The Aniketos Service Composition Framework
Analysing and Ranking of Secure Services

Achim D. Brucker[1], Francesco Malmignati[2], Madjid Merabti[3], Qi Shi[3], and
Bo Zhou[3]

[1] SAP SE, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
`achim.brucker@sap.com`
[2] Selex ES S.p.A, A Finmeccanica Company, Italy
`francesco.malmignati@guests.selex-es.com`
[3] Liverpool John Moores University, Liverpool, United Kingdom
`{m.merabti, q.shi, b.zhou}@ljmu.ac.uk`

Abstract. Modern applications are inherently heterogeneous: they are
built by composing loosely coupled services that are, usually, offered and
operated by different service providers. While this approach increases the
flexibility of the composed applications, it makes the implementation of
security and trustworthiness requirements much more difficult. Therefore
there is a need for new approaches that integrate security requirements
right from the beginning while composing service-based applications, in
order to ensure security and trustworthiness.

In this chapter, we present a framework for secure service composition
using a model-based approach for specifying, building, and executing
composed services. As a unique feature, this framework integrates secu-
rity requirements as a first class citizen and, thus, avoids the "security as
an afterthought" paradigm.

Keywords: secure service composition, BPMN, service modelling, ser-
vice availability.

1 Introduction

A service-oriented architecture (SOA) provides a platform for services devel-
oped by different providers to work together [23]. Facilitated by standardised
inter-operation and description languages, such as WSDL [10], services can be
composed to form a larger application based on users' requirements.

The focus of research in SOA was traditionally on the realisation of service
composition in terms of how to construct the services so that they can work
together seamlessly. With the continuous development of SOA, it has been re-
alised lately that the security issue has become a barrier that hinders wider
application of SOA. Apart from the conventional security problems faced by
other systems, e. g., confidentiality, integrity, privacy and so on, the situation in
SOA is more complicated given the fact that the services are developed by dif-
ferent providers. Concerns over inconsistent security policies and configurations
must be addressed as top priority.

We propose a *secure* and *trustworthy* service composition framework that supports the service developer with the capability of composing services with security requirements in mind. The services are modelled and composed using a toolchain supporting the Business Process Model and Notation (BPMN) [19]. A service developer first constructs a BPMN service composition plan based on his/her functional requirements. It specifies what are the tasks needed and how these tasks interact with each other. We extend the BPMN notations so that certain security requirements can be specified within the BPMN composition plan as well. After searching for suitable services in an open marketplace, the abstract BPMN composition plan will be associated with concrete services for each task in the plan. The service composition is verified and guaranteed to comply with the service developer's security requirements before deployment.

Unlike other SOA solutions, our framework takes the security requirements into account during the service composition process. A service developer can specify his/her security needs directly in the extended BPMN composition plan so only those services that satisfy the security requirements will be selected. In addition, the service developer is given the flexibility to set priorities that will be used to quantify and compare service compositions, from all three aspects of security, quality of service, and cost. This is particularly useful when the service developer faces a wide range of choices.

2   The Aniketos Secure Service Composition Framework

Building secure and trustworthy composite services on top of a SOA is a challenging task. At *design-time* the service developer needs to select the optimal set of services that satisfies both the functional and security requirements put by the end user. At *runtime*, a service may become unavailable due to various reasons and has to be replaced automatically with alternative services that, at least, offer the same security and trust guarantees. In addition, the service developer also needs to decide if a given security property should be enforced statically or dynamically. One the one hand, a static enforcement creates less overhead at runtime, it reduces the flexibility of service substitution or re-composition. On the other hand, a dynamic enforcement is usually more flexible but requires more system resources at runtime. Thus, a service designer needs to balance the system resources while fulfilling the security and compliance requirements.

To support the service developer in building flexible, secure, and trustworthy services through composition, we propose a *secure service composition framework* that addresses both the design-time and runtime service compositions. In this chapter we focus only on the technical parts of the design-time process, i.e., we exclude the requirements elicitation, as well as the service deployment and runtime adaptation parts.

Figure 1 gives an high-level overview of the *Aniketos Service Composition Framework* which is the design-time modelling and analysis part of the Aniketos platform [4]. At the beginning, domain experts together with requirement engineers specify the high-level business process as well as the security and trust
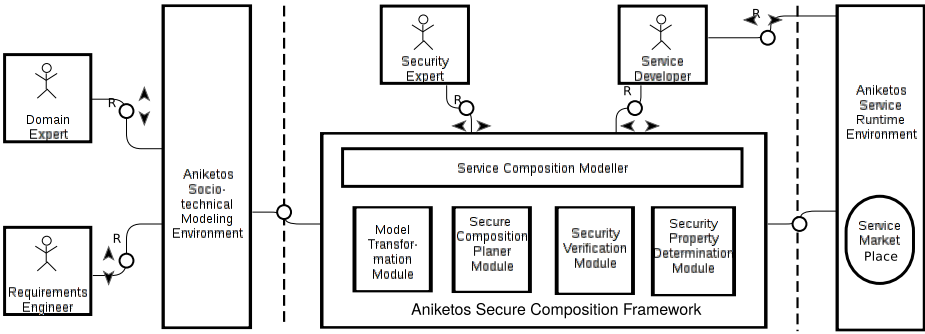
Fig. 1. The Aniketos Service Composition Framework.

requirements by using the *Aniketos Socio-technical Modelling Tool* [20]. It provides the opportunity to express security needs not just from technical, but also from social aspects. From these semi-formal descriptions, the *model transformation module* helps to generate composition plans, which are presented in BPMN format. These composition plans are coarse-grained. Thus, before these composition plans can be deployed in the *Aniketos Service Runtime Environment*, they will be refined by a service developer using the *Aniketos Secure Composition Framework*.

The *Aniketos Service Composition Framework* provides an Eclipse-based environment (the *Service Composition Modeller*) to the service developer for refining the composition plans as well as checking their security and trust properties. Specifically, the service developer can use the following component modules:

– *Model Transformation Module:* As described above, this module helps to generate the basic elements of the composition plan from the requirement document that expressed in the Aniketos Socio-technical Modelling Language [20] (see Chapter 5, Chapter 6, and Chapter 7).
– *Secure Composition Planer Module:* This module allows the service developer semi-automatically select the secure services for a given composition plan (see Section 4 for more details). To check the compositions comply with the security requirements, this module uses the Security Verification Module as well as the Security Property Determination Module.
– *Security Verification Module:* This module provides formal validation and verification solutions for composed services (and, not discussed in this chapter, atomic services [9]). For example, role-based access control and separation of duty properties (see Section 3 for details) are verified by the Security Verification Module (see Chapter 8 and Chapter 10).
– *Security Property Determination Module:* This module provides an uniform interface for accessing security properties of services. Moreover, this module stores the verification status of security properties to avoid an unnecessary (expensive) re-verification.

– *Service Marketplace:* This component registers and stores the services for open access. Secure Composition Planner Module selects services from the Service Marketplace (see Chapter 4).

The *Aniketos Service Composition Framework* supports composition of services, as well as the transformation from social to technical modelling of security requirements. It provides formal verification of these security requirements and helps the end user to choose the most suitable services. In the next two sections, we will focus on the two key research challenges: 1) Analysing the consistency of security properties and 2) quantifying and ranking service compositions.

3   Modelling and Verifying Security Properties

In this section, we present a validation approach for fine-grained separation-of-duty and binding-of-duty constraints. This work is implemented as core technique for the security verification module mentioned in Section 2 and assumes composition plans as discussed in Chapter 8.

3.1   Analysing SecureBPMN Models

Modelling non-functional requirements, right from the beginning, is important but it can only be the first step in building secure and trustworthy service-oriented systems, which requires various analysis techniques, e. g., for

1. checking the *internal consistency* of the security specification, for example ensure the access control requirements and need-to-know requirements do not contradict each other.
2. checking the *information (data) flow* on the process level to examine information flow requirements as well as high-level need-to-know requirements.
3. checking that process-level security requirements are fulfilled on the *implementation and configuration level*. This is particular important for implementation and configuration artifacts that are not generated in a model-driven approach.
4. checking that the service compositions (business processes) are *executable* if the security requirements are enforced, i. e., there exists a valid execution trace from a start to an end event.
5. analysing and *comparing* different techniques (e. g., resulting in different costs or runtime resource requirements) for implementing security requirements.

In this section, we discuss, as an example, an analysis method for checking the consistency between role-based access control (RBAC) and separation of duty (SoD)/binding of duty (BoD) specifications (see Chapter 10). This analysis method contributes to both the item 1 and item 5 mentioned above. Our modular architecture allows to integrate other analysis approaches easily and our prototypes already support other analysis as well. For example, [9] presents an analysis that contributes to the item 3.
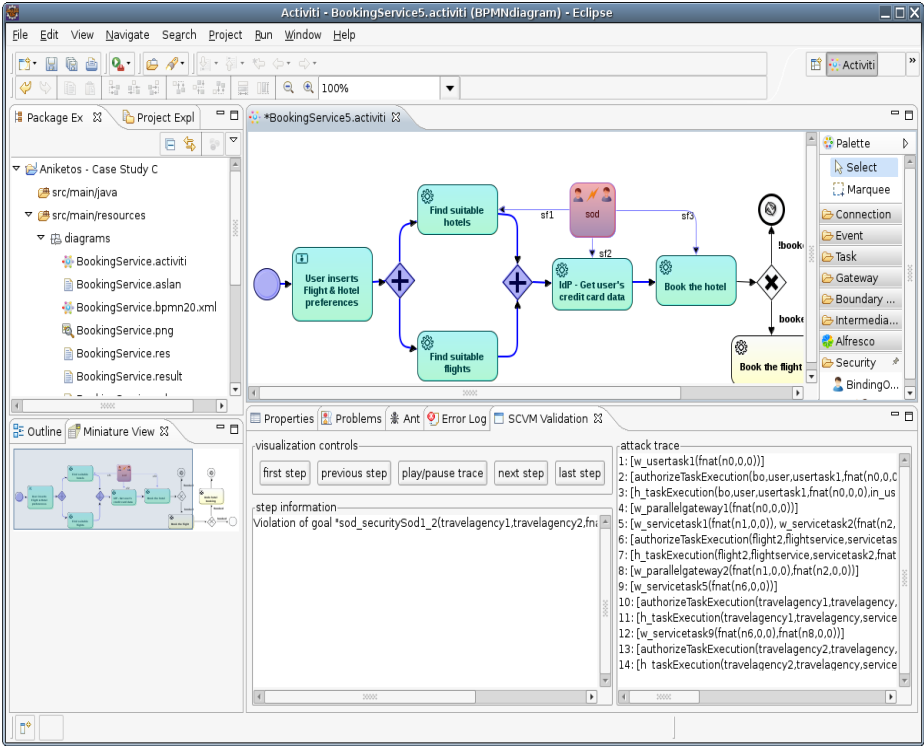
Fig. 2. Security Validation within the Activiti BPMN Editor

Adding constraints such as SoD or BoD to a system that is already restricted by RBAC results in questions like the following: "Is the SoD constraint already guaranteed by the RBAC configuration?" If an RBAC configuration ensures a SoD constraint, e. g., as two tasks are only executable by different roles r_1, r_2 and there is no user u_i that is assigned to both roles r_1 and r_2, we call this a *static separation of duty*. Otherwise, we call it a *dynamic separation of duty*.

While static separation of duty constraint do not need to be enforced at runtime and, thus, reduce the runtime costs, it requires to re-check the SoD constraint after each and every modification of the RBAC configuration (e. g., adding new roles, changing the role assignment of subjects). In contrast, dynamic separation of duty constraint requires a runtime check for each access to a resource that is constrained by the separation of duty. Although dynamic SoD is more flexible, it requires additional resources and, thus, costs, at runtime. Moreover, additional security checks might result in delays for users and, thus, might reduce the overall usability of the system.

To address these issues, we use an analysis method inspired by the work of Arsac et al. [5]. We extended their work significantly to support *n*-ary SoD (BoD) constraints as well as constraints on the level of constrained permission (instead

the task-level). As Arsac et al. [5], we use the AVANTSSAR tool suite (`www.avantssar.eu`) as back-end for our formal analysis. Consequently, we translate the service composition plan and its security requirements to ASLan [5], i.e., the input language of the AVANTSSAR tool suite. The choice of ASLan is based on two reasons: 1) the experiments carried out by Arsac et al. [5] show that ASLan is expressive enough to capture the requirements of security enriched service compositions and 2) the use of the same tools allows for developing a common verification back-end for our SecureBPMN-based approach as well as the approach developed by Arsac et al. [5]. In fact, we could demonstrate that the analysis can be provided as a cloud-based service that can be used by both modelling approaches [11].
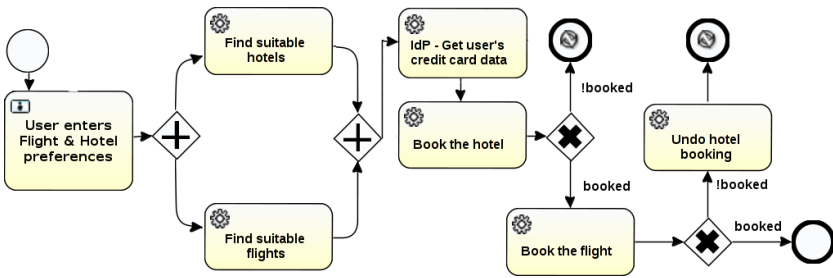

Fig. 3. A composed service for booking flights and hotels.

Assume, in our example from Chapter 8 (see Figure 3), we want to counterfeit fraud or price-fixing agreements. Therefore, we require that the services Find suitable flights and Book the flight are operated by different provides (and similarly, for the hotel booking). The actual RBAC configuration is inferred automatically from the information available in the service marketplace (i.e., the service-level agreements).

Our formal analysis translates the security configuration (here, RBAC and SoD/BoD) as well as the security properties that should be verified into the formal language ASLan [5]. In our example, the result of this translation (only an excerpt) for the security configuration looks like the following:

```
hc rbac_ac(Subject, Role, Task)
          := CanDoAction(Subject, Role, Task)
           :- user_to_role(Subject, Role), poto(Role, Task)
hc poto_T6 := poto(TravelAgency1, Find suitable flights)
hc poto_T7 := poto(TravelAgency1, Book the flight)
```

The security goal, in this case a SoD constraint between the services Find suitable flights and Book the flight looks as below:

```
attack_state sod_securitySod1_1(Subject0,Subject1,
                                Instance1,Instance2)
:= executed(Subject0,task(Find suitable flights,Instance1)).
   executed(Subject1,task(Book the flight;Instance2))
   &not(equal(Subject0,Subject1))
```

This configuration, obviously, violates the SoD constraint as the TravelAgency1 can do both searching for flights and booking them. In this case, a dishonest travel agency could prefer flights with a higher bonus for the travel agency that are not necessarily the cheapest for the traveller. This is detected by our formal analysis, e. g., the verification module returns the following "attack trace:"

```
 1. [w_usertask1(fnat(n0,0,0))]
 2. [authorizeTaskExecution(bo,user,usertask1,fnat(n0,0,0))]
 3. [h_taskExecution(bo,user,usertask1,fnat(n0,0,0),
                             in_usertask1,out_usertask1)]
 4. [w_parallelgateway1(fnat(n0,0,0))]
 5. [w_servicetask1(fnat(n1,0,0)),
                    w_servicetask2(fnat(n2,0,0))]
 6. [authorizeTaskExecution(flight1,flightservice,
                            servicetask2,fnat(n2,0,0)),
     authorizeTaskExecution(travelagency1,travelagency,
                            servicetask1,fnat(n1,0,0))]


                     ...

15. h_taskExecution(travelagency1, travelagency,
                    servicetask9,fnat(n8,0,0),
                    in_servicetask9,out_servicetask9)
```

Of course, this textual representation is not well-suited to practitioners. Therefore, we developed a user-friendly visualisation of such an attack in terms of the high-level composition plan (i. e., on the level of the BPMN model). Figure 2 shows how our prototype visualises such a violation to the service developer. Here, the service developer is able to manually step through all necessary actions that a dishonest agency would execute to actually violate the SoD constraint.

After such an analysis, the service developer needs to decide how to mitigate this risk. In general, there are several options, among them

– re-design the composition plan, to avoid the need for a particular separation of duty constraint,
– instruct the service composition framework to ensure the selection of different service providers, or
– enforce a dynamic separation of duty at runtime. For this, our prototype can generate configurations for XACML [18] based access control infrastructures.

Certainly, the concrete mitigation plan depends on the actual use case.

4   Quantifying and Ranking Service Compositions

The security property modelling and verification techniques allow the service consumer specify certain security properties that the service composition has to comply with. In practice, the number of compositions that satisfy the security requirements could still be large. Therefore another dilemma always faced by the service consumer is to make a choice from the service composition pools.

In this section, we introduce the mechanism used in Aniketos platform for quantifying and ranking service compositions, i. e., we support the service consumer in choosing, based on an automated recommendation, the most suitable service composition. This recommendation should be made based on the property

of the composition as a whole, rather than just based on individual sub-services in the composition. As a starting point, we try to solve this issue from three aspects, which are the three factors that mostly considered by service consumers: encryption (security), availability (QoS), and cost (business).

In most of the cases web services are made available together with a service-level agreement (SLA). SLA is a formal guarantee that has to be accepted by service consumers before the service being used. SLA normally specifies the properties of a service across different level. For example, on business level it describes what kind of functionality the service offers and how the users will be charged (cost); on technical level it may describe what kind of security protection is deployed (e. g., encryption) and the number of shutdowns the service might encounter each year (availability). We focus on these three properties in this section not only because they are normally included in the SLA, it is also because they are the properties that verifiable at runtime. For example, the availability of service can be easily recorded and calculated by examining the logs stored in the system.

This work is implemented as key part for the security composition planner module mentioned in Section 2.

4.1   Encryption – The Weakest Link

There are some cases when the weakest link principle is particularly applicable to service composition. It states that when services are composed together, the security capability of the composite service is equal to what the weakest service or link is offering. This security principle is applicable to many security properties and *encryption* is one of them. When encryption is applied to communications between services, the services may adopt different encryption algorithms or key lengths which give them different encryption strength. In order to communicate with each other, the service with advanced encryption algorithm may have to degrade its encryption strength during the composition. Thus the composite services literally use the weakest encryption strategy in part of their communications. For example, consider the case in Figure 4 where service A supports encryption algorithms of Blowfish and 3DES, service B supports Blowfish and AES, and service C supports 3DES and AES. To communicate with each other, the link between service A and B is encrypted with Blowfish and the link between B and C is encrypted with AES. Therefore the overall strength of the composition, in terms of keeping communications confidential, is the weaker one between Blowfish and AES.
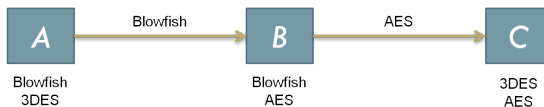

Fig. 4. Set Ranking Criteria.

In Aniketos, the weakest link principle is used to determine the security capacity of the composite services. It should be noted however, that the weakest link principle is not universally applicable. There are cases where alterations to a service composition can be utilised to improve the security of a composite service to be greater than that of the weakest component. An example might be where a firewall service is used to shield an otherwise vulnerable service from outside attack. The use of the firewall mitigates the vulnerability exposed by the weaker service. And vice versa it may also apply in reverse: the introduction of a component may serve as an exacerbating factor that reduces the security of the overall composition to a degree beyond that posed by the service were it to act in isolation. This often results from interactions between incompatible security properties.

To simplify the issue, in this study we focus on the encryption. Therefore each link between services is checked, and the encryption strength of the composition is determined by the weakest link, i. e.,

$$E = \min_{i=1}^{n} E_i$$

where E is the encryption strength of the composition and E_i is the encryption strength for each link i in the composition. E_i is determined by the strongest algorithm, which supported by both services at each end of the link i.

The quantitative value (from 0.9 to 0 in our case), however is predetermined by expertise in advance based on Table 1. Please note that as claimed in [14], the quantitatively ranking of encryption algorithms is possible but heavily depends on the metrics and target scenario. Table 1 is just a guideline and rather used to demonstrate our ideas in this front.

Table 1. Quantitative Value of Encryption Algorithms

Algorithm Name	Quantitative Value
Serpent	0.9
AES (Rijndael)	0.8
3DES	0.7
CAST128/256	0.6
Twofish	0.5
Blowfish	0.4
MARSH	0.3
Other algorithms	0.2
Codings	0.1
Plain text	0

4.2   Availability

Availability is another aspect being used to compare services. It relates to the quality of services (QoS). Availability in this scenario means the available time

ratio of a service. Unexpected shutdown of a service could cause severe damage to service consumers' business and service developer's reputation. Therefore seeking guarantee from service developer about the service availability is one of the top priories for service consumers, before they commit to use the service. The situation gets complicated in service composition because a composition's availability is decided by not only the technical specifications of the sub-services, but also by the structure of the composition.

Take the example of the travel agency in Figure 3 on page 126, most of the services are placed in sequential order. That means if one of the sub-service is not available, the entire composition will stop. Therefore the availability of sequential tasks is the product of all the sub-services' availability value in percentage. However, the services Find suitable hotels and Find suitable flights are executed in parallel. It means these two services can be carried out separately. Nonetheless they still have to be both finished before the next task Get user's credit card data can be executed. Therefore for parallel tasks the availability value is the minimum among them. For services that are exclusive to each other, the availability of the composition depends on which service has been eventually used.

Table 2 shows the rules that we used for calculating the availability of composite services. In this study we assume that the services are independent from each other. If in Figure 3 each service has the following availability value: Find suitable hotels: 0.99, Find suitable flights: 0.96, Get user's credit card data: 0.97, Book the hotel: 0.99, Book the flight: 0.98, and Undo hotel booking: 0.94. The *availability* value for a successful transaction will be calculated as:

$$A = \min(0.99, 0.96) \times 0.97 \times 0.99 \times 0.98 = 0.90$$

where A represents availability of the composition.

Table 2. Rules to Calculate Availability

	Description	Calculation
◯—▶◯	Sequence	$\prod_{i=1}^{n} A_i$
✦	Parallel	$\min(A_1, \ldots, A_n)$
✖	Exclusive	A_i

4.3   Cost

Finally the last factor that also plays important role in consumer's decision making is the *cost*. Higher security and quality of service normally means higher price, which must be within a consumer's budget. Comparing to *encryption* and *availability*, calculating the *cost* of a service composition is more straightforward. When discount is not considered, it is simply the sum of all the sub-services'

costs, i. e.:

$$C = \sum_{i=1}^{n} C_i$$

where C is the cost for the composition and C_i is the cost for sub-service i.

4.4   Ranking compositions

In Aniketos we implemented a simple user interface providing prioritising options so the service consumers can specify the criteria that they want to use to rank secure service compositions. As shown in Figure 5, the service consumer basically
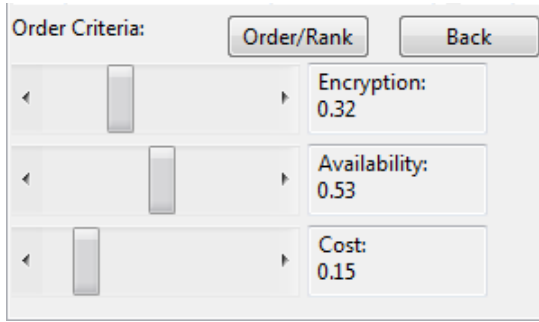
Fig. 5. Set Ranking Criteria.

can choose how much weights he wants to put on each criterion of encryption, availability, and cost. Assume the consumer sets the weights to 0.32, 0.53 and 0.15 respectively for availability and cost, the overall value V for each service composition will be:

$$V = 0.32 \times E + 0.53 \times A + 0.15 \times \frac{B - C}{B}$$

where E represents the value of encryption strength, A represents the value of availability, C represents cost, and B represents the consumer's budget. Apparently higher value of E and A, and lower value of C will result in greater value of V. In this way the generated service compositions can not only be security-wise verified by our SecureBPMN extensions, but also ranked easily based on consumer's other priorities.

The Aniketos platform targets secure service composition at both design-time and runtime. Therefore the prioritising options set by consumer at design-time will be stored in the consumer's policy configurations and referred back at runtime. So the ranking mechanism will still work on behalf of the consumer at runtime, in case the service composition changes.

Please note that the ranking is for service compositions that already satisfy users' requirements, i. e., the service compositions ranked here should have acceptable values in E, A and C first. This can be easily enforced by put threshold values for each of the criteria.

5 Conclusion and Related Work

5.1 Related Work

We see three areas of related work: 1. modelling of security requirements for process models, 2. analysing security properties of process models, and 3. determining security of composite services.

There is a large body of literature extending graphical modelling languages with means for specifying security or privacy requirements. One of the first approaches is SecureUML [16], which is conceptually very close to our 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 analysing SecureUML models, e. g., [6] or [8]. While based on the same motivation, UMLsec [15] is not defined using a meta-model. Instead, the security specifications are written, in an ad-hoc manner, in UML profiles. Integrating security properties into business processes is a quite recent development, e. g., motivated by [25]. In the same year, [21] presented a meta-model based approach introducing a secure business process type that supports global security goals. In contrast, our approach allows the fine-grained specification of security requirements for single tasks or data objects. Similar to UMLsec, [17] presented 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 [24] and [5] that both support the checking if an access control specification enforces binary static of duty and binding of duty constraints. Apart from security properties, there is also a strong need for checking the consistency of business process itself, e. g., the absence of deadlocks. There are several works that concentrate on this kind of process internal consistency validation, e. g., [12] and [2]. Moreover, there are several approaches for analysing access control constraints over UML models, e. g., [22], [8], and [15]. These approaches are limited to simple access control models, as the UML models are usually quite distant from business process descriptions that comprising high level security and compliance goals.

Last but not least, determining the properties of composite service based on its sub-services is another area that attracts attentions from research community. Most of the work related to security focus on determining trust due to its subjectiveness and openness. This undermines the need for solutions to determine other properties as well. [13] described how to present trustworthiness for composite services based on various factors such as reputations and qualities of

the services. The method took the structure of the composition into account. [28] proposed a classification method that abstracts and quantifies service compositions based on five key security aspects: confidentiality, integrity, availability, accountability and non-repudiation. There are also other works that focus on security properties of system-of-systems such as [27] and [26]. Comparing to these works, our approach concentrates on the most objective and justifiable properties in encryption, availability and cost, which represents security, QoS and business respectively. Our solution also gives the flexibility to the service consumers so they can decide how to rank the service compositions themselves.

5.2   Lessons Learned

We discussed our approach with various business process modelling experts at SAP SE. Overall, these experiences show that our approach is applicable to a wide range of applications domains.

Our evaluation showed that the discussed security and compliance requirements can be expressed at the business process level. Moreover, they are sufficient for most modelling needs. Still, in particular our telecommunication case study raised the need for various notions of confidentiality. As such, confidentiality is not (yet) supported by SecureBPMN; currently, SecureBPMN only supports a very specific form, the need-to-know-principle. Confidentiality, in terms of requiring encrypted communications between the different services (tasks) is another important requirement. The choice of the correct encryption technology (in fact, on a technical level, we need to ensure that data is only communicated over authenticated and secured channels) requires a multitude of technical decisions (e. g., encryption algorithms, length of cryptographic keys). As these are merely technical decisions, we can only record the high-level requirement on the process level and need to refine them interactively during the implementation of a secure service composition.

Moreover, our evaluation showed that in practice, most service compositions are rather small (e. g., less than 15 services or tasks). On these sizes of models, our formal analysis usually is able to validate security or compliance properties within less then 20 seconds. While this is fast enough for the (interactive) design of service compositions, it is too slow for automatic service re-compositions at runtime. Therefore, the efficient caching, which needs to ensure the authenticity and validity, of validation results is of outermost importance.

5.3   Conclusion and Future Work

In this chapter we presented an integrated framework for modelling, analysing, and ensuring secure service compositions. This framework, called *Aniketos Service Composition Framework* is part of a larger platform that supports the end-to-end (i. e., ranging from the requirements elicitation to the actual operation of the developed system) development of secure and trustworthy SOA and cloud-based systems. This end-to-end integration is a unique feature of our approach that not only enables traditional security and consistency analysis on the model

and implementation level, it also supports certain types of economical analysis approaches that allow the service consumers to decide between different security solutions based on their encryption strengths, availabilities and costs.

There are several lines of future work. One of them is the development of support for system audits, e. g., by integrating analysis techniques such as [1] or [3]. In particular, process mining approaches appear to be particularly interesting: combining process mining with our business process animation, i. e., the visualisation of attack traces, allows interactive investigation of the deviations of the actual service composition execution with the intended one. Moreover, we are also interested in the integration analysis techniques that check the internal consistency of processes, e. g., [12], as well as their reconfiguration, e. g., [2]. Finally, we intend to integrate security testing approaches, e. g., [7], for validating the compliance of services and (legacy) back-end systems in a black-box scenario.

References

[1] 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)

[2] 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-Verlag (2008)

[3] 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-Verlag (2010)

[4] Aniketos: Deliverable 5.1: Aniketos platform design and platform basis implementation (2011)

[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-Verlag (2011)

[6] Basin, D., Clavel, M., Doser, J., Egea, M.: Automated analysis of security-design models. Information and Software Technology **51**(5), 815–831 (2009)

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

[8] 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-Verlag (2006)

[9] Brucker, A.D., Hang, I.: Secure and compliant implementation of business process-driven systems. In: Rosa, M.L., Soffer, P. (eds.) Joint Workshop on Security in Business Processes (SBP), *LNBIP*, vol. 132, pp. 662–674. Springer-Verlag (2012)

[10] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web services description language (WSDL) 1.1. Tech. rep., W3C (2001)

[11] Compagna, L., Guilleminot, P., Brucker, A.D.: Business process compliance via security validation as a service. In: Oriol, M., Penix, J. (eds.) Testing Tools Track of ICST. IEEE Computer Society (2013)

[12] Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology **50**(12), 1281–1294 (2008)

[13] Elshaafi, H., McGibney, J., Botvich, D.: Trustworthiness monitoring and prediction of composite services. In: ISCC, pp. 580–587 (2012)

[14] Jorstad, N., Landgrave, T.S.: Cryptographic algorithm metrics. In: 20th National Information Systems Security Conference (1997)

[15] Jürjens, J., Rumm, R.: Model-based security analysis of the german health card architecture. Methods Inf Med **47**(5), 409–416 (2008)

[16] 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-Verlag (2002)

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

[18] OASIS: eXtensible Access Control Markup Language (XACML), version 2.0 (2005). URL `http://docs.oasis-open.org/xacml/2.0/XACML-2.0-OS-NORMATIVE.zip`

[19] Object Management Group: Business process model and notation (BPMN), version 2.0 (2011). Available as OMG document formal/2011-01-03

[20] Paja, E., Dalpiaz, F., Poggianella, M., Roberti, P., Giorgini, P.: Modelling security requirements in socio-technical systems with sts-tool. In: Kirikova, M., Stirna, J. (eds.) CAiSE Forum, vol. 855, pp. 155–162 (2012)

[21] 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)

[22] 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-Verlag (2005)

[23] Welke, R., Hirschheim, R., Schwarz, A.: Service-oriented architecture maturity. Computer **15**(1), 662–674 (2011)

[24] Wolter, C., Meinel, C.: An approach to capture authorisation requirements in business processes. Requir. Eng. **15**(4), 359–373 (2010)

[25] 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-Verlag (2007)

[26] Zhou, B., Arabo, A., Drew, O., Llewellyn-Jones, D., Merabti, M., Shi, Q., Waller, A., Craddock, R., Jones, G., Arnold, K.L.Y.: Data flow security analysis for system-of-systems in a public security incident. In: ACSF, pp. 8–14 (2008)

[27] Zhou, B., Drew, O., Arabo, A., Llewellyn-Jones, D., Kifayat, K., Merabti, M., Shi, Q., Craddock, R., Waller, A., Jones, G.: System-of-systems boundary check in a public event scenario. In: SoSE (2010)

[28] Zhou, B., Llewellyn-Jones, D., Shi, Q., Asim, M., Merabti, M., Lamb, D.: Secure service composition adaptation based on simulated annealing. In: ACSAC, pp. 49–55 (2012)