

# A Framework for Secure Service Composition

Achim D. Brucker  
SAP AG

Vincenz-Priessnitz-Str. 1  
76131 Karlsruhe, Germany  
achim.brucker@sap.com

Francesco Malmignati  
Selex ES S.p.A

A Finmeccanica Company  
Italy  
francesco.malmignati@guests.selex-es.com

Madjid Merabti Qi Shi Bo Zhou  
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 difficult. As the number of security requirements is increasing dramatically, there is a need for new approaches that integrate security requirements right from the beginning while composing service-based applications.

In this paper, 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 security requirements as a first class citizen and, thus, avoids the “security as an afterthought” paradigm.

## I. INTRODUCTION

A service-oriented architecture (SOA) provides a platform for services that developed by different providers to work together [23]. The focus of research in SOA has been on the realisation of service composition in terms of how to construct the services so that they can work together seamlessly. With its continuous development, it has been realised lately that the security issue has become a barrier that hinders wider application of SOA. Apart from the conventional security problems that faced by other systems, e.g., confidentiality, integrity, privacy and so on, the situation in SOA often is more complicated given the fact that the services are developed by different providers. Concerns over inconsistent security policies and configurations must be addressed as top priority.

As part of the work undertaken for the Aniketos project (<http://www.aniketos.eu/>), we propose a framework for building and executing *secure* and *trustworthy* service compositions. The services are modelled and composed using a toolchain supporting the Business Process Modelling 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 extended 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 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 also given the flexibility to set priorities that will be used to quantify and compare service compositions, from the aspects of availability and cost. This is particularly useful when the service developer faces a wide range of choices. To the best of our knowledge, this work is the first complete SOA platform that not only offers secure composite services at design-time, but also targets for the runtime operations.

## II. BACKGROUND: SOA AND BPMN

### A. Service-Oriented Architecture and Its Security

A *service* is a unit that provides certain functionality. The service-oriented architecture (SOA) allows users to reuse existing services depending on their requirements. Therefore services can be composed to form a larger application in an *ad hoc* manner. SOA platforms provide a foundation for modelling, planning, searching for and composing services. They specify the architectures required, as well as providing tools and support for service composition standards.

To facilitate service composition across different platforms, service modelling languages are used to describe the business requirements of a system and system resources. By expressing processes, in languages such as the Business Process Execution Language (BPEL) [20] or the Business Process Modelling and Notation (BPMN) [19], not only the services can be easily understood and composed, also the compositions can be validated against desired criteria and modified to suit required changes in operation. For describing the web service interfaces, the Web Services Description Language (WSDL) evolved as the de-facto standard.

Subsequent standards have been proposed to augment the basic description of WSDL, to add semantic, behavioural, and to a limited extent, authentication and security data. For example, Unified Services Description Language (USDL) [17] consists of standards that target trust and security, to bridge the previously-identified vendor divide. However, none of them tackles the security issue of SOA in the first place.



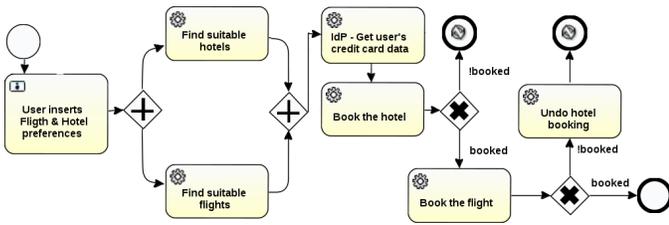


Fig. 1. A composed service for booking flights and hotels.

### B. Using BPMN to Construct Service Compositions

The modelling in BPMN is done by expressing business processes through business models. A BPMN model is an executable specification of the workflow, i.e., a flowchart based diagram that captures the basic structure and flow of activities and data within a business process. From a high-level perspective, the development of a system using BPMN is divided into two major phases:

- 1) During the *design phase*, a service developer—together with domain or business experts—designs the process model, i.e., the *service composition plan*. This process model comprises both automatic services, as well as human interactions with these services.
- 2) During the *deployment phase*, the process model is deployed in a business process execution engine, which can act as a service orchestrator.

This high-level view does not include several other tasks involved in system development, e.g., the implementation of actual services and design of user interface.

Figure 1 shows a BPMN diagram modelling a service composition that provides a travel booking service to customers. First, the customer enters his/her flight and hotel preferences into the system (such kind of user interactions are modelled by **user tasks** in BPMN). Next, two web services (modelled as **service tasks**) are executed and connected via **parallel gateways**. These web services can be operated by different service providers and, in our example, provide functionalities for finding suitable hotel and flight information respectively. Here the parallel gateways ensure that the service which queries customer's credit card data will only be executed if both the Find suitable hotels and Find suitable flights tasks are terminated successfully. By using **exclusive gateways** the service developer is able to indicate that the Book the hotel task might fail. In case the booking is failed (!booked), an error boundary event will be reached.

In our simple example, to avoid fraud or price-fixing agreements, we could demand that the services for finding hotels (flights) and the booking service, are from different service providers. Moreover, only authenticated users will be allowed to authorise a booking and the service providers should be trustworthy. In the next section we will explain our approach to extend the standard BPMN modelling process so these security requirements can be accommodated.

## III. A SECURE SERVICE COMPOSITION FRAMEWORK

A secure service composition framework should provide security at both *design-time* and *runtime*. At design-time the service developer will 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 needs to decide if a given security property should be enforced statically or dynamically. While a static enforcement creates less overhead at runtime, it reduces the flexibility of service substitution or re-composition. In contrast, dynamic enforcement is usually more flexible but requires more system resources at runtime. Thus, service developers have to consider economical aspects as well for realising security and compliance requirements.

To support the service developer in building flexible, secure, and trustworthy services through composition, we developed the *Aniketos Secure Composition Framework* as part of the Aniketos platform [3].

The *Aniketos Secure 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, among others, use the following component modules:

- *Model Transformation Module*: This module infers a draft composition plan from the functional and security requirement document that expressed in the Aniketos Socio-technical Modelling Language [21], by domain experts together with requirement engineers.
- *Secure Composition Planer Module*: This module allows the service developer semi-automatically select the secure services for a given composition plan (see Section V). 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 paper, atomic services [10]). For example, role-based access control and separation of duty properties (see Section IV) are verified by the Security Verification Module.
- *Security Property Determination Module*: This module provides a uniform interface for accessing security properties of services. Moreover, this module stores the verification status of security properties to avoid 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.

The *Aniketos Secure 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.

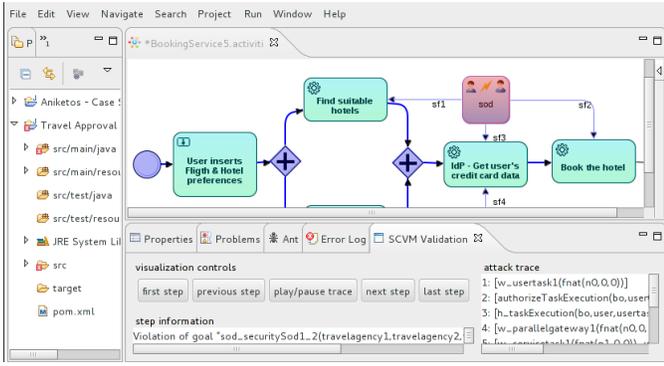


Fig. 2. Security Validation within the Activiti BPMN Editor

#### IV. MODELLING AND VERIFYING SECURITY PROPERTIES

In this section, we present an extension of BPMN, called SecureBPMN, that integrates security specifications into BPMN. Moreover, we present a validation approach for fine-grained separation-of-duty and binding-of-duty constraints.

##### A. Modelling Security Properties in BPMN: SecureBPMN

Modelling security properties, as a first class citizen of *business processes* or *service composition plans* expressed in BPMN, requires an integrated language for expressing both security and functional requirements. We address this need with SecureBPMN, a meta-model-based ([8] discuss the details of meta-model-based language extensions) security language that is integrated into the BPMN meta-model [19]. SecureBPMN supports, among others, the specification of:

- *Role-based access control (RBAC)*: SecureBPMN contains a hierarchical role-based access control language supporting arbitrary constraints on the permissions.
- *Permission-level separation and binding of duty*: SecureBPMN models  $n$ -ary separation of duty (SoD) and binding of duty (BoD) as authorisation constraints. This results in a fine-grained notion of these properties on the level of single permissions.
- *Delegation*: SecureBPMN supports the delegation of tasks with and without transfer of access rights.
- *Need-to-know principle*: This mainly refers to restrictions on access to process variables or data objects, given the process can model internal data-flow as well as the communication to (external) services.

Due to space limitations, more advanced features of SecureBPMN, such as the support for break-glass access control policies (similar to the work of [11]), history-resets for binding-of-duty for processes with loops (similar to the work of [5]), or negotiable delegations (in contrast to orders) are not discussed in this paper.

##### B. Analysing SecureBPMN Models

As an example, we discuss an analysis method for checking the consistency between RBAC and SoD/BoD specifications. Our modular architecture allows to integrate other analysis approaches easily (for details, see Brucker and Hang [10]).

Supporting SoD together with RBAC raises questions like “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 *static SoD*. Otherwise, we call it *dynamic SoD*.

While a static SoD does not need to be enforced at runtime and, thus, reduces 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, a dynamic SoD constraint only requires a runtime check for each access to certain resources that are constrained by separation of duty. Therefore dynamic SoDs are more flexible, but require additional resources and cost more at runtime. To address these issues, we extend the work of Arzac et al. [4] with support for  $n$ -ary SoD (BoD) constraints on the permission level (instead of task level). Similar to Arzac et al. [4], we use the AVANTSSAR tool suite (www.avantssar.eu) to support our formal analysis at the back-end. Consequently, we translate the service composition plan and its security requirements to ASLan [4], 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 Arzac et al. [4] 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 commonly-used verification back-end for different approaches. In fact, we could show that the analysis can be provided as a cloud-based service supporting multiple front-ends [12].

In our example (recall Figure 1), to counterfeit fraud or price-fixing agreements, we assume that the services Find suitable flights and Book the flight are operated by the same provider (TravelAgency1). The actual RBAC configuration is inferred automatically from the information available in the service marketplace (i.e., the service-level agreements). Our formal analysis tool will translate the security configuration as well as the security properties that should be verified into the formal language ASLan as below (simplified and shortened):

```

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)

```

On the other hand the security goal, e.g., a SoD constraint between the services Find suitable flights and Book the flight looks as follows:

```

attack_state sod_securitySod1_2(Subject0, Subject1,
    Instance1, Instance2)
:= executed(Subject0, task(Find suitable flights, Instance1)).
   executed(Subject1, task(Book the flight, Instance2))
   &not (equal(Subject0, Subject1))

```

Obviously the security configuration violates the SoD constraint as the TravelAgency1 has been put in the position that can do both searching for flights and booking them. It casts risks that a dishonest travel agency could prefer flights with

higher profits over flights that provide better value for money to the travellers. 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)

```

This textual representation is not well-suited to practitioners. Therefore, we provide a user-friendly visualisation at the service composition level. 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 travel agency would execute to 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.

## V. 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 sanctify 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 that is used in Aniketos platform for quantifying and ranking service compositions.

### A. Quantifying service compositions by Availability and Cost

Service compositions should be quantified and ranked based on the property of the composition, rather than just based on its sub-services. The property of a composition depends not only on the properties of the sub-services, but also on the way they have been constructed. As a starting point, we try to solve this issue from business perspective. In many business cases, apart from functionality and security, the two factors that have been mostly considered by service consumers are *availability* (service downtime ratio) and *cost*.

Availability 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 priorities for

TABLE I  
RULES TO CALCULATE AVAILABILITY

	Description	Calculation
	Sequence	$\prod_{i=1}^n A_i$
	Parallel	$\min(A_1, \dots, A_n)$
	Exclusive	$A_i$

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 1 on page 648, 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. In contrast, 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. Thus 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 I shows the rules that we used for calculating the availability a composite service. Assume in Figure 1 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.

Comparing to *availability*, calculating the *cost* of a service composition is more straightforward. It is 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  $i$ .

### B. 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. It basically allows the service consumer to set weights on each criterion of availability and cost (Figure 3). Assume the consumer sets the weights to 0.72 and 0.28 respectively for availability and cost, the overall value  $V$  for

Fig. 3. Set Ranking Criteria.

each service composition will be:

$$V = 0.72 \times A + 0.28 \times \frac{B - C}{B}$$

where  $A$  represents the value of availability,  $C$  represents cost and  $B$  represents the consumer's budget. Apparently higher value of  $A$  and lower value of  $C$  will result in greater value of  $V$ . In this way the generated service compositions are not only 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.

## VI. CASE STUDIES

We implemented a prototype of the framework based on the Activiti BPMN tool suite (<http://www.activiti.org>) and evaluated this prototype in the context of two industrial case studies.

### A. Case Study 1: e-Government

The e-Government domain provides a real-life scenario where the usage of composite service is relevant and that can benefit from the practical application of our framework. In this scenario a service developer is appointed by a Municipality to build a web application that allows an interested buyer to find an available lot in a specific geographical area. Through this application, lot owners or real estate agents can provide information about the lots they intend to make available for sale. The main security concern in this case is to ensure that the lot information is published in an accurate and transparent manner.

### B. Case Study 2: Future Telecommunication Services

Telecommunication domain is an area of constant change, where the TLC operators leverage now on web services paradigm to provide a new set of integrated IT and Telco services. The competition between the operators is getting intensified as they are looking for enhanced telecommunication services to further increase their revenue from mobile and Internet services.

The Aniketos framework effectively comes to help in this scenario, allowing a TLC operator, as described in Section III, to compose and expose secure and trustworthy services. A TLC operator, by exploiting the framework, can discover service components conforming its security requirements in

the Aniketos *Marketplace* and then securely expose its network resources as new web services to its end users.

### C. Lesson Learnt

Due to page limitations we only mentioned these case studies in brief. 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. Choosing the appropriate encryption technology (in fact, on technical level, we have to ensure that data is only communicated over authenticated and secure channels) requires making 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 this scale, our formal analysis usually is able to validate security or compliance properties within less than 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 be ensured by authenticity and validity, of validation results is of utmost importance.

## VII. CONCLUSION AND RELATED WORK

### A. Related Work

There is a large number of literature extending graphical modelling languages with means of specifying security or privacy requirements (e.g., [15, 16, 22]). Conceptually, the closest is SecureUML [16], with which we share the same sublanguage for specifying RBAC. The 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 [9]. With respect to the validation of security requirements on the business process level, closely related works include [24] and [4] that both support the checking if an access control specification enforces binary static separation and binding of duty constraints. Determining the properties of composite service based on its sub-services is another challenge. [14] calculates the trustworthiness of composite services based on various factors such as reputations and qualities of the services. Zhou et al. [27] proposes a classification method that abstracts and quantifies service compositions based on key security aspects such as confidentiality, integrity and accountability. Some works such as [26] and [25] try to determine security properties for system-of-systems.

## B. Conclusion and Future Work

We presented the *Aniketos Secure Composition Framework* for modelling, analysing, and ensuring secure service compositions. This framework 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 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 economical analysis approaches that allow the service consumers to decide between different security solutions based on their availability and costs.

There are several lines of future work, among them the development of support for system audits, e. g., by integrating analysis techniques such as [2]. 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 interested in the integration analysis techniques that check the internal consistency of processes, e. g., [13], as well as their reconfiguration, e. g., [1]. 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.

### ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant no. 257930 (<http://www.aniketos.eu/>).

### 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, LNCS*, vol. 4961, pp. 46–61. Springer-Verlag (2008)
- [2] 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)
- [3] Aniketos: Deliverable 5.1: Aniketos platform design and platform basis implementation (2011)
- [4] 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)
- [5] Basin, D., Burri, S.J., Karjoth, G.: Separation of duties as a service. In: *ASIACCS*, pp. 423–429. ACM Press (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 healthcare applications. In: *SACMAT*, pp. 133–142. ACM Press (2011)
- [8] Brucker, A.D., Doser, J.: Metamodel-based uml notations for domain-specific languages. In: Favre, J.M., Gasevic, D., Lämmel, R., Winter, A. (eds.) *Workshop on Software Language Engineering (ATEM)* (2007)
- [9] 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)
- [10] 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)
- [11] Brucker, A.D., Petritsch, H.: Extending access control models with break-glass. In: Carminati, B., Joshi, J. (eds.) *SACMAT*, pp. 197–206. ACM Press (2009)
- [12] 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)
- [13] Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. *Information & Software Technology* **50**(12), 1281–1294 (2008)
- [14] Elshaafi, H., McGibney, J., Botvich, D.: Trustworthiness monitoring and prediction of composite services. In: *ISCC*, pp. 580–587 (2012)
- [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] Marienfeld, F., Höfig, E., Bezzi, M., Flügge, M., Pattberg, J., Serme, G., Brucker, A.D., Robinson, P., Dawson, S., Theilmann, W.: Service levels, security, and trust. In: Barros, A., Oberle, D. (eds.) *Handbook of Service Description: USDL and its Methods*, chap. 12, pp. 295–326. Springer-Verlag (2012)
- [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] Organization for the Advancement of Structured Information Standards: Web services business process execution language (BPEL), version 2.0 (2007). Available as *OASIS standard*
- [21] 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)
- [22] 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)
- [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] 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)
- [26] 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)
- [27] 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)