegate his/her permission to another user over a certain time-frame. This is particularly useful when a user is not available due to vacation or sick leave.

The BPMN2-SEC schema offers two ways for specifying the delegation policy:

- Implicit: a simple language allows for quickly specifying standard delegation policies
- Explicit: a set of delegation rules allows for specifying complex fine-grained delegation policies

Listing 2: BPMN2-SEC: Implicit delegation

```
1  /* Delegation to any: */
2  <implicitDelegationOfExecution>
3    <delegators>
4      <permitted>execute</permitted>
5    </delegators>
6    <delegatees>
7      <any />
8    </delegatees>
9  </implicitDelegationOfExecution>
10
11
12 /* BPM Client - NetWeaver BPM: */
13 <implicitDelegationOfExecution>
14   <delegators>
15     <permitted>execute</permitted>
16     <notProhibited>execute</notProhibited>
17   </delegators>
18   <delegatees>
19     <notProhibited>execute</notProhibited>
20   </delegatees>
21 </implicitDelegationOfExecution>
22
23 /* BPM Client - Activiti: */
24 <implicitDelegationOfExecution>
25   <delegators>
26     <permitted>execute</permitted>
27   </delegators>
28   <delegatees>
29     <permitted>execute</permitted>
30   </delegatees>
31 </implicitDelegationOfExecution>
```

For sake of simplicity we illustrate in Listing 2 some examples of implicit delegation policies:

- Delegation to any (lines 1-9): it captures a delegation policy where any users allowed to execute a task can delegate to any other user.
- BPM Client - SAP Netweaver BPM (lines 12-21): similar to the previous one, but the delegatee must be a user for which the execution of the task is not prohibited. This is the delegation policy used by SAP NetWeaver BPM in which potential owners can delegate the execution of an activity to any other user that is not in the excluded owner list of that activity.
- BPM Client - Activiti (lines 23-32): similar to the previous one, but the delegatee must be a user for which the execution of the task is allowed.

2) Security Properties: The Properties element lists the security properties that the business process should achieve. These are the property 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 SVaaS 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 should access only those sensitive data 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.

Listing 3 presents two simple properties referring to the 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).

Listing 3: BPMN2-SEC: property example

```
1  <properties>
2    <separationOfDuty id="sod1" maxUserActions="1"
3           minUsers="2">
4       <activityRef>bpmn2:main#requesttravel</activityRef>
5       <activityRef>bpmn2:main#approvetravel</activityRef>
6    </separationOfDuty>
7    <needToKnow id="needtoknow1">
8       <activityRef>bpmn2:main#approvetravel</activityRef>
9       <dataObjectRef>bpmn2:main#traveldata</dataObjectRef>
10      <privatefield>reason</privatefield>
11   </needToKnow>
12   ...
13 </properties>
```

3) Results: The Result element describes the validation result. Let us illustrate it via the example of Listing 4. The validation result is not inconclusive (line 1) meaning that the model checker was able to determine whether there is an attack or not (this is normally the case when the business process does not feature complex loops). 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.

Listing 4: BPMN2-SEC: Validation results

```
1  <securebpmn2:result inconclusive="false">
2    <securebpmn2:summary>
3       Separation of Duty between Request Travel and
4       Approve Travel
5    </securebpmn2:summary>
6    <securebpmn2:attacks>
7      <securebpmn2:attack name="Separation Of Duty"
8            propertyRef="securebpmn2:main#sod1" type="SoD">
9        <securebpmn2:par>karl</securebpmn2:par>
10       <securebpmn2:par>requesttravel</securebpmn2:par>
11       <securebpmn2:par>approvetravel</securebpmn2:par>
12     </securebpmn2:attack>
13   </securebpmn2:attacks>
14   <securebpmn2:trace>
15     <securebpmn2:step
16        flowElementRef="bpmn2:main#requesttravel"
17        name="Request Travel">
18       <securebpmn2:subStep type="claimed">
19          <securebpmn2:par>staff</securebpmn2:par>
20          <securebpmn2:par>karl</securebpmn2:par>
21          <securebpmn2:par>requesttravel</securebpmn2:par>
22       </securebpmn2:subStep>
23       <securebpmn2:subStep type="executed">
24          <securebpmn2:par>staff</securebpmn2:par>
25          <securebpmn2:par>karl</securebpmn2:par>
26          <securebpmn2:par>requesttravel</securebpmn2:par>
27       </securebpmn2:subStep>
28     </securebpmn2:step>
29     ...
30     <securebpmn2:step
31        flowElementRef="bpmn2:main#approvetravel"
32        name="Approve Travel">
33       <securebpmn2:subStep type="delegationOfpermission">
34          <securebpmn2:par>mickael</securebpmn2:par>
35          <securebpmn2:par>manager</securebpmn2:par>
36               <securebpmn2:par>karl</securebpmn2:par>
37          <securebpmn2:par>approvetravel</securebpmn2:par>
38       </securebpmn2:subStep>
39     </securebpmn2:step>
40     ...
41     <securebpmn2:step
42           flowElementRef="bpmn2:main#approvetravel"
43              name="Approve Travel">
44       <securebpmn2:subStep type="claimed">
45          <securebpmn2:par>manager</securebpmn2:par>
46          <securebpmn2:par>karl</securebpmn2:par>
47          <securebpmn2:par>approvetravel</securebpmn2:par>
48       </securebpmn2:subStep>
49       <securebpmn2:subStep type="executed">
50          <securebpmn2:par>manager</securebpmn2:par>
51          <securebpmn2:par>karl</securebpmn2:par>
52          <securebpmn2:par>approvetravel</securebpmn2:par>
53       </securebpmn2:subStep>
54     </securebpmn2:step>
55   </securebpmn2:trace>
56 </securebpmn2:result>
```

B. SVaaS architecture

A more detailed architecture view of SVaaS is depicted in Figure 4. The SVaaS Connector includes a Loader component to load from the BPM Client the 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 SVaaS 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 SVaaS Server. The Controller component coordinates the interaction among all the components of the SVaaS Connector. The Persistency component can be optionally implemented to enrich the SVaaS Connector in keeping track of all the validations that were carried out by business analysts within this specific SVaaS Connector. It is worth noticing that the BPM Client can opt for different integration strategy with SVaaS ranging from the most customizable one in which the SVaaS Connector components Loader and UIs are implemented with high customization for the targeted BPM Client, up to lighter integrations where, for instance, the rendering of the validation result or even

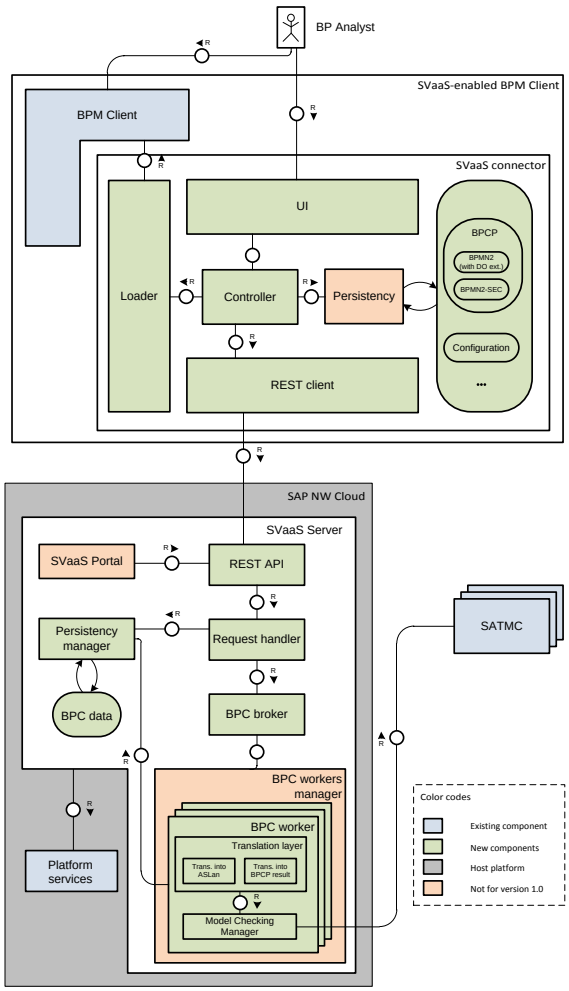

Figure 4: SVaaS architecture - medium level view

the entire validation activity (including security requirement specification) is outsourced to the SVaaS Server on the Cloud.

The SVaaS 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 [5] (this follows the lines of the translation presented in the original security validation approach [1]) 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 (this follows the approach presented in [1]). The validation result is now ready to be consulted. The SVaaS 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 SVaaS Portal also offers a full-fledged security validation environment available for those BPM Clients that wants to opt for a light integration with SVaaS. Indeed, even customers employing BPM Clients that are not augmented with SVaaS Connectors could get advantage of SVaaS by just accessing the SVaaS Portal and managing the entire security validation life-cycle there. Of course the level of interactivity and usability would definitely be not comparable to those BPM Clients featuring customized SVaaS Connectors. This is why we consider more promising those business scenarios in which BPM Clients are enriched with SVaaS Connectors.

C. Interaction protocol

As mentioned the SVaaS Connectors and the SVaaS 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 SVaaS 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 SVaaS 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 SVaaS 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. Figure 5 depicts the corresponding sequence diagram: in this example `123` is the value returned as resource location upon the POST request and two PUT requests are made to upload the BPMN2 and BPMN2-SEC elements.

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. Also it would be inefficient to ask the client to idle an HTTP request to the server, waiting the process to be finished. To answer this problem, we could have chosen to implement a WebSocket mechanism, however this technology being very young is not easily accessible. Same with long-polling techniques which can be

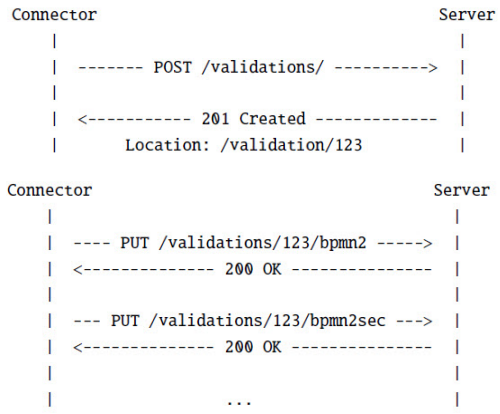

Figure 5: REST API: creation of BPCP

complex to implement for a SVaaS Connector. This is why SVaaS still recommends using a simple polling technique, with a reasonable interval of say 5-10 seconds. All along the flow of polling requests, the client will have to deal with different response status codes, in order to know the status of the validation and handle error that may happen during the validation process. Figure 6 shows the related self-explanatory sequence diagram.

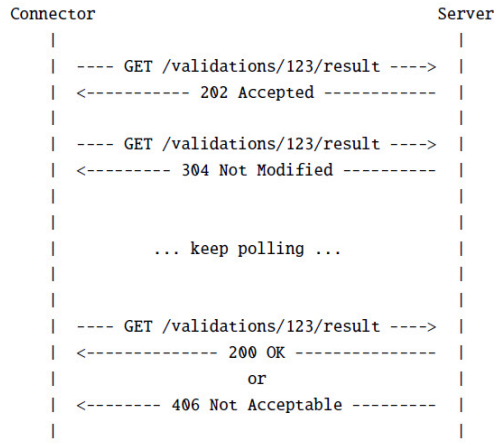

Figure 6: REST API: get result of BPCP

III. Lessons Learned

In [2] and [6] 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. Though likely both the on-demand and the on-premise BPM Clients could have been augmented with an implementation of the original security validation approach, this double effort was a clear obstacle as well as the not obvious aggregation of data outcomes from security validation activity performed on these two BPM Clients and by multiple business analysts. Similar commercialization obstacles were also perceived on the BPM Client software producer side for whom the integration of the Security Validation approach meant of course a nice-to-have differentiating feature, but also the inherited complexity of mandatory long-term maintenance that does not go very well with the idea of a research proof-of-concept that even integrates third-party academic modules. All in all, the following requirements motivated us to switch for a cloud-based solution:

- SVaaS 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;
- SVaaS shall be scalable with respect to multiple customer landscapes;
- SVaaS shall be flexible enough to offer various degree of integration within a BPM Clients ranging from the most customizable one (see *(a)*) up to the lightest/simplest one (see *(b)*):
  (a) the BPM Client is augmented with its own customized SVaaS UIs for e.g., specifying the security requirements of the business process under-design, rendering the results of the validation, etc;
  (b) 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;
- SVaaS 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;
- SVaaS shall be extensible enough to easily integrate new security properties within the validation life-cycle;
- SVaaS 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 SVaaS 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. In order to assess and demonstrate our overall SVaaS approach we focused on the proof-of-concept shown in Figure 7 in which the SVaaS Server is deployed on the SAP NetWeaver Cloud and its REST API is consumed by two BPM Clients, SAP NetWeaver BPM (see Figure 8) and Activiti (see Figure 9). Both these BPM Clients use an Eclipse-based business process design environment. This

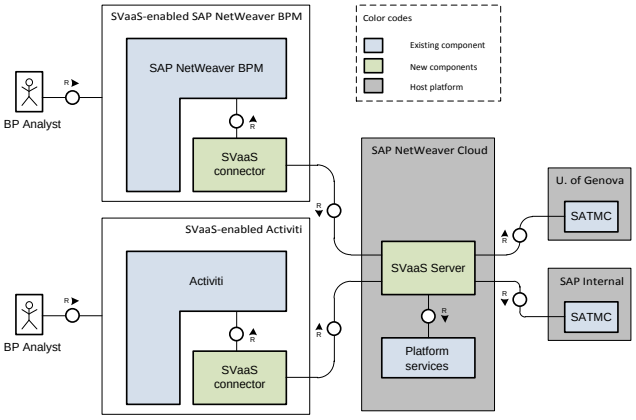

Figure 7: Proof-of-Concept

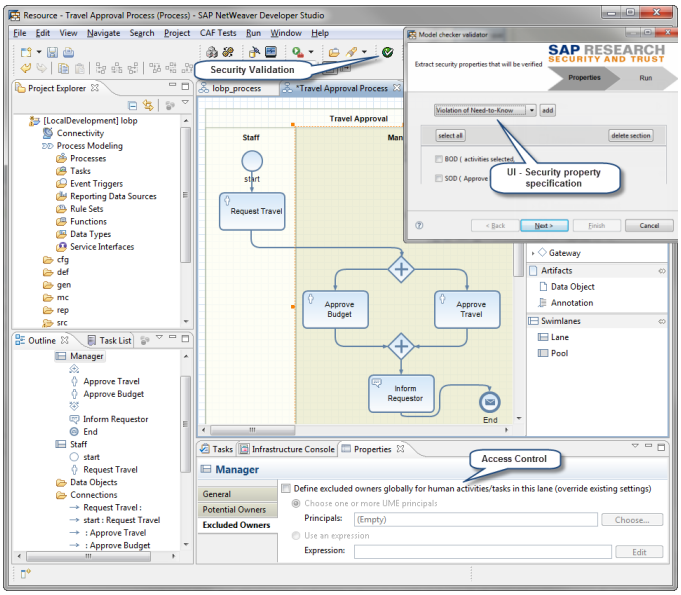

Figure 8: Security Validation within SAP NetWeaver BPM

is why we first developed a generic SVaaS Connector for the Eclipse environment and then we customized 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 [2]. This is simply due to more powerful machines hosting the SATMC model checker. More interesting, the SVaaS 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.

IV. Conclusion and Related Work

A. Related work

While there is a large body of literature extending business process modeling languages with means for expressing security and regulatory compliance properties, e. g., [7], [8], [9], [10] only a few approaches support a validation or testing of the specified properties. The closest related work is [11] which uses SPIN for checking that if an access control specification enforces binary static separation of duty and binding of duty constraints. Additionally,[6] 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 the strong need for checking the consistency of business processes itself, e. g., the absence of deadlocks. There are several works, e. g., [12], [13] that integrate these kind of process internal consistency validation checks locally into the business process modeling environment.

B. Conclusion and Further Work

In this paper we presented SVaaS, a promising approach and research prototype to test business process compliance. SVaaS takes advantage of the Cloud paradigm to provide on-demand security validation services to BPM systems. Once properly interfaced via a SVaaS Connector, the BPM Client is enabled to consume SVaaS 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 SVaaS architecture meets core business requirements collected during internal projects run at SAP with the ultimate goal of increasing its chances to reach industrial commercialization. We developed and deployed a proof-of-concept on top of our SVaaS 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 SVaaS 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 [6]. Last, but not least, we would like to explore if the availability of a common security validation technique like SVaaS could pave the way for *(i)* establishment of domain-specific repositories of compliance requirements accessible for any BPM system, and *(ii)* a

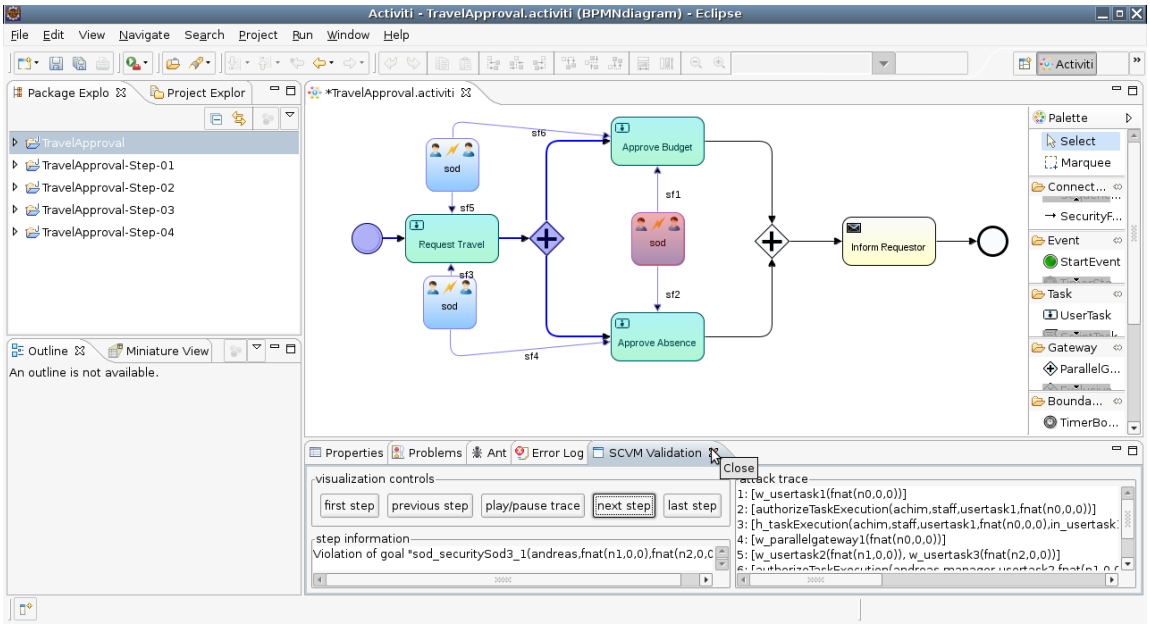

Figure 9: Security Validation within the Activiti BPMN Editor

systematic certification of business processes under-design that could be then compared in this regards and e.g., sold at difference prices depending also on the security and compliance they offer.

ACKNOWLEDGMENT

This work was partially supported by the FP7-ICT Projects ANIKETOS (no. 257930, http://www.aniketos.eu/home) and SPaCIoS (no. 257876, http://www.spacios.eu)

REFERENCES

[1] W. Arsac, L. Compagna, G. Pellegrino, and S. E. Ponta, "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.

[2] W. Arsac, L. Compagna, S. P. Kaluvuri, and S. E. Ponta, "Security validation tool for business processes," in *SACMAT*, R. Breu, J. Crampton, and J. Lobo, Eds. ACM, 2011, pp. 143–144.

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

[4] A. Armando, R. Carbone, and L. Compagna, "LTL Model Checking for Security Protocols," *Journal of Applied Non-Classical Logics*, vol. 19, no. 4, pp. 403–429, 2009.

[5] AVANTSSAR, "Deliverable 2.3: ASLan final version with dynamic service and policy composition," 2010, available at `http://www.avantssar.eu`.

[6] A. D. Brucker and I. Hang, "Secure and compliant implementation of business process-driven systems," in *Joint Workshop on Security in Business Processes (SBP)*, ser. Lecture Notes in Business Information Processing (LNBIP), vol. 132. Springer-Verlag, 2012.

[7] C. Wolter and A. Schaad, "Modeling of task-based authorization constraints in bpmn," in *BPM*, 2007, pp. 64–79.

[8] A. Rodríguez, E. Fernández-Medina, and M. Piattini, "A BPMN extension for the modeling of security requirements in business processes," *IEICE - Trans. Inf. Syst.*, vol. E90-D, pp. 745–752, March 2007.

[9] J. Mülle, S. von Stackelberg, and K. Böhm, "A security language for BPMN process models," University Karlsruhe (KIT), Tech. Rep., 2011.

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

[11] C. Wolter and C. Meinel, "An approach to capture authorisation requirements in business processes," *Requir. Eng.*, vol. 15, no. 4, pp. 359–373, 2010.

[12] R. M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN," *Information & Software Technology*, vol. 50, no. 12, pp. 1281–1294, 2008.

[13] W. M. P. van der Aalst, M. Dumas, F. Gottschalk, A. H. M. ter Hofstede, M. L. Rosa, and J. Mendling, "Correctness-preserving configuration of business process models," in *FASE*, 2008, pp. 46–61.