Compliance in Service-oriented Architectures: A Model-driven and View-based Approach

Huy Tran[*,a,1], Uwe Zdun[a], Ta'id Holmes[b], Ernst Oberortner[b], Emmanuel Mulo[b], Schahram Dustdar[b]

[a] *Software Architecture Group, Faculty of Computer Science, University of Vienna, Austria*
[b] *Distributed Systems Group, Institute of Information Systems, Vienna University of Technology, Austria*

Abstract

Context: Ensuring software systems conforming to multiple sources of relevant policies, laws, and regulations is significant because the consequences of infringement can be serious. Unfortunately, this goal is hardly achievable due to the divergence and frequent changes of compliance sources and the differences in perception and expertise of the involved stakeholders. In the long run, these issues lead to problems regarding complexity, understandability, maintainability, and reusability of compliance concerns.
Objective: In this article, we present a model-driven and view-based approach for addressing problems related to compliance concerns.
Method: Compliance concerns are represented using separate view models. This is achieved using domain-specific languages (DSLs) that enable non-technical and technical experts to formulate only the excerpts of the system according to their expertise and domain knowledge. The compliance implementations, reports, and documentation can be automatically generated from the models. The applicability of our approach has been validated using an industrial case study.
Results: Our approach supports stakeholders in dealing with the divergence of multiple compliance sources. The compliance controls and relevant reports and documentation are generated from the models and hence become traceable, understandable, and reusable. Because the generated artifacts are associated with the models, the compliance information won't be lost as the system evolves. DSLs and view models convey compliance concerns to each stakeholder in a view that is most appropriate for his/her current work task.
Conclusions: Our approach lays a solid foundation for ensuring conformance to relevant laws and regulations. This approach, on the one hand, aims at addressing the variety of expertise and domain knowledge of stakeholders. On the other hand, it also aims at ensuring the explicit links between compliance sources and the corresponding implementations, reports, and documents for conducting many important tasks such as root cause analysis, auditing, and governance.

Key words: Compliance, model-driven, view-based, service-oriented architectures, process-driven SOAs, domain-specific languages

1. Introduction

In general, compliance, in the context of information systems, means ensuring that the software and systems of an organization act in accordance with multiple established laws, regulations, and business policies (from now on called compliance sources) [1]. Compliance is a major issue in many organizations because any compliance violation will

[*] Corresponding author
Email addresses: huy.tran@univie.ac.at (Huy Tran), *uwe.zdun@univie.ac.at* (Uwe Zdun), *tholmes@infosys.tuwien.ac.at* (Ta'id Holmes), *e.oberortner@infosys.tuwien.ac.at* (Ernst Oberortner), *e.mulo@infosys.tuwien.ac.at* (Emmanuel Mulo), *dustdar@infosys.tuwien.ac.at* (Schahram Dustdar)
[1] Software Architecture Group, Faculty of Computer Science, University of Vienna, Berggasse 11/Top 2, A-1090 Vienna, Austria. Phone: +43-1-4277-39693. Fax: +43-1-4277-396 99.

lead to severe financial penalties or losses of reputation. We highlight two important issues that hinder the compliance of organizational software and systems.

Firstly, organizations have to deal with an increasing number of diverse compliance sources, such as the Basel II Accord [2], the International Financial Reporting Standards (IFRS) [3], the Markets in Financial Instruments Directive (MiFID) [4], the French financial security law (LSF) [5], Tabaksblat [6], or the Sarbanes-Oxley Act (SOX) [7], to name just a few. These compliance sources generally prescribe business practices for a wide range of compliance domains such as risk management, privacy, security, quality of services, intellectual property or licensing. It is very difficult to devise a *one-size-fits-all* representational language or model that is able to accommodate the divergence of compliance sources in the software and systems of a certain organization. Instead, in the current practice, compliance concerns are implemented on a per-case basis using ad-hoc, hard-coded solutions that are undesirable because they are hard to maintain, hard to evolve or change, hard to reuse, and hard to understand. Moreover, this practice makes it difficult to systematically and quickly keep up with constant changes in regulations, laws, and business policies.

Secondly, compliance cannot be implemented and enacted solely by either business experts (or compliance experts) or IT experts but rather involves an enterprise-wide scope. The fact that compliance sources are typically specified in highly abstract legal writing requires business expert (or compliance experts) to interpret and translate them into concrete requirements. Subsequently, IT experts (e.g., software engineers or system administrators) have to ensure that their software and systems meet these requirements. The aforementioned process of implementing compliance must be documented and periodically reported to the executive boards or the auditors [7]. Unfortunately, each stakeholder group has different interests, knowledge, and expertise, and often their work is performed at very different abstraction levels. For instance, domain and compliance concepts and knowledge are primarily formulated by business and compliance experts at analysis and design time. These experts are, however, often not familiar with software and system engineering practices, which are specialization areas of the IT experts involved in implementing, deploying, enacting, and maintaining organizational software, systems, and compliance. In addition, from the managers' or auditors' perspectives, adequate, timely reports and documentation of the processes and internal controls that adhere to the relevant laws, regulations, and business policies are the most important indicators.

To the best of our knowledge, none of existing approaches to business compliance have fully addressed the aforementioned issues. A number of existing approaches to business compliance have been proposed but they rather focus on particular compliance concerns and/or particular development phases (see [8–34]; discussed in detail in Section 5).

In this article, we propose a model-driven development (MDD) approach [35–37] for overcoming these issues. We support stakeholders in dealing with the divergence of compliance sources by using domain-specific languages (DSLs) which can be tailored to directly accommodate concepts and rules from particular compliance domains [37]. To support involvement of both non-technical stakeholders (e.g., business and compliance experts) as well as technical stakeholders (e.g., software engineers and system administrators) in the software, system, and compliance engineering process, a separation into high-level, domain-oriented and low-level, technical DSLs is provided. The great advantage of this separation of abstraction levels is that we can provide different stakeholders with appropriate representations to formulate the domain problem (i.e., compliance concerns) according to their particular expertise. Moreover, the representations of compliance concerns in terms of DSLs can be independently developed and evolved, which means we are able to keep up with the constant changes in existing regulation or creation of new regulation.

However, this raises the challenge of integrating these different compliance DSLs as well as correlating the compliance DSLs with the existing business functionality. So far, Tran et al. [38] have developed a View-based Modeling Framework (VbMF) – a specialization of the MDD paradigm – that provides a number of view models for specifying a service-oriented architecture (SOA). This approach enables the integration and traceability of different view models as well as the generation of executable code out of these models [39–41]. Thus, we exploit this important capability and extend VbMF with a Compliance Metadata model that serves as a bridge between compliance DSLs designed by business and compliance experts, the compliance requirements and compliance sources on the one hand, and the business functionality of software and systems on the other hand. In addition, we devise a number of other components extending the framework: a model validator statically validates the compliance concerns at design time, and code generators create components for supporting runtime enactment or monitoring compliance as well as to generate reports and documentation for auditing and compliance demonstration purposes. Our approach aims at supporting the systematic and automated implementation of various kinds of compliance concerns including controls in QoS policies, license policies, security policies, and others.

In the scope of this article, we consider process-driven SOAs – a particular kind of SOAs utilizing processes to

orchestrate services – to exemplify our approach. The rationale behind the selection of SOAs is that enterprises increasingly rely on service-oriented architectures for creating complex distributed software systems as well as leverage process-centric information systems to automate their business processes and services. A more detailed discussion of this kind of SOAs is given in Section 2.1. In this article, we present VbMF as a means to support integrating business compliance concerns in process-driven SOAs. To illustrate these capabilities of VbMF we will present one central field of compliance concerns in detail: QoS-related compliance concerns. To illustrate the involvement of the different stakeholders to model the required QoS compliance concerns, a DSL is exemplified which was developed especially for the QoS domain. In the same way, other DSLs developed for other concerns such as licensing [42] and security [43] (not presented in detail in this article) can also be integrated using VbMF in the same manner.

The remainder of this article is organized as follows. Section 2.1 explains and illustrates process-driven SOAs as the working context of this article. Next, Section 2.2 describes the problem of dealing with compliance in SOAs in detail. Our view-based model-driven approach to supporting compliance in SOAs is elaborated in Section 3. A CRM Fulfillment process with QoS compliance concerns from an industrial case study illustrates our approach and the realization of our view-based, model-driven compliance framework in Section 4. Section 5 discusses and compares our approach to the relevant literature. Finally, a summary and an outlook on future research are provided in Section 6.

2. Background

2.1. Process-driven Service-oriented Architectures

SOAs are typically realized as layered architectures [44, 45]. Based on a communication layer that abstracts from platform and communication protocol details, a remoting layer that provides the typical functionalities of distributed object frameworks, such as request handling, request adaptation, and invocation. Service clients invoke service providers using this infrastructure. In a *process-driven* SOA, a service composition layer is provided on top of the client application/service provider layer. This layer provides a process engine (or workflow engine) that orchestrates services to realize individual activities in a business process.

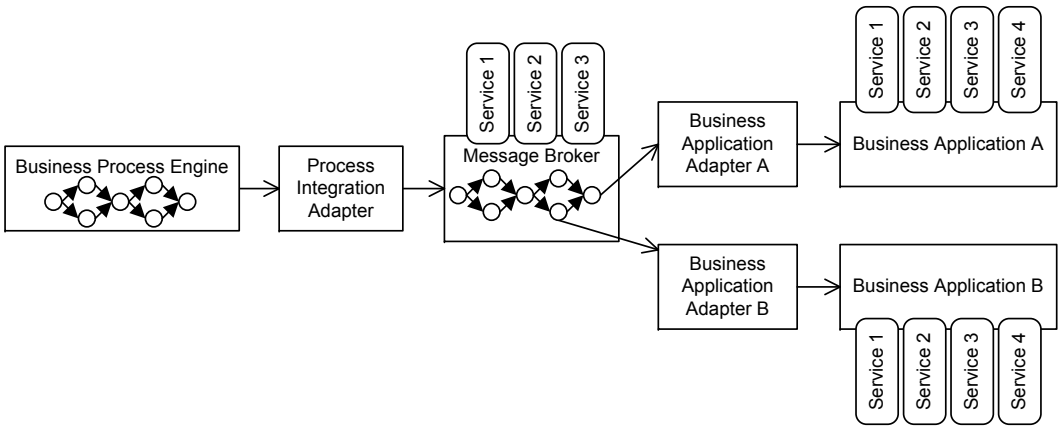

Figure 1: Illustrative small-scale SOA

The main goal of such process-driven SOAs is to increase the productivity, efficiency, and flexibility of an organization via process management. This is achieved by aligning the high-level business processes with the applications supported by IT. Changes in business requirements are carried out as changes in the high-level business processes. The processes are implemented by linking their activities to existing or new IT-supported applications. Organizational flexibility can be achieved because explicit business process models are easier to change and evolve than hard-coded business processes. In the long run, the goal is to enable business process improvement through IT.

A small-scale process-driven SOA is illustrated in Figure 1. A single business process engine accesses a service-based message broker, e.g., offered by an Enterprise Service Bus (ESB), via service-based process integration adapters. Service-based business application adapters are used to access back-end components, such as databases or legacy

systems. A typical SOA in enterprise organizations today is much larger than this illustrative example. That is, multiple process engines – e.g., one per department – are deployed, plus multiple instances of all other components. We limit the scope of our compliance approach to these process-driven SOAs. Apart from the notion of processes, which is specific to process-driven SOAs, other concepts and elements considered in our approach such as services, service collaborations, data objects exist in most of SOA-based systems. In the next section, we describe various kinds of compliance concerns that might occur in a process-driven SOA.

2.2. Compliance Concerns Occurring in SOAs

The compliance laws and regulations like the Basel II Accord or SOX cover issues such as auditor independence, corporate governance, and enhanced financial disclosure. Although these are typical cases for compliance, there are other compliance concerns, with similar characteristics, that occur in process-driven SOAs including:

- Compliance to *service composition policies*: There might be specific service composition rules for the SOA that must be met. For instance, a service can require a specific interface or behavior of another service so that they can be composed, or a service collects a client's private data for only a particular purpose, e.g., a credit card number is collected and disclosed only to check the account solvency and for nothing else.
- Compliance to *service deployment policies*: The runtime composition might be governed by business rules as well. For example, the business might require that a service can only interact with other service instances deployed within the same geographical borders, to ensure location-based data correctness.
- Compliance to *service sequencing or ordering policies*: It is possible that services may be allowed to compose only in specific orders. For instance, the business may require that a credit validation process is triggered before a shipment process is started.
- Compliance to *information sharing/exchange policies*: This applies to service conversations that follow some information sharing or exchange protocol. For instance, to get information from a service, a requestor must use a specific message type describing the information of interest. In response to such requests, the service may send a message to the requestor with a locator for the requested information. The requestor can subsequently obtain the requested information.
- Compliance to *security policies*: The business may have specific security requirements that are also pervasive throughout the SOA, such as the nondisclosure of personal data.
- Compliance to *QoS policies*: The business may require compliance of the runtime infrastructure to a certain quality of service (QoS). For instance, a specific performance window, a maximum latency, a particular mean downtime (i.e., availability), or a certain throughput, may be required to fulfill a service-level agreement (SLA).
- Compliance to *business policies*: The business side may provide specific policies for running the services, triggered by certain business events. For instance, it might be a business policy in a company that, upon a server failure, an SMS notification is sent to an administrator.
- Compliance with *jurisdictional policies*: Such policies occur in situations where a service produces a product in location under different legal jurisdictions. For instance, selling or buying a car in different EU countries involves different activities, taxes, and fees.
- Compliance to *intellectual property and licenses*: In a service composition there is the need to respect individual services licenses and terms of use. For instance, a service could include both royalty-based operations and freely available operations, only available for non-commercial use. A composed service has to comply with these licensing clauses, also in a service composition created on-demand.

Clearly, all of these concerns are driven by the business requirements. The concerns arise in process-driven SOAs, firstly, because SOA is often the integration architecture of an organization, and therefore, usually concerns all architectural components that must comply with certain business requirements; also, process-driven SOAs include

high-level abstractions such as business processes, which implies that the concerns should be integrated in the business process perspective offered to the business experts.

There might be compliance issues that can belong to more than one of the above categories. In fact many elements and concepts of models and DSLs used in our approach for describing various process-driven SOAs and compliance concerns in fact relate to the others. Nevertheless, the categorization aims at describing different aspects of SOAs that might be influenced by laws and regulations and therefore, need to be considered. It merely serves as background knowledge and does not influence our approach in the paper.

An integrated compliance framework can reduce development and maintenance costs for the IT in large companies, and enable small companies to compete. Business compliance can achieve additional goals: it can be understood as a business and technology enabler: When tackled using a strategic implementation approach based on a sound architecture, compliance concerns can drive a business and offer added value beyond solely meeting specific compliance demands. Please note that the goal of our approach is *not* to implement all of the compliance concerns listed above, but rather provide the means for an organization to implement any such compliance concern in process-driven SOA in traceable, changeable, understandable, and reusable fashion.

3. Model-driven approach to supporting compliance in SOAs

In this section, we discuss our model-driven and view-based approach for compliance in SOAs in detail. We firstly give a brief overview of the approach. All aspects of the approach will be explain in detail, including (i) the view-based modeling framework and compliance DSLs for explicitly formulating process-driven SOA and compliance concerns and (ii) a Compliance Metadata model for explicitly representing the relationships between compliance sources, view models, and DSLs and generating compliance documentation and reports.

3.1. Approach overview

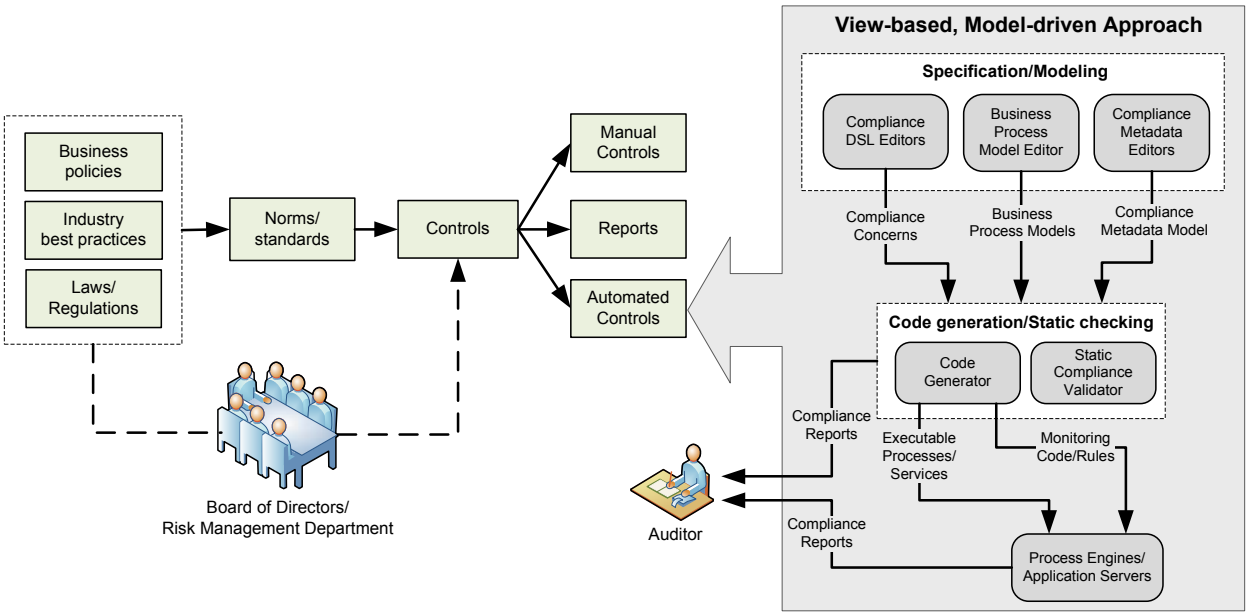

Figure 2: Overview of the view-based, model-driven approach for supporting compliance in SOAs

On the left hand side of Figure 2, we depict a high-level overview of our approach in relation to the typical view on compliance from an auditor's perspective. As described before, many compliance requirements are derived from different sources such as laws, regulations, and business policies. Such compliance sources can be realized using a number of so-called *controls*. A control is any measure taken to assure a compliance requirement is met. For instance,

an intrusion detection system, a penetration test, or a business process realizing separation of duty requirements are all controls for ensuring systems security. Most of compliance sources primarily focus on the "*what*" (i.e., what controls are needed), rather than on "*how*" to realize the controls. Thus, the regulations are often mapped to established *norms* and *standards* describing more concretely how to realize the controls for a regulation. Controls can be realized in a number of different ways, including reports, manual controls (e.g., controls that require human intervention), or automated controls (e.g., controls that can be executed without human intervention).

Our work shown on the right hand side of Figure 2 focuses on *automated controls* in the area of process-driven SOAs (i.e., mainly processes and services are considered). Our goal is to provide a unique framework for realizing all automatic controls in this realm and support as many automated controls as possible (that is, potentially increase the level of automation). This is achieved by an architecture covering the whole compliance life cycle: A view-based, model-driven framework is introduced for *modeling* or *specifying* the processes, services, and compliance concerns – to realize the automatic controls. In addition, metadata about the compliance controls is modeled to document the compliance controls. Some compliance concerns can be *statically checked* at design time, for instance, static checking of separation of duties. For many compliance concerns this is not possible: it is necessary to monitor and assess them at runtime. Hence, the code for implementation and supporting runtime monitoring the compliance concerns are *generated*. Besides, compliance control documentation and reports for auditing and demonstrating purpose are also automatically generated. The generation step in our approach is realized using model-driven development (MDD) paradigm [35–37]. In MDD, domain-specific languages (DSLs) are used as specification languages that are tailored to be particularly expressive in a certain problem domain. In our approach, DSLs are used to engage different stakeholders into the SOA and compliance engineering process.

One important aspect of our MDD approach for realizing a compliance framework is that it provides one or more DSLs on top of a model, either in textual or graphical syntax, representing the content of the abstract syntax (aka the language models) in a user-friendly way. That is, the DSLs' syntaxes are targeted at the end-user of the DSL. For instance, if a business process is shown, technical experts might prefer a textual syntax that is machine-processable and includes the more technically detailed concerns in various views (such as BPEL/WSDL-specific views). A business expert might rather prefer a graphical syntax that omits the technical details and only shows the high-level control flow augmented with compliance concerns.

3.2. View-model-based compliance framework

Our compliance framework for SOAs uses the model-driven approach to compose business processes and services as a foundational layer. To enable reuse and integration of both compliance concerns and service compositions, the compliance framework shall augment business process specifications, such as the Business Process Model And Notation (BPMN) [46], Business Process Execution Language (BPEL) [47], etc., with compliance concerns. As there are multiple other, similar concerns on which compliance concerns can be based than the process specifications, such as service specifications, collaboration specifications, data specifications, etc., and even the process specifications can use multiple specification types (such as BPEL, BPMN, and UML Activity Diagrams [48]), we abstract each of these basic concerns and each of the compliance concerns in its own model. This, however, imposes the challenge of how to integrate the various models.

We have solved this problem using a view-based approach (this is explained in detail in [38, 40, 41, 49–51]). In this approach, a view is a representation of a process from the perspective of related concerns. A view is specified using an adequate view model. Each view model is a (semi)-formalized representation of a particular SOA or compliance concern. Therefore, the view model specifies entities and their relationships that can appear in the corresponding view.

As mentioned in Section 2.2, there are many different kinds of business compliance that companies have to consider. Each of those compliance concerns embodies distinct concepts and constraints which are merely interpreted by domain experts or compliance specialists. Therefore, our approach introduces different DSLs to support those experts in better eliciting such compliance concepts and constraints and applying the resulting compliance concerns in the specific context of business processes. We present our approach for modeling process-driven systems and the compliance concerns in Figure 3.

On the left hand side, the modeling framework provides view models for describing the functionality of process-driven SOA-based software systems. As stated in [38, 49], the Flow, Collaboration, and Information view models represent the essential concerns of a business process. Other concerns, such as transactions, human integration, event

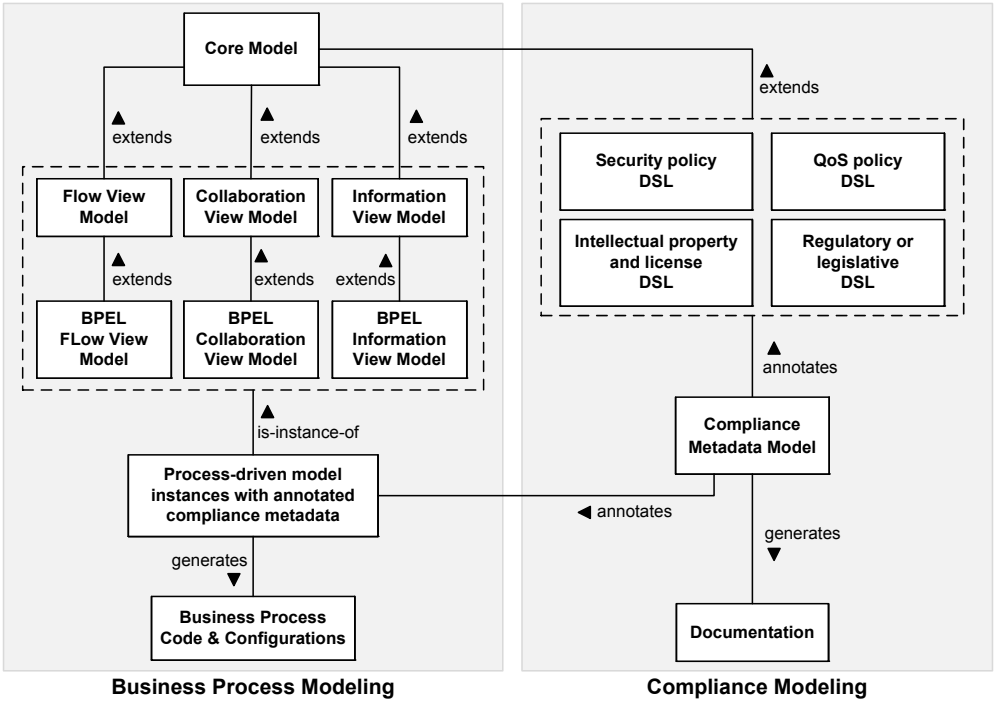

Figure 3: View-based approach for modeling processes and business compliance

handling, traceability, etc., are also developed and integrated to VbMF accordingly thanks to its extensibility [38, 40, 51]. For the sake of readability and concentration on compliance modeling, these process view models are not presented in Figure 3. Dashed rectangle boxes are used to form logical modeling groups, for instance, a group of view models or compliance DSLs, in the sense that the elements inside a group shall naturally be treated in the same way if not explicitly specified otherwise.

On the right-hand side, our approach offers facilities for describing compliance concerns in terms of DSLs, such as DSLs for representing security policy, QoS policy, intellectual property and licenses, and regulatory or legislative provisions. Each DSL can be seen as an extension view model that derives and enriches the basic elements of the Core model. That is, these DSLs represent extensional view models for expressing compliance controls in parts of the SOA that are not directly related to processes and services. Using extension mechanisms described in [38, 49], the framework can be extended with additional DSLs for expressing various kinds of compliance concerns. More details on extending the framework with additional compliance DSLs are elaborated in Section 3.3. Without loss of generality, we will illustrate in this paper these capabilities of VbMF for one of these DSLs: the DSL for specifying QoS-related compliance concerns. Other DSLs can be devised and integrated to VbMF in the same manner.

Whereas the DSLs mentioned above define compliance controls in specific areas, namely, security, QoS, intellectual property and licenses, and regulatory or legislative provisions, there is another DSL, namely, Compliance Metadata model, that has a special purpose: To annotate the controls defined both in the VbMF process views (e.g., the Flow view, Collaboration view, Information view, etc.) as well as the controls defined in the four extensional DSLs with compliance metadata. Details on the Compliance Metadata model are presented in Section 3.4. Using the Compliance Metadata model, all compliance controls can be annotated with metadata about the compliance, such as which regulation, standard, rules, compliance requirements, and so on, have led to the design and implementation of the control. The main goal of the Compliance Metadata model is hence to support the automatic documentation of all compliance-related concerns (cf. Section 4.5.1). In other words, the Compliance Metadata model allows us to not only model "*how*" compliance is achieved, but also "*why*".

VbMF provides three important mechanisms for formulating and manipulating view models: *view extension*, *name-base view integration*, and *code generation* [38, 49]. The view extension mechanism enables VbMF to be

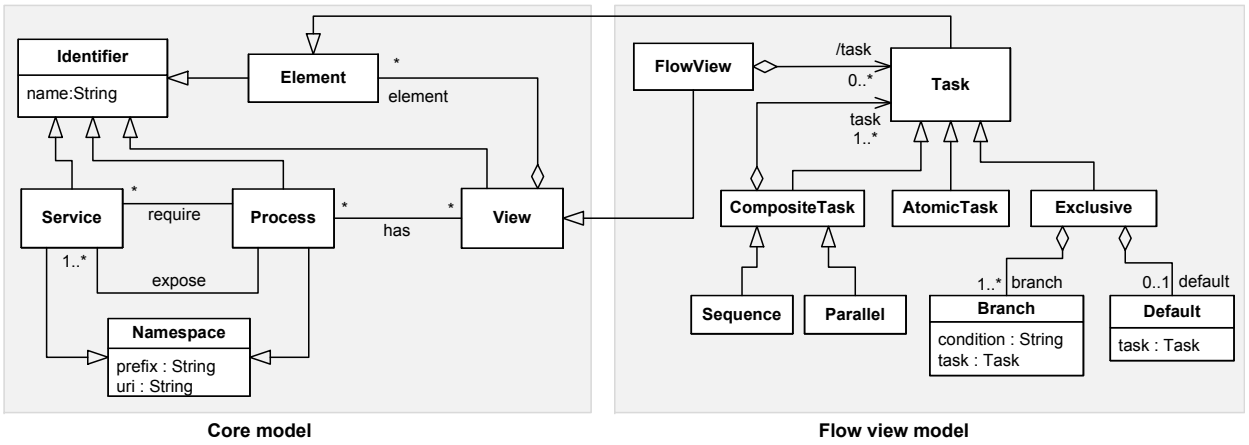

Figure 4: The Core model and the Flow view model

extended to additional modeling concerns by adding new view models that derive and enrich existing view models and view elements. The Core model shown in Figure 4 is the basis for creating the other view models. The Flow view model, which derives and extends the basic concepts of the Core model, represents the control flows of business processes. Further examples of existing extensions are including views for data object access [52] and human interactions [40]. In addition to the aforementioned horizontal extensions, VbMF also considers the separation of levels of abstraction by organizing view models into two essential layers: the abstract layer and technology-specific layer [38, 49]. The abstract layer includes the views without the technical details such that the business experts can better understand and manipulate these views. The technology-specific layer contains the views that embody concrete information of technologies or platforms (e.g., the BPEL-* views in Figure 3).

The view extension mechanism facilitates the separating of concern principle to reduce the complexity of tangled process concerns by using different view models and enhance the flexibility in formulating these view models independently. According to the specific needs of the stakeholders, views might have to be combined to provide a richer view or a more thorough view of a certain process. This can be achieved by using the name-based matching view integration mechanism presented in [41, 50]. The view integration mechanism enables loose-coupling cross references among view models based on the names of view elements and supports the integration of view models based on these cross references, which are called *view integration points*. During the course of view integration, static validation can also be leveraged for checking the conformity and consistency of the involving view models. The name-based matching technique has been proved to be very effective at the view level. It can also be enhanced further by incorporating other model merging approaches such as those using database schema matching, class hierarchical structures, or ontology-based structures. Further details on the name-based matching view integration can be found in our other works [41, 50]. In this article, the name-based matching view integration is the basis for relating view models with compliance DSLs and the Compliance Metadata model. Last but not least, code generation mechanism will take view models such as the low-level view models for describing the technology specifics of the business processes, the compliance DSLs (cf. Section 3.3), the Compliance Metadata model, and so on, as inputs for producing code and configurations that are necessary for deploying and monitoring the execution of the business process [38, 49].

We present a small example of modeling the business process of a Travel Booking agency [53] using VbMF in Figure 5. The Travel Booking process (see Figure 5(a)) starts when a customer initiates an itinerary request. After updating the customer's profile for later promotions or advertisements, the process invokes three other services for booking airline tickets, hotels, and cars, respectively. Finally, the process sends back an itinerary confirmation to the customer. The diagram in Figure 5(a) is drawn using BPMN notations [46] to visualize the main function of the Travel Booking process.

The aforementioned description of the Travel Booking process is modeled by using VbMF's views. The Travel Booking Flow view (see Figure 5(b)) specifies necessary tasks to fulfill the customer's request and the execution order of these tasks. The details of each task are not embodied in the Travel Booking Flow view, but represented in other

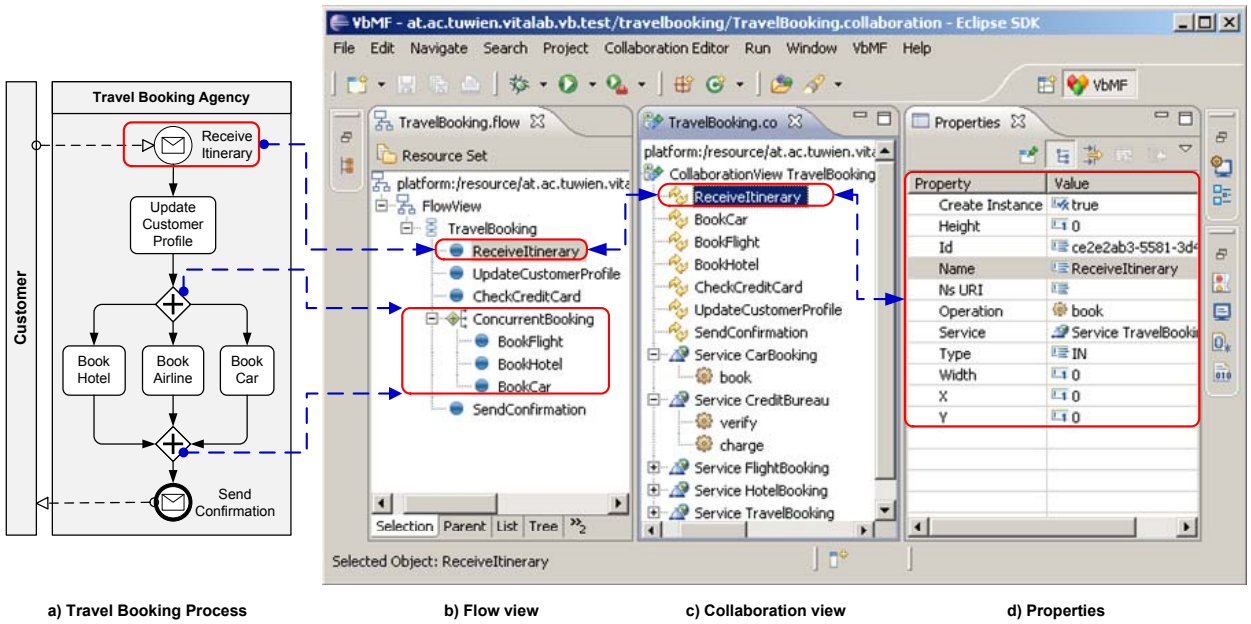

Figure 5: Modeling business process with VbMF: A Travel Booking process example

views of the process concerns. For instance, the task *ReceiveItinerary* waits for the customer's request, and therefore, is described in the Travel Booking Collaboration view (see Figure 5(c) and (d)). Note that the Flow view has been implemented based on the tree editors of Eclipse Modeling Framework[2] for illustration purpose. Nevertheless, it offers basic control flow structures such as sequential and concurrent executions, exclusive choices, and so on, that exist in most of existing process modeling languages such as BPMN, BPEL, etc [38]. As a result, a more friendly, graphical representation of the Flow view similar to the diagram shown in Figure 5(a) can also be achieved with reasonable effort.

3.3. DSLs for compliance concerns

In order to offer expressive and appropriate languages for the different stakeholders, our approach proposes a separation of DSLs into multiple sub-languages, where each sub-language can be tailored for the appropriate stakeholders [54–58]. As such, our approach is able to target different levels of abstraction where each level is tailored for the designated stakeholders. The number of different levels of abstraction depends on the problem domain as well as on the expertise and interests of the stakeholders. As shown in Figure 3, these DSLs shall derive or reference to the basic concepts of the Core model. The name-based matching view integration mechanism shall be used to integrate the information specified in the DSLs and view models.

The Travel Booking process presented in the subsequent sections exemplifies (1) how compliance concerns – specifically QoS compliance concerns – can be integrated into VbMF, (2) how the low-level DSL extends the high-level DSL with the additionally needed technical aspects for expressing QoS compliance concerns of business processes, and (3) how domain and technical experts can use the high-level and low-level DSLs, respectively. The QoS compliance concern is targeted in this section as its realizations involve the whole life cycle of a business process from design time to runtime. Therefore, it is reasonable for conveying the use of DSLs to specify and incorporate compliance concerns to process-driven SOAs. Nevertheless, other DSLs can be developed and integrated with VbMF in the same manner. For this reason, we shall not describe every detail of the compliance DSLs but rather emphasize the integration of those with VbMF. Detailed specifications of the QoS DSLs can be found in [54–58], the descriptions of the licensing DSLs are proposed in [42], and specifications of the security DSL are provided in [43].

[2] `http://www.eclipse.org/modeling/emf`

3.3.1. The high-level QoS DSL

The fundamental purpose of the high-level QoS DSL is to enable domain experts specify which QoS compliance concerns have to be measured with regard to Service-Level Agreement (SLA) requirements for a particular business process or service. The DSL also enables specification of actions to be taken whenever SLAs are violated. As a result, the high-level DSL shall provide expressive notations for representing concepts and terminologies of the QoS and SLA domains.

The scope of the high-level QoS DSL covers the annotation of services and processes with QoS measurements. Each QoS measurement is defined in an SLA between the service provider and the service consumer. In this work we leverage the definitions of runtime- and performance-related QoS measurements proposed in [59]. An example of a high-level specification of the QoS compliance concerns is: "*If the availability of a service is less than* 99*%, an e-mail must be sent to the system administrator*".

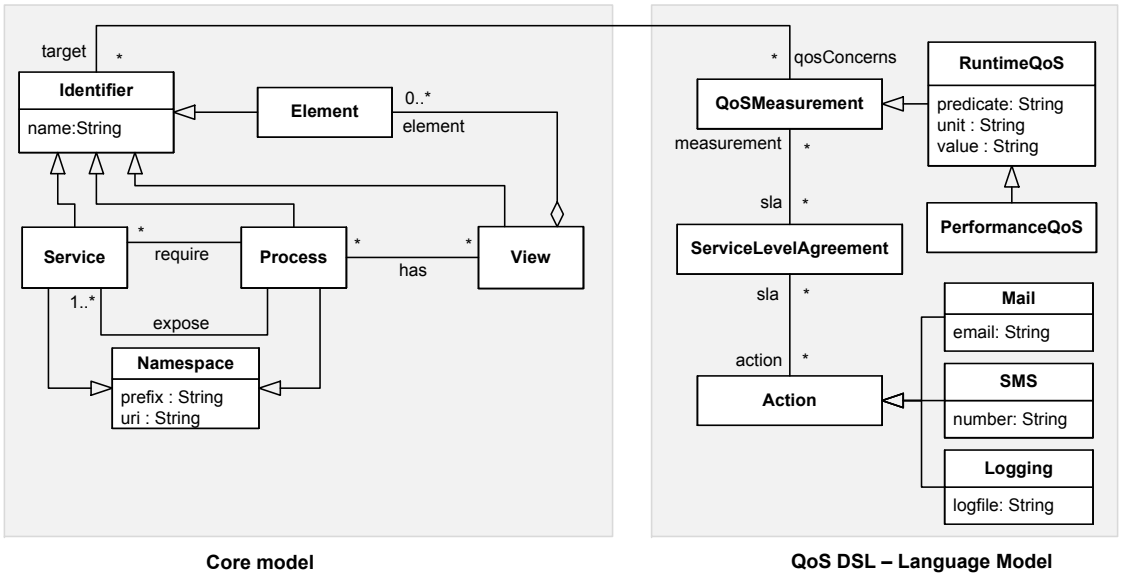

Figure 6: An example of annotating processes or services with QoS compliance concerns

The language model of the high-level QoS DSL is shown in Figure 6. The elements of this language model extend the aforementioned VbMF Core model to enable representation of relevant QoS compliance concerns. In particular, the *Identifiers* – specifically, *Service* and *Process* elements – of the VbMF Core model can be targets of a *QoSMeasurement*. This language model provides the possibilities for specifying *PerformanceQoS* and *RuntimeQoS* related QoS measurements. Each QoS measurement has relations to contractually negotiated *ServiceLevelAgreements* which might be associated with a number of *Action*s. Our QoS DSL also enables the stakeholders to define appropriate actions that are taken if a certain SLA is violated. In Figure 6, we present essential actions such as *Mail* – for sending notification emails, *SMS* – for sending notification SMS messages, and *Logging* – for recording execution events to event logs. One can extend the high-level QoS DSL by defining sub-classes of the *Action* element to specify different other types of actions such as inspection, recovery, adaptations, and so on [60]. Additionally, we devise OCL-like formal constraints for ensuring the consistency of high-level QoS models. For illustration purpose, an excerpt of these constraints is also shown in Listing 1. These constraints have been implemented in our approach using the Check language provided in the Eclipse Model to Text (M2T) project.[3]. Note that the QoS DSL shall be related to process-driven SOA elements via VbMF view models by the name-based matching view integration mechanism. The aforementioned constraints and further constraints can also be leveraged during this stage to verify the consistency of view models and DSLs.

[3] `http://www.eclipse.org/modeling/m2t`

```
context QoSMeasurement
    inv: self.target <> null implies
    (target.oclIsKindOf(core::Service) or target.oclIsKindOf(core::Process)) and sla->notEmpty())
context RuntimeQoS
    inv: self.predicate <> null and self.unit <> null and self.value <> null
context Mail
    inv: self.email <> null
context SMS
    inv: self.number <> null
context Logging
    inv: self.logfile <> null
```

Listing 1: An excerpt of the OCL constraints for the high-level QoS DSL language model

To illustrate the use of the high-level QoS DSL, let us assume that the task *ReceiveItinerary* of the Travel Booking process example in Figure 5 must satisfy a latency of less than 4 days and an availability of more than 99%. We create new instances of *PerformanceQoS* and *RuntimeQoS* in order to describe the aforementioned requirements and associate them with the *ReceiveItinerary* task of the Travel Booking process (see Figure 7).

```
## define a required latency
PerformanceQoS create ItineraryLatency -superclasses Latency \
      -predicate LESSTHAN -value 4 -unit DAYS

## define a required availability
RuntimeQoS create ItineraryAvailability -superclasses Availability \
      -predicate GREATERTHAN -value 99 -unit PERCENT

## assign the QoS compliance concerns to the service
ReceiveItinerary qosConcerns {ItineraryLatency IniteraryAvailability}
```

Figure 7: Using concepts of the high-level QoS DSL to represent QoS requirements

3.3.2. The low-level QoS DSL

The aforementioned high-level QoS DSL is mainly used by the domain experts to express domain concepts or to communicate with customers or technical experts in the requirement analysis stage. Beyond requirements analysis, technical users like developers and system administrators need to augment specifications in the high-level DSL with technology-specific details, in order to support code generation for implementing technologies. For instance, the *ReceiveItinerary* task shall be realized by using a Web service invocation. Therefore, the QoS requirements specified in Figure 7 shall be measured in the underlying Web service framework. The low-level QoS DSL in this case extends the high-level QoS language model with technology-specific aspects. Note that the constructs and expressions of the low-level DSL are extracted from the concepts of the underlying technologies that the technical experts understand and can populate with necessary technical information. In Figure 8, we extend the high-level QoS DSL with technology-specific details for the open-source Apache CXF Web service framework[4].

The language model of the low-level QoS DSL.

The language model of the low-level QoS DSL shown in Figure 8 represents the technical details regarding the operation of the underlying Web service framework. The message-flows between the service client and the service provider are based on *Chain*s. Each service client and service provider has two chains. An incoming chain is responsible for incoming messages, and an outgoing chain is responsible for outgoing messages. Each chain – incoming or outgoing – consists of a number of *Phase*s, during which the QoS values can be measured. Every phase can contain one or more interceptors which are implemented in Java and are responsible for measuring the QoS values. By specifying QoS measurements and the corresponding phases, interceptors can be generated automatically. Runtime QoS concerns, such as the *ResponseTime*, can be measured within the corresponding phases by these automatically generated interceptors.

[4] `http://cxf.apache.org`

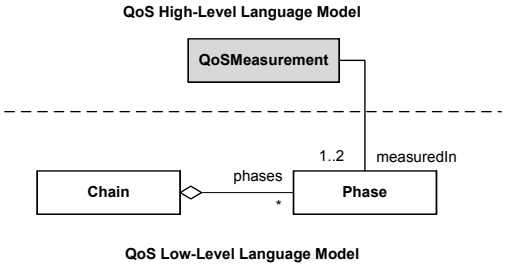

Figure 8: Extending the high-level QoS language model with technology-specific elements

```
# define the phases
OutPhase create OutSetup
OutPhase create OutSetupEnding

## define in which phases the Response Time is measrued
PerformanceQoS create Latency \
      -measuredInPhases {OutSetup OutSetupEnding}
```

Figure 9: Using the low-level QoS DSL to represent concrete QoS measurements

We illustrate how technical experts can use the textual syntax of the low-level DSL to map the high-level domain concepts to corresponding technical aspects (see Figure 9). The *Latency* metric shall be measured between two phases *OutSetup* and *OutSetupEnding*, respectively.

The idea of proposing the two DSLs is, therefore, to separate platform-independent elements from their platform-specific counterparts. This way, we can map elements of the high-level QoS model to several technologies by deriving adequate low-level QoS DSLs. Other Web service technologies can be applied in the same manner to the example we have presented.

3.4. Compliance Metadata model

In this section we present a compliance metadata model which serves as a bridge between the compliance concerns – represented in terms of aforementioned compliance DSLs and organizational functionality specified by VbMF process view models – on the one hand, and compliance sources on the other hand. As described in Section 3.2, the Compliance Metadata model correlates process view models and/or compliance DSLs with compliance metadata such as compliance documents, requirements, risks, and so on. On the one hand, such annotations can be used for facilitating automated compliance *controls*. That is, services or processes are distinguished by annotation to implement and realize automated controls. On the other hand, the metadata serves as an information source for automatically generating reports and documentation of compliance requirements and implementations.

A compliance requirement may directly relate to an organizational unit such as a process, a service, or a business object. Nonetheless compliance requirements not only introduce new but also depict orthogonal concerns to these: although usually related to process-driven SOA elements, they are often pervasive throughout the SOA and express independent concerns. In particular, compliance requirements can be formulated independently until applied to a SOA. As a consequence, compliance requirements can be *reused*, e.g., for different processes or process elements.

The proposed model for expressing compliance metadata is shown in Figure 10. The annotation of specific SOA elements with compliance metadata are achieved using the *Control* element that is associated with concrete compliance implementations such as processes, process tasks, services, or concepts of the DSLs that derive from the *Identifier* element the Core model. The aforementioned association is represented by the relationship "*implements*". Note that this relationship is the basis based on that the name-based matching algorithm (cf. [38, 41, 49, 50]) can be used to correlate and integrate the Compliance Metadata model with other view models and/or DSLs.

A compliance *Control* can contain a number of sub-controls. This way, compliance controls can be grouped and combined. A number of *Controls* might fulfill a certain *ComplianceRequirement* that, in turn, relates to some *ComplianceDocument*s such as *Regulation*s, *Legislation*s, or *InternalPolicies*. Such *RegulatoryDocuments* can be mapped to

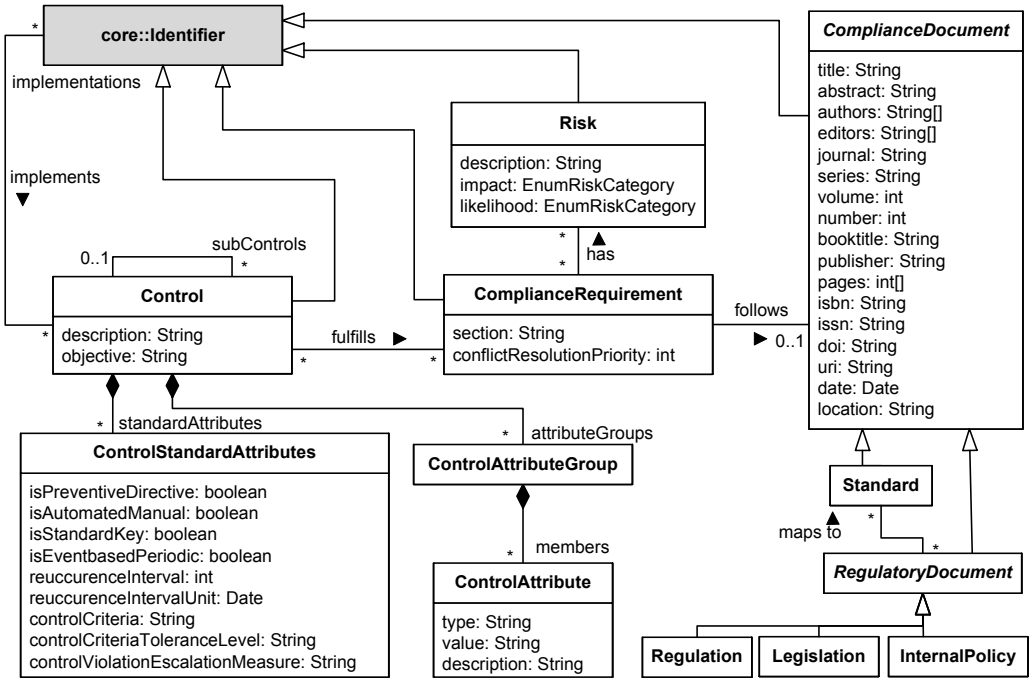

Figure 10: The Compliance Metadata model

Standards that represent another type of *ComplianceDocument*. A compliance requirement often comes with risks that arise due to compliance violations. *Risks* have dimensions such as *likelihood* or *impact*. In this work we provide basic support for specifying such dimensions using linear comparable constants. Of course, these can be refined with more elaborative modeling elements that allow for non-trivial functions and the use of parameters, e.g., for probability density functions. For documentation purposes (see the "*Documentation*" element in Figure 3) and for the implementation of compliance controls, the *ControlStandardAttribute*s help to specify general metadata for compliance controls, e.g., if the control is automated or manual (e.g., using the attribute *isAutomatedManual*). Besides these standard attributes, individual *ControlAttribute*s can be defined for a compliance control within *ControlAttributeGroups*.

Essential OCL-like constraints similar to those shown in Figure 6 are also necessary to ensure the consistency of the Compliance Metadata model. These constraints shall be used in the model validation step that is described in Section 3.5. We note that a certain compliance control can be realized by any elements of a software system, for instance, including a process task, a single service, a large and complex process or composite service, an event log, etc. This generality of the compliance meta-data model is intentional as it is not possible to foresee what kinds of other DSLs will be integrated: These elements might be parts of given view models and DSLs or can also be parts of extensional views and DSLs added later using the same techniques presented in our paper. This characteristic of our approach backed by the view extension and name-based matching mechanisms of VbMF aims at supporting developers on better dealing with changes in regulations and policies. Nevertheless, that potentially leads to the issue of assuring the consistency and soundness of these models and DSLs. This issue is solved in our approach from two perspectives. On the one hand, the DSL approach shall engage compliance and/or domain experts pro-actively to better formulate the problems/requirements in their domain of expertise. On the other hand, we enable developers to specify additional relevant constraints for validating view models and DSLs as shown in the validation and code generation tool chain described in Section 3.5.

One important aspect when implementing compliance for a SOA is that we want to make the relationship of a compliance requirement derived from, e.g., a certain regulation or standard, with the respective annotated SOA element *persistent*. This enables further identification and resolution of SOA elements, compliance controls, regulations, risks and compliance documents, e.g., in the case of a root-cause analysis of a compliance violations.

A model instance of the compliance metadata that contains a directive from the European Union on the protection

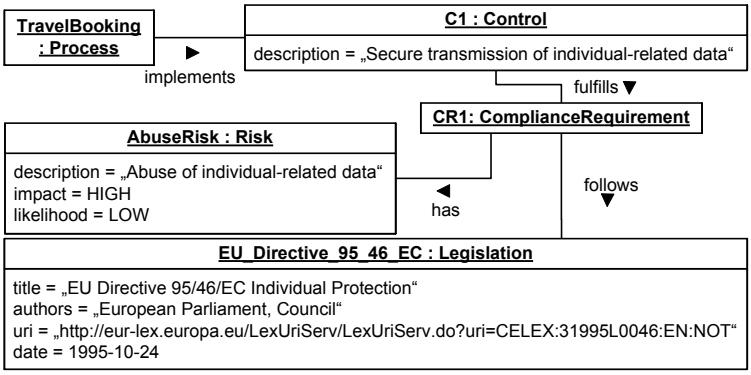

Figure 11: Example for a Compliance Metadata model instance

of individuals with regard to the processing of personal data is given in Figure 11. The *C1* compliance control instance for a secure transmission of personal data annotates process *TravelBooking*. The fulfilled requirement *CR1* follows the legislative document and is associated with an *AbuseRisk*.

With the proposed compliance view, it is possible to specify compliance statements such as *CR1 is a compliance requirement that follows the EU Directive 95/46/EC on Individual Protection*[5] *and is implemented by the Travel Booking process* within the VbMF. Other processes that implement controls for fulfilling the *CR1* requirement can easily be identified. Similarly, for a given legislation the various controls that realize derived compliance requirements can be listed.

3.5. OCL-based model validation and code generation

To ensure the integrity and correctness of the defined model instances, our approach also aims at supporting the integration of different kinds of design time validations. The tool chain shown in Figure 12 illustrates the model validation and code generation phase in our framework. As described in the previous sections, various DSLs exist for creating instances of the compliance concerns' models whilst VbMF can be used to create process models as well as correlated elements of the process models with compliance concerns. Process descriptions created by existing process modeling tools can also be leveraged but at first they need to be adequately mapped into view models [39] in order to take full advantage of the VbMF techniques such as view extension, view integration, code generation, and traceability [38, 41]. Example inputs for the Travel Booking process mentioned above include the process views such as the Flow view, Collaboration view, Information view shown in Figure 5, compliance DSLs such as the high-level and low-level QoS DSL instances shown in Figure 7 and Figure 9, respectively, and the instance of the Compliance Metadata model shown in Figure 11.

The aforementioned instances shall go through the *Model Validator* that is responsible to check whether all static OCL-like constraints hold and whether the models can be properly integrated. The upper box labeled with *Constraints* shows some constraints which were defined using the Check language[6] – an OCL-like language for specifying static model constraints. Following our approach, before the code generation process can start, the defined constraints have to be checked on the model instances.

After the validation step, valid process models and compliance DSLs are handed over to the code generator. The code generator uses transformation templates to transform the model instances into code in executable languages such as BPEL and Java service code. The code generator also generates the compliance documentation based on the Compliance Metadata models. We use the template-based transformation technique provided by the Eclipse Xpand language[7]. Xpand is a powerful typed template language that can be used to generate any kind of textual output. The generated source code can be used for checking the runtime compliance rules or generating reports and

[5] `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT`
[6] `http://www.eclipse.org/modeling/m2t`
[7] `http://www.eclipse.org/modeling/m2t/?project=xpand`

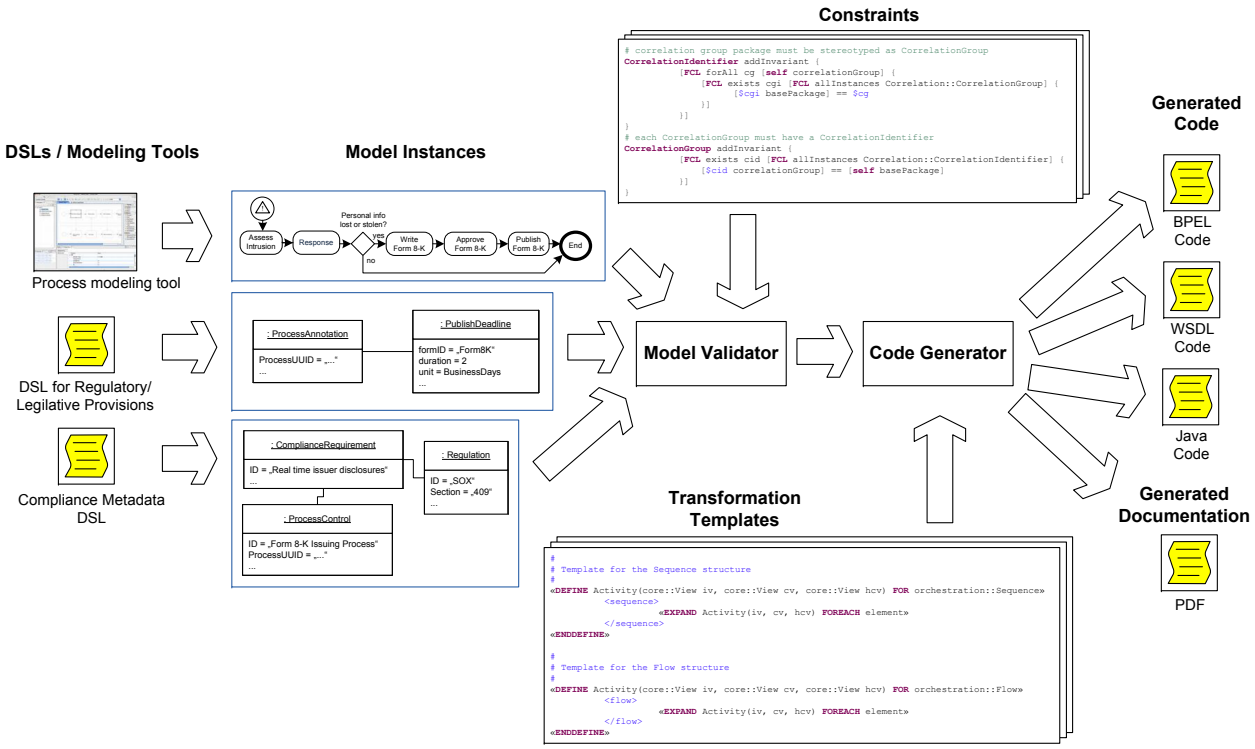

Figure 12: Generation and validation example

documentation. These compliance rules are often encoded in the transformation templates, and therefore, can be reused in other development scenarios.

Proposing a runtime infrastructure for fully deploying, enacting, and monitoring compliance is beyond the scope of this article. Nevertheless, our approach aims at supporting compliance at runtime via model to code transformations that translate the concepts and rules described in compliance DSLs into runtime components, configuration, or directives. These components, configuration, and directives can be deployed in process engines, for instance, Apache ODE BPEL engine[8], application servers, for instance, Apache Tomcat[9], and complex event engine, for instance, Esper[10], to monitor and assess relevant compliance requirements. In the next section, an industrial research case study is presented. The case study was used to test and evaluate the functioning of our approach.

4. Case study

We illustrate the realization of the aforementioned concepts using the CRM Fulfillment process adapted from an industrial case study concerning customer care, billing, and provisioning systems of an Austrian Internet Service Provider. The process is designed using BPMN [46] and implemented using process-driven SOA technology: BPEL [47] and WSDL [61]. BPMN, BPEL, and WSDL are used for exemplification because these are widely adopted in research and industry today. Nevertheless, our approach is not bound to those technologies but is generally applicable for other process-driven SOA technologies. The mapping from and to process-driven SOA modeling languages such as BPMN, BPEL, etc. can be achieved by leveraging the view-based reverse engineering approach in our other works in [39, 62].

[8] http://ode.apache.org
[9] http://tomcat.apache.org/
[10] http://esper.codehaus.org

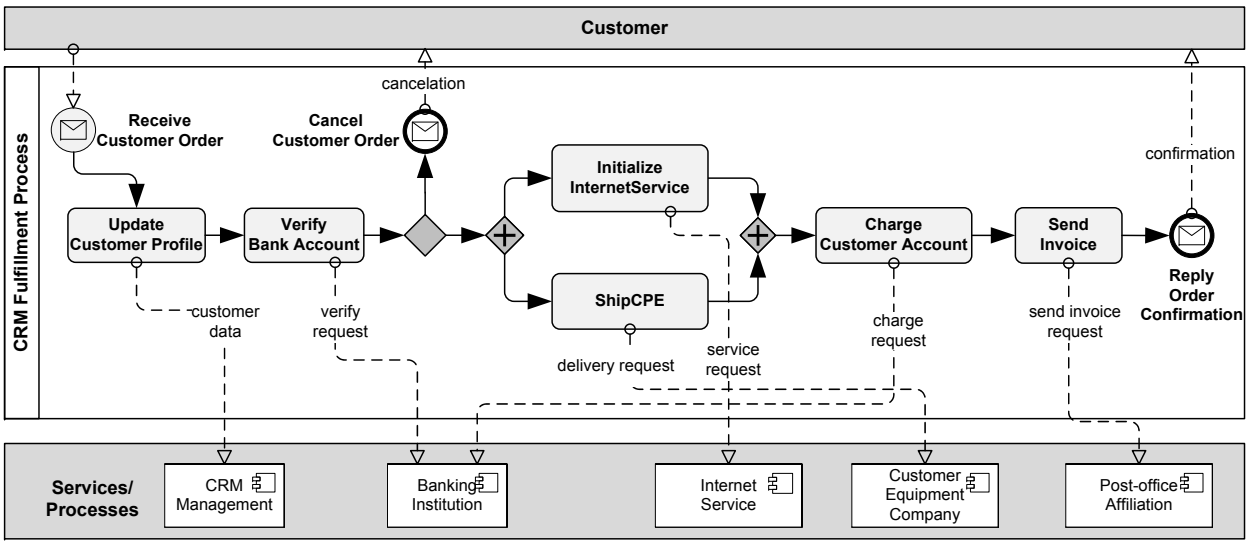

Figure 13: Case study: A CRM Fulfillment process

In the context of the CRM Fulfillment process (see Figure 13), the runtime platform provisions a wide variety of in-house services and external services provided by various partners. For instance, the company has developed in-house services for customer relationship information management and assigning fax numbers, SIP URLs, and mail boxes. Banking partners provide services for verifying the customer account status and charging customer orders. Customer premise equipment (CPE) companies supply services for ordering and shipping home modems or routers. Post-office affiliations are responsible for sending postal invoices to the customers. These services expose their functionalities in terms of WSDL interfaces that can be orchestrated using BPEL processes.

In the subsequent sections, we illustrate the modeling and development of the CRM Fulfillment process and the implementation of compliance requirements step by step in our approach. First, we present the compliance requirements for the CRM Fulfillment process elicited by the collaboration of domain and compliance experts. Next, we develop essential process views including the Flow view, Collaboration view, and the Information view. These views accomplish the process modeling part of our framework. The remainder, which is compliance modeling, follows up with the high-level and low-level DSLs that capture compliance requirements and embody monitoring directives at runtime. The Compliance Metadata model comes to bridge the process views and the compliance DSLs. Finally, the automatic generation of the documentation of the processes and their compliance concerns by using the Compliance Metadata model is demonstrated.

4.1. CRM Fulfillment process

The CRM Fulfillment process is initiated when a customer places an order for Internet services. Customer orders are retrieved via the *ReceiveCustomerOrder* task. The process then invokes the customer relationship management services to update the customer's profile extracted from the order. After that, the banking service is invoked to validate the customer's account status in the *VerifyBankAccount* task. The banking service requires the customer's personal and account data such as the owner's name, billing address, account number, and bank routing code, which are also included in the order. The control after validating the customer's account status is divided into two branches according to the particular status. In case a negative confirmation is issued from the bank service, e.g., because the account number is invalid or the owner and account do not match, the customer will receive an order cancelation response along with an explaining message via the *CancelCustomerOrder* task. Otherwise, the positive confirmation triggers the second branch in which the process continues with two major concurrent tasks to fulfill customer requests: the *InitializeInternetService* task invokes an in-house service, namely, *InternetService*, for initializing the mailbox and for assigning the SIP URL and the fax number, and the *ShipCPE* task calls an external service of the process's partner that asks to independently deliver home router/modem to the customer's shipping address. As fault handling is beyond the

scope of this article, we assume that those activities finish without errors. After all of them have finished, the next task, *ChargeCustomerAccount*, is activated to receive the payment from the customer's account. The *SendInvoice* task will enact the postal service affiliation for sending the customer's invoice to the appropriate address. The process finishes with a confirmation of success to the customer.

4.2. Compliance requirements for the CRM Fulfillment process

Compliance	Risk	Control
Information Security	**R1**: Customer data (resident address, SSN, bank account, etc.) are insecure from potential abuses (Basel II Accord)	**C1**: Communicating channels of customer personal data must be adequately encrypted to ensure privacy
Order Approval	**R2**: Return consignments because of wrong delivery addresses **R3**: Sales to fictitious customers are not prevented and detected	**C2**: Customer's identifications are verified with respect to identification types and information, customer's shipping and billing addresses are checked against some pre-defined constraints (countries, post code, phone number, etc).
Segregation of Duties (SoD)	**R4**: Duties are not adequately segregated (SOX 404)	**C3**: The status of the account verification must be checked and set by a Financial Department staff. The customer's invoice must be checked and signed by a Sales Department staff.
QoS (temporal)	**R5**: The order processing is indefinitely delayed	**C4**: The CRM Fulfillment process must be initiated immediately after receiving a customer order
QoS (latency)	**R6**: The verification of bank account is indefinitely late	**C5**: The verification process must be finished within a certain amount of time
QoS (latency)	**R7**: The customer wants to cancel the order or the ordered goods must be shipped for free	**C6**: The CRM Fulfillment process must finish as soon as possible
QoS (availability)	**R8**: The verification of bank account is not available or heavily loaded	**C7**: The banking service must be checked to ensure a negotiated availability

Table 1: Compliance requirements for the CRM Fulfillment process

From the beginning of the development life cycle, business and domain experts work together with the compliance experts in order to elicit various necessary compliance requirements for the CRM Fulfillment process. The interpretation of laws, regulations, standards, business contracts, etc., given by the business and compliance experts, according to the context of the CRM Fulfillment process, transforms compliance requirements into corresponding controls. Because most of the laws and regulations are very vague and abstract, this kind of transformation is hardly automated, but requires specific and deep juristic knowledge and experience of compliance experts to completely and precisely interpret and formulate the compliance requirements. Table 1 shows an excerpt of compliance requirements used for illustrating our approach in modeling and developing business compliance. In the subsequent sections, we present in detail the steps of business process development along with the modeling of these compliance requirements for the CRM Fulfillment process using our approach in this article.

4.3. Modeling the CRM Fulfillment process

The View-based Modeling Framework [38] supports stakeholders in modeling and developing business processes by using the notion of views to separate the various process concerns. Figure 14 shows the Eclipse-based realization of the View-based Modeling Framework that we use to develop the CRM Fulfillment process. From left to right of the upper row, three basic views, namely, Flow view, Collaboration view, and Information view separately represent the following concerns of the CRM Fulfillment process:

- The Flow view embodies the control flow of the process
- The Collaboration view captures the interactions of the CRM Fulfillment process with other services or processes
- The Information view represents data objects and data processing of the process

The stakeholders, for instance, business analysts, process modelers, or developers, according to their specific needs, skills, and knowledge, can analyze and manipulate the CRM Fulfillment process via each of those individual views or a perspective of interest which combines many, or even all, of those views. Apart from being able to accommodate different perspectives to the stakeholders, VbMF is also able to significantly reduce the complexity of process descriptions by separating tangled process concerns into (semi-)formalized view models [38, 49, 63]. The integration of views can be accomplished by using the name-based matching and view integration mechanisms

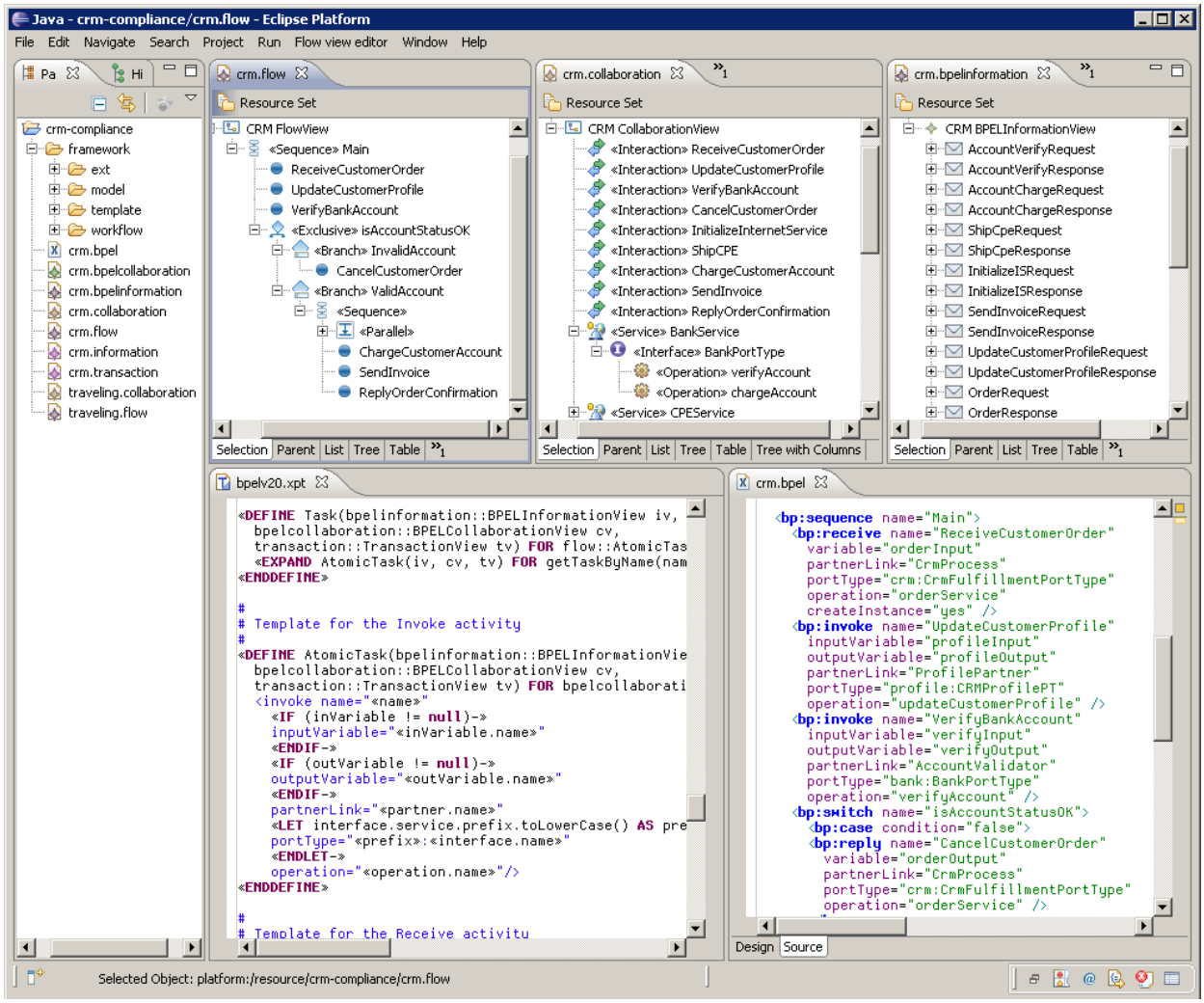

Figure 14: Using VbMF to model the CRM Fulfillment process

developed in VbMF [41]. These process views are then passed to the VbMF code generator in the last step described in Section 3.5 for generating process executable code in BPEL and process service interface in WSDL. In the lower part of Figure 14, we present, from left to right, an excerpt of the templates for the code generation along with an excerpt of the BPEL code of the CRM Fulfillment process generated from the process views. After the business process has been modeled, the appropriate compliance concerns described in terms of DSLs shall be integrated into the business process through the Compliance Metadata model and the name-based matching mechanism [41]. These steps are explained in the following sections.

4.4. Modeling the QoS compliance concerns of the CRM Fulfillment process

Following the examples of Section 3.3, in this section we want to illustrate the use of the high-level and low-level QoS DSLs for annotating the CRM Fulfillment Process with the required QoS compliance concerns as mentioned in Table 1.

4.4.1. Using the high-level QoS DSL

Using the implemented high-level QoS DSL, one can annotate the CRM Fulfillment process with QoS compliance concerns as illustrated in Figure 15. We exemplify the modeling of the two controls *C6* and *C7* described in Table

```
## import the definition of the CRM Fulfillment Process
import CRMFulfillmentProcess

## C6 - the CRM Fulfillment Process must be finished within 3 hours
PerformanceQoS create CustomerOrderLatency -superclasses Latency \
      -predicate LESSTHAN -value 3 -unit HOURS

## assign the modeled response time to the process
CRMFulfillmentProcess qosConcerns {CustomerOrderLatency}

## C7 - The banking service must be available at 99% of the time
RuntimeQoS create BankingServiceAvailability -superclasses Availability \
      -predicate GREATERTHAN -value 99 -unit PERCENT

## assign the modeled availability to the service
CRMFulfillmentProcess::BankingService qosConcerns {BankingServiceAvailability}
```

Figure 15: Specifying the required QoS compliance concerns by using the high-level QoS DSL

1. First, the business expert will *import* the CRM Fulfillment process which was modeled in the VbMF. Now, the QoS compliance concerns can be specified and assigned to the process, its activities, or services. The excerpt of the high-level QoS DSL shown in Figure 15 states that the allowed latency of the CRM Fulfillment process must be less than 3 hours and the availability of the banking service must be greater than 99%.

4.4.2. Using the low-level QoS DSL

In Figure 16, we present how the low-level QoS DSL is applied in the CRM scenario. The technical expert shall specify that two phases, namely, *OutSetup* and *OutSetupEnding* of the Apache CXF Web service framework shall be used to measure the *Latency* element specified in the high-level QoS DSL shown in Figure 15. Note that the *CustomerOrderLatency* element is a sub-class of *Latency*, and therefore, shall be measured by the two phases mentioned above. As we can see, the low-level specification in Figure 16 are similar to the one shown in Figure 9.

```
# define the phases
OutPhase create OutSetup
OutPhase create OutSetupEnding

## define in which phases the Response Time is measrued
PerformanceQoS create Latency \
      -measuredInPhases {OutSetup OutSetupEnding}
```

Figure 16: Specifying the additionally needed technical aspects by using the low-level QoS DSL

Now, we can see the advantage of the separation of high- and low-level languages. The QoS values have to be measured always in the same phases of the Apache CXF Web service framework. Hence, the technical experts have to specify the technological aspects just once. The requirements of the used technology do not change as often as the compliance concerns of process-driven SOAs. That is, the low-level requirements do not change as often as the high-level ones. In case the underlying technology changes, for instance, due to a software update, it is likely that the technical aspects have to be re-modeled to accommodate these changes. By extending the number of QoS measurements, such as by adding scalability and throughput measurements, the main work lies in the extension of the code generator which generates executable code.

After all needed compliance concerns are associated with processes or services, the compliance experts can specify the meta-data of the compliance aspects by using the Compliance Meta-data model. Afterwards, the model validation and code generation phases can start.

4.4.3. Generated code for QoS compliance concerns

The specified QoS compliance concerns have to be measured during runtime. Now, some generated codes are presented which is executed during the runtime of the system to measure the required QoS values. In the used

```
public class BankingServiceLatencyInterceptor1
      extends AbstractPhaseInterceptor<Message> {
      public BankingServiceLatencyInterceptor1() {
            super(Phase.SETUP);
      }
      public void handleMessage(Message msg) throws Fault {
            msg.put("Latency",new Long(System.currentTimeMillis()));
      }
}



public class BankingServiceLatencyInterceptor2
      extends AbstractPhaseInterceptor<Message> {

      public BankingServiceLatencyInterceptor2() {
            super(Phase.SETUP_ENDING);
      }

      public void handleMessage(Message msg) throws Fault {
            if(msg.get("Latency")!=null) {
                  long nMeasuredTime = System.currentTimeMillis() - ((Long)msg.get("Latency")).longValue();

                  /* send the measured response time to the QoS monitor */
                  ...
            } else {
                  throw new Fault(new Exception("Latency not found in message!"));
            }
      }
      ...
}
```

Figure 17: The generated interceptors for measuring the required QoS values

technology, the Apache CXF Web service framework, interceptors can be integrated into the message-flow between service consumer and service provider. In our case, such interceptors shall be used for measuring the required QoS concerns (cf. 3.3.2). Figure 17 illustrates excerpts of the generated interceptors for measuring the required latency of the banking service.

The first interceptor, *BankingServiceLatencyInterceptor1*, is responsible for storing the current timestamp into the header of the message which flows between the service clients and the service providers. The second interceptor, *BankingServiceLatencyInterceptor2*, retrieves the first timestamp of the message's header and compares it with the current timestamp. If there is no timestamp in the header of the message, an exception is thrown. Otherwise, the actual latency of the banking service is determined by the difference between both timestamps. The measured latency is delivered to a monitor component that monitors and stores the measured QoS values during the runtime of the system. The phases, in which the interceptors have to be executed, are specified in their constructors.

After the generation of executable code of the process and its services as well as the interceptors for measuring the required QoS values, the compliance metadata can be specified. The following section shows the specification of the compliance metadata and the generated compliance metadata matrix for reporting and documentation to managers or auditors.

4.5. Compliance metadata: the coalescence of process-driven SOAs and business compliance

So far we have presented the modeling of the CRM Fulfillment process using VbMF and the expressing of concrete compliance concerns using multiple DSLs. Now, we take the final step in which we correlate the process and its services and compliance DSLs with compliance metadata.

We demonstrate an instance of the compliance metadata model in Figure 18. The model instance describes the control *C1* of Table 1. The CRMFulfillment process must implement the control *C1* which fulfills the compliance requirement *CR1*. The compliance requirements *CR1* originate from the *Basel_II Standard* compliance document and have an *AbuseRisk*. The *impact* and *likelihood* of the risk are *HIGH*.

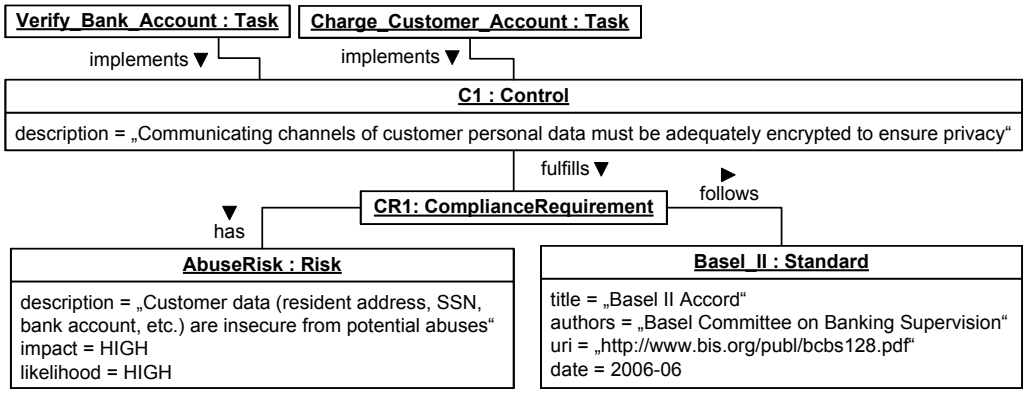

Figure 18: Compliance Metadata view for the Information Security compliance requirement

Finally, let us consider that compliance stakeholders need to react to changes quickly because legislations and policies are subject to change and hence, compliance requirements alter too. Therefore, and because such changes often need to be implemented on time, existing processes need to be adapted. Using our approach, the stakeholders benefit from the separation of concerns, and therefore, only have to formulate the Compliance Metadata model for accommodating a relevant control with the current requirements and compliance concerns. That is, a new requirement is added by the compliance expert by specifying the relation to the corresponding compliance documents and associated risks. Furthermore, the compliance expert substitutes a deprecated requirement within the accordant compliance control. Similarly, the compliance concerns are formulated and associated with the control. As a consequence, it is not necessary to modify the, e.g., control flow, information, or collaboration model of a process when compliance changes occur. Existing metadata and DSLs can be reused for annotating SOA elements and changes to the compliance metadata can be realized in an agile way.

4.5.1. Compliance documentation

The compliance metadata not only serves for specifying the compliance aspects of a process-driven SOA but also can be used for reporting and documentation purposes. In particular, it can be used for generating documentation. Such documentation visualizes compliance relevant information for relevant stakeholders, such as executive managers and auditors, and therefore, help them to quickly gain an overview of a thorough view. Hyperlinks to other documentation pages allow the user to navigate to related information or to request more specific details.

Other generated documentation of the compliance metadata focuses on e.g., the relation of compliance requirements and compliance documents, such as standards or legislative documents. Also, the *coverage* of SOA elements in regard to compliance aspects with their relation to compliance documents can be visualized and highlighted.

In Figure 19, we present an excerpt of the matrix for the CRM Fulfillment process that depicts the relation of different compliance controls with some risks. The corresponding Xpand template is shown in Figure 20. Compliance controls, such as QoS or SoD, are associated with risks of legal sanctions. In contrast, the Information Security compliance control also comes with a loss of customer trust risk. If the availability of the banking service is inadequate, it may result in a loss of total sales.

5. Related work

As mentioned above in the introduction, our previous works in [38, 40, 41, 49, 50] are the foundation of the approach presented in this paper. Besides, Tran et al. [64] briefly summarize the overall achievements of the COMPAS project[11] of which the methods and techniques presented in this paper play a crucial role. This paper targets the creation and integration of compliance metadata, compliance DSLs, and SOA concepts through the view-model-based approach. Oberortner et al. provided detailed specifications of the QoS DSL in [54–58] whilst the specifications

[11] `http://www.compas-ict.eu`

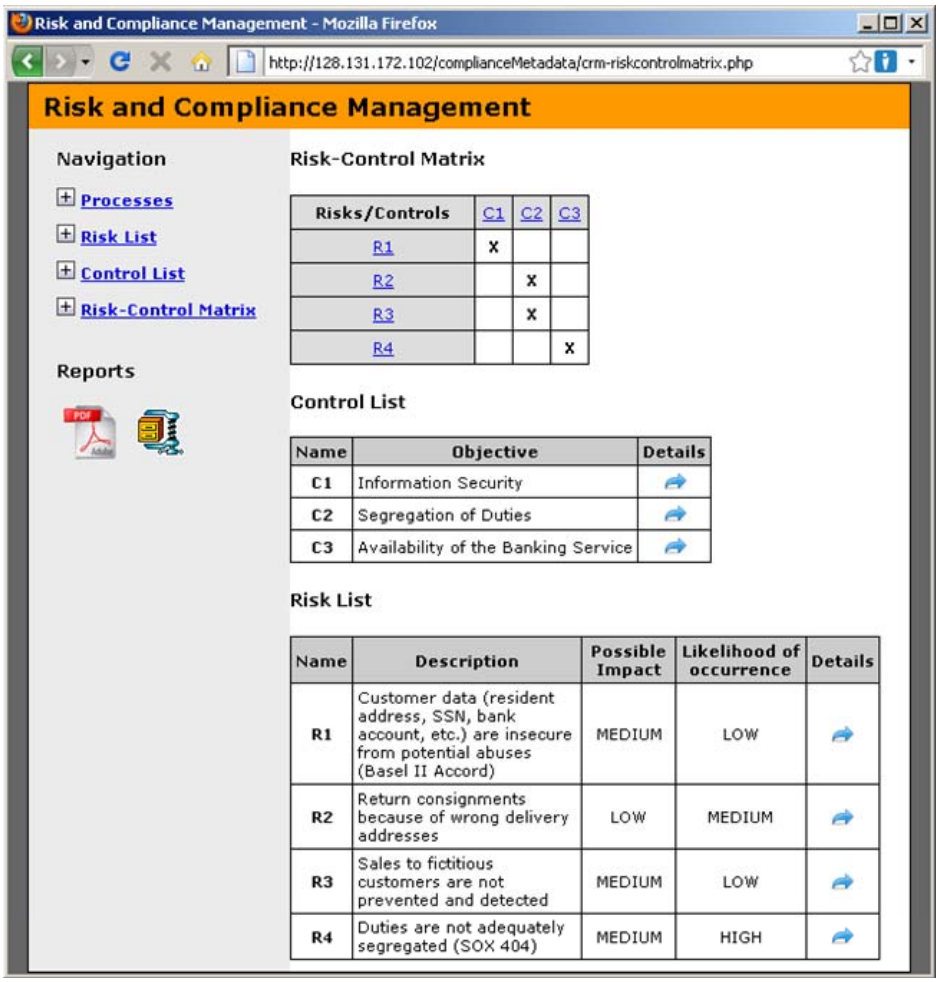

Figure 19: Compliance risk-control matrix

```
<h2>Risk-Control Matrix</h2>
<table>
  <tr>
    <th>Risks/Controls</th>
«FOREACH cv.control AS c»
    <th><a href="«cv.processName+"_C_"+c.uuid+".html"»">«c.name»</a></th>
«ENDFOREACH»
  </tr>
«FOREACH cv.risk AS r»
  <tr>
    <th><a href="«cv.processName+"_R_"+r.uuid+".html"»">«r.name»</a></th>
  «FOREACH cv.control AS c»
    <td>«IF (c.requirements.risks.contains(r))»X«ENDIF»</td>
  «ENDFOREACH»
  </tr>
«ENDFOREACH»
</table>
```

Figure 20: Templates written in Xpand language for generating the compliance risk-control matrix

of the licensing DSL and security DSL are provided in [42] and in [43], respectively. The traceability approach, namely, VbTrace, proposed in [41] can complement to the work in this paper in the sense that VbTrace can enable us to efficiently establish and maintain the dependency links between model artifacts.

Assuring compliance can be broadly categorized into two main strategies: "compliance by design", i.e., implementation of compliance through designing it into a system, and "compliance by detection", i.e., implementation of compliance by observing a system to ensure that its execution was compliant [65]. The different works presented in this section address compliance from either one of or both of these perspectives. Our approach aims at supporting both compliance strategies assuring perspectives in one integrated framework. This is necessary because these two perspectives are not mutual alternatives for fully solving all compliance problems but both approaches can be useful in different design situations. In the subsequent paragraphs, we briefly summarize existing works relate to our approach. Then, we clarify the crucial distinction of our approach to the related work.

Elgammal et al. [34, 66] present a thorough study of existing approaches for formalizing compliance requirements such as Linear Temporal Logic (LTL), Computational Tree Logic (CTL), and Formal Contract language (FCL). Based on that, the authors suggest essential features for a comprehensive formal language aiming at providing powerful expressiveness with reasonable complexity as well as supporting design time verification and runtime monitoring. Arbab et al. proposed a channel-based, formal coordination model, namely, REO, that can be used to model compliance for behavioral models, including business processes represented in, for instance, BPMN and BPEL [8]. The behavior of a business process is mapped to so-called REO circuits (the channel-based coordination models). Other types of compliance concerns than behavioral models cannot be supported. This approach can be extended by mapping other behavioral models to REO circuits with additional efforts. Ghose and Koliadis [14] present an approach for ensuring compliance in which process tasks, such as atomic tasks, loops, and compensations, and sub-processes are annotated with (in)formal annotations. Compliance violations can be detected by an exhaustive path exploration algorithm. Ly et al. [11, 12] report a semantic-based compliance verification approach in which compliance requirements are transformed into mutual exclusion constraints and dependency constraints. These constraints are used to verify the semantic correctness of process models, process instances, and process evolutions. Another compliance validation approach introduced by Namiri et al. [13] is based on the assumption that process models are by default not compliant. These non-compliant processes are enriched with controls by the compliance experts and executed with the support of a monitoring infrastructure and a knowledge base of controls and process models to detect compliance violation at runtime. The disadvantage of this approach is that the compliance rules that are intrusively embedded in process descriptions can be unintentionally broken by developers due to their unawareness of those rules. The separation of compliance concerns and process models in our approach can help the developers avoid this issue. Awad et al. [15] introduce an automated approach for checking compliance of business process models based on a visual query language BPMN-Q to describe compliance rules and model checking to assure compliance requirements are fulfilled. Schumm et al. [33] propose the notion of process fragments and techniques for extraction, integration, and visualization of such fragments for modeling compliance in business processes. Kabicher et al. [28] propose an activity-oriented clustering technique for managing the dependencies between process models and compliance rules and optimizing compliance and consistency checking with regard to a considerably large amount of business processes. Schleicher et al. [67] target the compliance concerns of business processes in cloud-based systems but merely focus on the data-sovereignty issues. Weigand et al. [31] propose an approach to business policy compliance in SOA-based systems that addresses a clear separation of business level and execution level in order to achieve more flexibility and adaptability. In [32], Schleicher et al. introduce the concept of *compliance scope*s that are the area where certain compliance conditions must hold. The main objective is to better support human business process designers at design time.

Liu et al. [9] propose a framework for static checking of business processes in which BPEL processes are transformed into Finite State Machine (FSM) whilst the compliance requirements are translated into LTL. Static compliance checking is accomplished by the model-checker NuSMV2 [68] that validates FSM and LTL expressions. This framework merely supports checking behavioral compliant requirements at design time. Lotz et al. propose a compliance framework within the scope of the EU project MASTER [10] to map abstract controls to concrete control structures and processes, enforce the controls in business operations, and evaluate the effectiveness of the controls. This approach merely focuses on security related control objectives, i.e., those controls that are leveraged for protecting assets.

The conformity of processes with business contracts which are legal document and important sources of compliance requirements are exploited in [16, 17]. Both approaches use the same formal representation, namely, Formal

Contract Language (FCL), for expressing the compliance constraints derived from business contracts. In [17], the authors introduce the *Ideal semantics* to indicate the compliance degree ranging from no violation, some repairable violations, non-repairable violations, and irrelevant. Execution paths of business processes represented in an event-oriented language are checked against contract conditions, described in FCL, to determine the compatibility of business contracts and the business processes fulfilling the contracts. The approach in [16] goes further by translating the FCL representation of a business contract into BPMN-based abstract processes that consists of different parties involving the contract and the message exchanges between the parties or private processes that specifies the internal business logic of a certain party. The contract specifies legal constraints between parties, and therefore, embodies compliance requirements that the (as-is or to-be) business process implementing the contract have to satisfy (i.e., WHAT) rather than the actual business logic of the process (i.e., HOW). As a consequence, the processes translated from business contracts are far from being executable.

The compliance aware business process design framework proposed by Sadiq et al. [18–22] supports stakeholders at design time wherein processes, represented in graph based models, are annotated with control tags derived from FCL expressions of compliance requirements. These control tags represent control objectives and relevant internal controls that enable the visualization of the controls attached to a particular process as well as support an analysis tool that generates a quantitative measure of deviation of a process to certain compliance rules [17]. The framework described in [69, 70] supports stakeholders in representing compliance regulations and laws using a formal policy language, namely, ExPDT, which is extended to enable it to specify and validate the adherence of business processes to compliance regulations. This approach concentrates on automatically assuring enforceable policies. Non-enforceable policy rules are manually handled via log auditing. Giblin et al. [23, 24] propose REALM, a meta-model for the specification of different regulation, and a compliance management framework based on REALM. REALM provides a concept model that captures the concepts and relationships in the regulations, a compliance rule set in a real-time temporal object logic, and a meta-data providing information of the source regulations and validity dates. Compliance policies are then translated into monitoring rules used for runtime monitoring. The transformation from REALM models, i.e., policy rule, into process models is an important and difficult step of the framework but has not been fully described. The support for other compliance requirements, for example, those considered in this article, have not been covered by this approach. Mahoney and Gandhi [29] develop an integrated framework, namely, SCADASiM, for simulating and monitoring regulatory compliance, especially security concern, in near-real-time for SCADA-based systems. The root-cause analysis approach based on property patterns presented in [71] can be used at design time to specify compliance constraints as well as detect compliance violations. Our framework can leverage this work as an important suitable mean for enabling business experts to involve more in describing and analyzing compliance requirements.

These aforementioned approaches mostly focus on the design time. However, some compliance concerns, for instance, QoS latency or availability, can not be verified at design time but runtime. Van der Aalst et al. [26] proposed an approach for checking compliance at runtime wherein an extended LTL is used for formalizing the dynamic properties of the running systems. These properties then can be checked against the events mined from log files. Rozinat et al. [25] presented another approach focusing on conformance checking of processes at runtime. Processes are formalized using Petri-net and an event log is represented by a set of event sequences. Validations are performed to answer whether the real processes behaviors, recorded in the event logs, actually comply with the specified behaviors in the process models. In his dissertation work [27], Accorsi tackles the shortcomings of existing posteriori auditing systems by using a policy language to describe policies and automatically examine selected system log records against the corresponding policy and generate evidence. The examination is supported by a falsification method which retrieves counterexamples of adherence to the policy from the log records in order to refute compliance violations of the corresponding system.

In summary, we discuss the distinct characteristics of our view-based model-driven approach for business compliance with respect to the aforementioned work. *Firstly*, most of these approaches concentrate on control-flow related aspects, which is called behavioral compliance in these approaches. Our approach aims at supporting a wider range of compliance concerns as mentioned in Section 2.2 by adequately using different DSLs which are tailored for particular business and compliance domains.

Secondly, these approaches (except those of [9, 19]) are still very distant from the perception of an important stakeholder: the business analyst (or the compliance expert) due to the lack of suitable and tailorable languages with respect to his/her knowledge and expertise. We address this issue by the separation of high level and low level DSLs

as well as the separation of DSLs into sub-languages which are appropriately tailored for particular stakeholders. Furthermore, the separation of abstraction levels along with the separation of process concerns and compliance concerns enhances the extensibility of our approach into both vertical and horizontal dimensions.

Thirdly, these approaches (except [10]), support business compliance at a certain phase, for instance, either design time or runtime. On the contrary, our approach is a fully integrated approach aiming at supporting stakeholders in achieving compliance by design, statically assessing compliance at design time (cf. Section 4.4) as well as automatically generating of processes, services, monitoring directives, etc., that define rules for runtime checks (cf. Section 4.4.3).

Last but not least, documentation of the implementation of relevant compliance requirements in business processes are crucial evidence for compliance auditing. Moreover, the documentation are important for stakeholders to better understand and analyze processes and the associated compliances. This aspect has not been considered in any of above literatures yet. In our approach, the Compliance Metadata model, which is the bridge between compliance sources and requirements and the realization of compliance in terms of compliance DSLs and process models (see Figure 3), can be used to generate documentation for the processes and relevant business compliance (cf. Section 4.5.1).

Table 2 and 3 summarize these distinctions in details through a qualitative comparison of the state-of-the art and our approach.

	Support for compliance by design	**Support for compliance by detection at design time**	**Support for compliance by detection at runtime**
Awad et al. [15]	Not supported	External model checking validates queried process models against compliance constraints and PLTL expressions	Not supported
ExPDT [69, 70]	Compliance concerns are interpreted by domain experts and partially translated into policy rules which are inputs for automatic validation	Potentially but not mentioned	Policy rules derived from compliance requirements can be inputs for runtime checking doned by other works, e.g., REALM [23]
Contract compliance [16, 17]	Business contracts are translated into formal representations (deontic logic and FCL), then mapped to event-based BPMN processes	Model-checking via formal representations of business contracts represented by FCL expressions and business processes mapped into event-oriented languages	Not supported
Ghose et al. [14]	Not supported	Exhaustive path explorations for detecting compliance violations on the BPMN models annotated with *effect annotations* in formal languages, e.g., FCL or informal, e.g., Controlled Natural Languages(CNLs)	Not supported
Liu et al. [9]	Not supported	Process models in BPELs are mapped into Pi-calculus, then, into Finite State Machine, and checked against compliance rules being represented in Business Property Specification Language (BPSL) and translated into Linear Temporal Logic (LTL)	Not supported
Ly et al. [11, 12]	Not supported	Process models are verified against semantic constraints for their correctnesses	Not supported
MASTER [10]	Introduce a full life cycle for modeling, assessment, monitoring, etc., for security related compliance concerns	Compliance detection at design time is performed within the assessment infrastructure and/or with the feedback from the online enforcement infrastructure	The observation layer that includes monitoring infrastructure on top of an event-based signaling infrastructure for collecting and processing events generated by underlying services
Namiri et al. [13]	Compliance experts add control patterns to the process models to make processes compliant	Not supported	Unintentional removing of controls in the annotated processes by process developers can be detected at runtime. Events emitting during the execution of annotated processes are monitored and validated in the SemanticMirror detecting violations and firing relevant recovery actions
REALM [23, 24]	Not supported	Not supported	Regulations are interpreted and translated into REALM policy rules and relevant correlation rules. The runtime monitoring is enacted by IBM Active Correlation Technology.
REO [8]	Modeling compliance in BPMN process models and mapping them to REO circuits.	Model verification of the REO circuit via constraint automata and other formalisms	Not supported.
Run-time validation approaches [25–27]	Not supported	Not supported	Process models formalized and verified against the events producing during process executions to detect the non-compliant behaviors
SCADASiM [29]	Not supported	Not supported	Event-based monitoring capabilities support near real-time monitoring of security compliance concerns.

(Continued on next page)

	Support for compliance by design	Support for compliance by detection at design time	Support for compliance by detection at run-time
Sadiq et al. [18–22]		Regulatory compliances are described using FCL rules whilst process models are graph-based representations wherein each node has semantic annotations. These formalizations are compared to measure the deviations of compliance that lead to the reparation of process models	Not supported
SoaML [72]	Support for defining service contracts, such as QoS agreements	Not supported	Not supported
VbMF	Supported through view-based models in various compliance concerns.	Support via model validation.	Supported via code generation of processes, services, monitoring directives, etc. that define rules for runtime checks.
Weigand et al. [31]	Partially supported by transforming formal business rules to executable descriptions	Partially supported by formal model checking	Not supported

Table 2: Comparing compliance solutions

	Supported compliance concerns	Extensibility options	Support for involving domain experts	Documentation of compliance
Awad et al. [15]	Mainly support the control flow related compliance rules	Not supported	The query language based on BPMN is intuitive for domain experts	Not supported
Contract compliance [16, 17]	Formal languages for representing compliances (FCL, deontic) solely cover behavioral compliance concerns	Based on FCL and deontic logic, this approach hardly support other compliance concerns which are not mappable to deontic logic and FCL, for instance, temporal and licensing requirements, etc.	Formal languages used in this approach have friendly syntax and semantics for domain experts whilst process models are supposed to be BPMN alike	Not supported
ExPDT [69, 70]	Supported privacy related concerns	Not supported	ExPDT is the formal language aiming at supporting domain experts in translating compliance requirements into policy rules	Potentially supported, especially on privacy based compliance concerns, but not mentioned
Ghose et al. [14]	Control flow-based concerns akin to BPMN process models which are then mapped to Semantic Process Networks (SPNets)	(1) Merely focuses on the control flow; (2) Mapping of compliance requirements from CNLs to *effect annotations* leads to the fact that only the compliance concerns which are able to be described in CNLs akin are supported.	(1) High-level process models, such as BPMN diagrams alike, are supported; (2) Compliance requirements are encoded using CNLs which are close to domain experts	Not supported
Liu et al. [9]	Mainly dealing with the compliant requirements which can be described by temporal logics. Other concerns, such as those considering in this article, are not mentioned	There is no support for the other formalisms which are different from those using in this approach, i.e., Pi-calulus, FSM, LTL, and BPSL	BPSL is an intuitive formalism for business experts, but BPEL is much more technology-specific. High level languages, for instance, BPMN, are often hardly leveraged due to the difference of formalisms	Compliance checking reports
Ly et al. [11, 12]	Semantic constraints are used to describe the mutual exclusion and dependency of process tasks	Semantic constraints aren't rich enough for other kinds of compliance concerns, such as obligations, locative, QoS, licensing, etc.	Semantic constraints and graph-based process models are suitable for domain experts	Not supported
MASTER [10]	Solely focus on security related compliance concerns	Not supported	Introduce two level of abstractions: business models for domain experts, and technical models for IT experts	Potentially supported but not mentioned
Namiri et al. [13]	Mainly focus on behavioral compliance concerns that can impact the process execution	Some other concerns exist in the high level control patterns, but are not mentioned how to apply those patterns in business processes	Compliance representations (i.e., control patterns) and the graph-based process model are suitable for compliance experts	Not supported
REALM [23, 24]	REALM is intentionally designed to support the formalization of regulations	Supporting other compliance concerns, such as those mentioned in this article, are not mentioned	REALM provides adequate representations and tool supports for domain experts	REALM potentially provides documentation for regulatory associated with policy rules via the metadata
REO [8]	Behavioral concerns akin to BPMN process models (other process models such as UML activity diagrams can be mapped to REO circuits, too).	REO is extensible with new channels, new import mappings, and additional formal semantics and model checkers.	High-level model such as BPMN models can be mapped to REO.	Not supported.
Run-time validation approaches [25–27]	Mainly focus on the behavioral concerns	Not supported	Process models are represented in high level and intuitive formalisms, e.g., Petri-net	Not supported
SCADASiM [29]	Mainly support security compliance	Not supported	Not supported	Partially supported via an XML-based security compliance specification

(Continued on next page)

	Supported compliance concerns	Extensibility options	Support for involving domain experts	Documentation of compliance
Sadiq et al. [18–22]	Based on FCL, same as [16, 17]	Same as [16, 17]	[19] provides annotated process visualizations for domain experts	Potentially supported but not mentioned
SoaML [72]	Provides modeling of service contracts between service provider and service client, such as requirements, service interactions, QoS agreements, interface and choreography agreements, and commercial agreements	SoaML is implemented as a UML2 profile and hence can be extended easily	domain experts must have UML2 knowledge	Not supported
VbMF	Views for compliance in process models, compliance in services, QoS policies, licenses, regulatory provisions, compliance in data models, security policies	View models are extensible with any new kind of view	Support for high-level/low-level DSLs; reports, visualizations, and documentation can be generated	Compliance Metadata model: reports, visualizations, and documentation can be generated
Weigand et al. [31]	Business policies and business rules	Not supported	Partially supported by separation of business rules and policies from the execution level	Partially supported via the business policies and business rules specifications

Table 3: Comparing compliance solutions (cont'd)

6. Conclusions

There are crucial shortcomings of existing approaches to business compliance for SOAs, for instance, concentrating on control-flow aspects, lacking of suitable and tailorable languages that appropriately target stakeholders with respect to his/her domain knowledge and expertise, addressing business compliance at a certain phase, for instance, either design time or runtime, and not considering the documentation of the implementation of compliance requirements in business processes which are vital evidences for compliance auditing.

We have presented in this article a novel approach and associated architecture for dealing with compliance in process-driven SOAs. Our approach can support stakeholders in a more extensible and flexible way, comparing to the existing works, in dealing with the divergence of multiple compliance sources realized using all possible kinds of automatic controls, including, but not limited to, controls in processes, services, QoS policies, license policies, security policies, and so on. This includes both design time and runtime controls. The control code, as well as the compliance control documentation, can be automatically generated from the models. Due to the generated documentation that is associated with the models, the compliance information cannot get lost during the evolution of the architecture. DSLs and view models can be used to present compliance concerns to each stakeholder in a view that is most appropriate for the stakeholder's current work task.

The view-based, model-driven framework for compliance in SOAs presented in this article lays a solid foundation for *compliance engineering*. Our ongoing work is to complement this framework with an integrated development environment that facilitates collaborative model-driven design with different stakeholders as well as a runtime governance infrastructure that enacts the detection of compliance violations and compliance enforcement according to the monitoring directives generated from compliance DSLs and the Compliance Metadata model. Besides, we also consider to improve the capability of transforming or importing compliance sources specified in standard specification languages.

Acknowledgment

We would like to thank the anonymous reviewers for providing insightful and constructive comments that greatly helped us to improve this article. This work was partially supported by the European Union FP7 project COMPAS (`http://www.compas-ict.eu`), grant no. 215175 and the European Union FP7 project INDENICA (`http://www.indenica.eu`), grant no. 257483.

References

[1] A. Tarantino, Governance, Risk, and Compliance Handbook: Technology, Finance, Environmental, and International Guidance and Best Practices, Wiley, 2008.

[2] Basel Committee on Banking Supervision, Basel II: International Convergence of Capital Measurement and Capital Standards: a Revised Framework, http://www.bis.org/publ/bcbs107.htm, [Accessed 2011/11/01] (Jun. 2004).
[3] IASB, International Financial Reporting Standards (IFRSs), http://www.ifrs.org/IFRSs/IFRs.htm, [Accessed 2011/11/01] (2007).
[4] U.K. Financial Services Authority, Markets in Financial Instruments Directive (MiFID), http://www.fsa.gov.uk/pages/About/What/International/mifid, [Accessed 2011/11/01] (Nov. 2007).
[5] Ministre de l'?conomie, des finances et de l'industrie, Loi de Sécurité Financière (LSF), http://www.senat.fr/leg/pjl02-166.html, [Accessed 2011/11/01] (Aug. 2003).
[6] The Netherlands Corporate Governance Committee, The dutch corporate governance code, http://commissiecorporategovernance.nl/page/downloads/CODE%20DEF%20ENGELS%20COMPLEET%20III.pdf, [Accessed 2011/11/01] (Dec. 2003).
[7] U.S. Congress, Sarbanes-Oxley Act of 2002, http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=107_cong_bills&docid=f:h3763enr.tst.pdf, [Accessed 2011/11/01] (Jan. 2002).
[8] F. Arbab, N. Kokash, S. Meng, Towards using reo for compliance-aware business process modeling., in: Proc. of the Third Intl. Sym. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008), Vol. 17 of CCIS, Springer, 2008, pp. 108–123.
[9] Y. Liu, S. Müller, K. Xu, A static compliance-checking framework for business process models, IBM Syst. J. 46 (2) (2007) 335–361.
[10] V. Lotz, E. Pigout, P. M. Fischer, D. Kossmann, F. Massacci, A. Pretschner, Towards systematic achievement of compliance in service-oriented architectures: The MASTER approach, WIRTSCHAFTSINFORMATIK 50 (5) (2008) 383–391.
[11] L. T. Ly, K. Gser, S. Rinderle-Ma, P. Dadam, Compliance of semantic constraints - A requirements analysis for process management systems, in: 1st Int'l Workshop on Governance, Risk and Compliance - Applications in Information Systems (GRCIS'08), 2008.
[12] L. T. Ly, S. Rinderle, P. Dadam, Integration and verification of semantic constraints in adaptive process management systems, Data Knowl. Eng. 64 (1) (2008) 3–23.
[13] K. Namiri, N. Stojanovic, Pattern-Based Design and Validation of Business Process Compliance, in: Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I, Springer-Verlag, 2007, pp. 59–76.
[14] A. Ghose, G. Koliadis, Auditing business process compliance, in: 5th International Conference on Service-Oriented Computing (ICSOC), Springer-Verlag, 2007, pp. 169–180.
[15] A. Awad, G. Decker, M. Weske, Efficient Compliance Checking Using BPMN-Q and Temporal Logic, in: 6th International Conference on Business Process Management (BPM), Springer-Verlag, 2008, pp. 326–341.
[16] Z. Milosevic, S. W. Sadiq, M. E. Orlowska, Translating business contract into compliant business processes, in: Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, IEEE Computer Society, 2006, pp. 211–220.
[17] G. Governatori, Z. Milosevic, S. W. Sadiq, Compliance checking between business processes and business contracts, in: Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, 2006, pp. 221–232.
[18] R. Lu, S. W. Sadiq, G. Governatori, Compliance aware business process design, in: Proceedings of the 2007 international conference on Business process management, Springer-Verlag, 2008, pp. 120–131.
[19] S. W. Sadiq, G. Governatori, K. Namiri, Modeling control objectives for business process compliance, in: Proceedings of the 5th international conference on Business process management (BPM), Springer-Verlag, 2007, pp. 149–164.
[20] R. Lu, S. W. Sadiq, G. Governatori, Measurement of compliance distance in business processes, IS Management 25 (4) (2008) 344–355.
[21] S. Sadiq, G. Governatori, Managing Regulatory Compliance in Business Processes, Handbook of Business Process Management 2 Edition, Springer, 2010, pp. 159–175.
[22] G. Governatori, J. Hoffmann, S. Sadiq, I. Weber, Detecting regulatory compliance for business process models through semantic annotations, in: BPD-08: 4th Intl. Workshop on Business Process Design, 2008, pp. 5–17.
[23] C. Giblin, S. Müller, B. Pfitzmann, From regulatory policies to event monitoring rules: Towards model-driven compliance automation, Tech. Rep. RZ 3662, IBM Research (2006).
[24] C. Giblin, A. Y. Liu, X. Zhou, Regulations expressed as logical models (REALM), in: A. I. O. S. Press (Ed.), Proc. of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX 2005), 2005, pp. 37–48.
[25] A. Rozinat, W. M. P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Inf. Syst. 33 (1) (2008) 64–95.
[26] W. M. P. van der Aalst, H. T. de Beer, B. F. van Dongen, Process mining and verification of properties: An approach based on temporal logic, in: On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBAS, Springer, 2005, pp. 130–147.
[27] R. Accorsi, Automated counterexample-driven audits of authentic system records, Ph.D. thesis, University of Freiburg, Germany (2008).
[28] S. Kabicher, S. Rinderle-Ma, L. T. Ly, Activity-Oriented Clustering Techniques in Large Process and Compliance Rule Repositories, in: Proc. BPM'11 Workshops, 1st Int. Workshop on Process Model Collections (PMC 2011), Springer, 2011.
[29] W. Mahoney, R. A. Gandhi, An integrated framework for control system simulation and regulatory compliance monitoring, International Journal of Critical Infrastructure Protection 4 (1) (2011) 41–53.
[30] P. Silveira, C. Rodrguez, A. Birukou, F. Casati, F. Daniel, V. D'Andrea, C. Worledge, Z. Taheri, Aiding Compliance Governance in Service-Based Business Processes, Non-Functional Properties for Service-Oriented Systems: Future Directions (NFPSLA-BOOK-2011) Edition, IGI Global, 2011.
[31] H. Weigand, W.-J. van den Heuvel, M. Hiel, Business policy compliance in service-oriented systems, Information Systems 36 (4) (2011) 791–807, selected Papers from the 2nd International Workshop on Similarity Search and Applications SISAP 2009. doi:DOI:10.1016/j.is.2010.12.005.
[32] D. Schleicher, F. Leymann, D. Schumm, M. Weidmann, Compliance scopes: Extending the BPMN 2.0 meta model to specify compliance requirements, in: International Conference on Service-Oriented Computing and Applications (SOCA 2010), 2010, pp. 1–8.
[33] D. Schumm, F. Leymann, A. Streule, Process views to support compliance management in business processes, in: E-Commerce and Web Technologies, 11th International Conference, EC-Web 2010, Bilbao, Spain, September 1-3, 2010. Proceedings, 2010, pp. 131–142.
[34] A. Elgammal, O. Turetken, W.-J. van den Heuvel, M. Papazoglou, On the formal specification of regulatory compliance: a comparative analysis, in: Proceedings of the 2010 international conference on Service-oriented computing (ICSOC), Springer-Verlag, Berlin, Heidelberg,

2011, pp. 27–38.
[35] T. Stahl, M. Völter, Model-Driven Software Development, John Wiley & Sons, 2006.
[36] J. Greenfield, K. Short, S. Cook, S. Kent, Software Factories: Assembling Applications with Patterns, Frameworks, Models & Tools, J. Wiley and Sons Ltd., 2004.
[37] S. Kelly, J. P. Tolvanen, Domain-Specific Modeling: Enabling Full Code Generation, John Wiley & Sons, 2008.
[38] H. Tran, U. Zdun, S. Dustdar, View-based and Model-driven Approach for Reducing the Development Complexity in Process-Driven SOA, in: Intl. Working Conf. on Business Process and Services Computing (BPSC'07), Vol. 116 of LNI, 2007, pp. 105–124.
[39] H. Tran, U. Zdun, S. Dustdar, View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs, in: 10th Intl. Conf. on Software Reuse (ICSR'08), LNCS, Springer, 2008, pp. 233–244.
[40] T. Holmes, H. Tran, U. Zdun, S. Dustdar, Modeling human aspects of business processes – A view-based, model-driven approach, in: Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA), Springer, 2008, pp. 246–261.
[41] H. Tran, U. Zdun, S. Dustdar, Name-based view integration for enhancing the reusability in process-driven SOAs, International Journal of Business Process Integration and Management 5 (3) (2011) 229–239. doi:10.1504/IJBPIM.2011.042527.
[42] G. R. Gangadharan, V. D'Andrea, Managing copyrights and moral rights of service-based software, IEEE Softw. 28 (2011) 48–55.
[43] COMPAS Deliverable D5.4, Reasoning Mechanisms to Support the Identification and the Analysis of Problems Associated with User Requests, http://compas-ict.eu/compas_results/deliverables/m23/D5.4_Reasoning-mechanisms.pdf (Dec. 2009).
[44] C. Hentrich, U. Zdun, Patterns for process-oriented integration in service-oriented architectures, in: Proceedings of 11th European Conference on Pattern Languages of Programs (EuroPLoP 2006), Irsee, Germany, 2006, pp. 1–45.
[45] U. Zdun, S. Dustdar, Model-driven and pattern-based integration of process-driven SOA models, Int. J. of Business Process Integration and Management (IJBPIM) 2 (2) (2007) 109–119.
[46] OMG, Business process model and notation (BPMN) 2.0, http://www.omg.org/spec/BPMN/2.0/PDF (Jan. 2011).
[47] OASIS, Web Services Business Process Execution Language (WSBPEL) v2.0, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf (May 2007).
[48] OMG, Unified modelling language 2.0, http://www.omg.org/spec/UML/2.0 (Jul. 2005).
[49] H. Tran, T. Holmes, U. Zdun, S. Dustdar, Modeling Process-Driven SOAs - a View-Based Approach, in: J. Cardoso, W. M. P. van der Aalst (Eds.), Handbook of Research on Business Process Modeling, IGI Global, 2009, Ch. 2.
URL https://www.infosci-online.com/reference/details.asp?id=33287
[50] H. Tran, U. Zdun, S. Dustdar, Name-based view integration for enhancing the reusability in process-driven SOAs, in: BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers, Vol. 66 of LNBIP, Springer, 2010, pp. 338–349.
[51] H. Tran, U. Zdun, S. Dustdar, VbTrace: using view-based and model-driven development to support traceability in process-driven SOAs, Software & Systems Modeling 10 (1) (2011) 5–29. doi:10.1007/s10270-009-0137-0.
[52] C. Mayr, U. Zdun, S. Dustdar, Model-Driven Integration and Management of Data Access Objects in Process-Driven SOAs, in: Proceedings of the 1st European Conference on Towards a Service-Based Internet: ServiceWave '08, Springer-Verlag, 2008, pp. 62–73.
[53] IBM, Travel Booking Process, http://publib.boulder.ibm.com/bpcsamp/scenarios/travelBooking.html, [Accessed 2011/11/01] (2006).
[54] E. Oberortner, U. Zdun, S. Dustdar, Tailoring a model-driven Quality-of-Service DSL for various stakeholders, in: MISE '09: Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering, Vancouver, BC, Canada, 2009, pp. 20–25.
[55] E. Oberortner, U. Zdun, S. Dustdar, Patterns for Measuring Performance-Related QoS Properties in Distributed Systems, in: 17th Conference on Pattern Languages of Programs (PLOP), Nevada, USA, 2010.
[56] E. Oberortner, U. Zdun, S. Dustdar, A. B. Cavalcante, M. Tluczek, Supporting the Evolution of Model-driven Service-Oriented Systems: A Case Study on QoS-aware Process-driven SOAs, in: IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010), Perth, Australia, 2010, pp. 1–4.
[57] E. Oberortner, S. Sobernig, U. Zdun, S. Dustdar, Monitoring of performance-related qoS properties in service-oriented systems: A pattern-based architectural decision model, in: Proceedings of the 16th European Conference on Pattern Languages of Programs (EuroPLoP), Irsee, Germany, 2011.
[58] E. Oberortner, D. Damian, Towards Patterns to Enhance the Communication in Distributed Software Development Environments, in: 18th Conference on Pattern Languages of Programs (PLOP), Portland, OR, USA, 2011.
[59] S. Ran, A model for web services discovery with QoS, SIGecom Exch. 4 (1) (2003) 1–10.
[60] L. Baresi, S. Guinea, Self-supervising BPEL processes, IEEE Trans. Softw. Eng. 37 (2011) 247–263. doi:http://dx.doi.org/10.1109/TSE.2010.37.
[61] W3C, Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl (May 2007).
[62] H. Tran, U. Zdun, S. Dustdar, View-based Integration of Process-driven SOA Models At Various Abstraction Levels, in: Int'l Workshop on Model-Based Software and Data Integration (MBSDI), Springer CCIS, Berlin, Germany, 2008.
URL http://cis.cs.tu-berlin.de/Forschung/Projekte/bizycle/mbsdi2008/
[63] H. Tran, T. Holmes, U. Zdun, S. Dustdar, Using Model-Driven Views and Trace Links to Relate Requirements and Architecture : A Case Study, in: J. Grundy, J. G. Hall, P. Avgeriou, Patricia Lago, I. Mistrik (Eds.), Relating software requirements and architectures, Springer, 2011, Ch. 14, pp. 233–256.
[64] H. Tran, T. Holmes, E. Oberortner, E. Mulo, A. B. Cavalcante, J. Serafinski, M. Tluczek, A. Birukou, F. Daniel, P. Silveira, U. Zdun, S. Dustdar, An end-to-end framework for business compliance in process-driven SOAs, in: International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE Computer Society, 2010, pp. 407–414. doi:http://doi.ieeecomputersociety.org/10.1109/SYNASC.2010.52.
[65] S. Sackmann, M. Kahmer, M. Gilliot, L. Lowis, A classification model for automating compliance, in: Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, IEEE

Computer Society, 2008, pp. 79–86.
[66] A. Elgammal, O. Türetken, W.-J. van den Heuvel, M. P. Papazoglou, On the formal specification of regulatory compliance: A comparative analysis, in: ICSOC Workshops, 2010, pp. 27–38.
[67] D. Schleicher, C. Fehling, S. Grohe, F. Leymann, A. Nowak, P. Schneider, D. Schumm, Compliance domains: A means to model data-restrictions in cloud environments, in: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2011), 2011, pp. 257–266.
[68] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV 2: An opensource tool for symbolic model checking, in: 14th Intl. Conf. Computer Aided Verification (CAV'02), Springer, 2002, pp. 241–268.
[69] M. Kähmer, M. Gilliot, G. Muller, Automating Privacy Compliance with ExPDT, in: Proceedings of the 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008, pp. 87–94.
[70] S. Sackmann, M. Kähmer, ExPDT: A Policy-based Approach for Automating Compliance, WIRTSCHAFTSINFORMATIK 50 (5) (2008) 366–374.
[71] A. Elgammal, O. Türetken, W.-J. van den Heuvel, M. P. Papazoglou, Root-cause analysis of design-time compliance violations on the basis of property patterns, in: Service-Oriented Computing - ICSOC 2010 International Workshops, PAASC, WESOA, SEE, and SOC-LOG, Revised Selected Papers, 2010, pp. 17–31.
[72] OMG, Service-Oriented Architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services, Tech. rep., OMG (2008).