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

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

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

Abstract

Compliance in service-oriented architectures (SOA) means in general complying with laws and regulations that apply to a distributed software system. Unfortunately, the divergence and frequent changes of different compliance sources make it hard to systematically and quickly accommodate new compliance requirements due to the lack of an adequate methodology for system and compliance engineering. Moreover, the difference of perception and expertise of multiple stakeholders involving in system and compliance engineering further complicates the analyzing, interpreting, implementing, and assessing of compliance. These two issues lead in the long run to problems regarding complexity, understandability, maintainability, changeability, and reusability of compliance concerns in a SOA. In this article, a model-driven and view-based approach for addressing these problems will be presented. Various compliance concerns can be modeled in separate view models. This is done using domain-specific languages that enable the non-technical experts as well as the technical experts to model in each case only the relevant excerpt of the SOA. The compliance implementations, reports, and documentations are generated from the models and hence become traceable, changeable, understandable, and reusable – independently from the rest of the SOA. Our approach also enables and supports the model-based static checking of compliance at design time and model-based compliance monitoring at runtime.

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 comply with multiple laws, regulations, and business policies (from now on called compliance sources). Compliance is a major issue in many organizations because any compliance violation will 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 [4], the International Financial Reporting Standards (IFRS) [14], the Markets in Financial Instruments Directive (MiFID) [45], the French financial security law (LSF) [27], Tabaksblat [42], or the Sarbanes-Oxley Act (SOX) [46], 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 hardly 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, which is undesirable because the

Email addresses: {uwe.zdun|huy.tran}@univie.ac.at (Uwe Zdun, Huy Tran), {tholmes|e.oberortner|e.mulo|dustdar}@infosys.tuwien.ac.at (Ta'id Holmes, Ernst Oberortner, Emmanuel Mulo, Schahram Dustdar)

resulting solutions are hard to maintain, hard to evolve or change, hard to reuse, and hard to understand. Moreover, this also 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 involve 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 [46]. Unfortunately, each stakeholder group has different interests, knowledge, and expertise than the other stakeholder groups, and often the 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 of the specialization areas of the IT experts who involve in implementing, deploying, enacting, and maintaining organizational software, systems, and compliance. In addition, from the managers' or auditors' perspectives, adequate timely reports and documentations 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 [1–3, 6–10, 19–24, 26, 28, 36, 39, 40, 47]; discussed in detail in Section 5).

In this article we propose a model-driven development (MDD) approach [11, 18, 41] for overcoming these issues. *First*, 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 [18]. To support the involvement of non-technical stakeholders (like business and compliance experts) and technical stakeholders (like software engineers and system administrators) into 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, on the one hand, we can provide different stakeholders with appropriate representations to formulate the domain problem (i.e., compliance concerns) according to their particular expertise. On the other hand, the representations of compliance concerns in terms of DSLs can be independently developed rather than polluting existing business functionality.

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. have developed a View-based Modeling Framework (VbMF) – a specialization of the MDD paradigm – that provides a number of view models for specifying an SOA [13, 43, 44]. In addition, this approach offers a number of mechanisms to enable the integration of different view models as well as generate executable code out of these models [13, 43, 44]. Thus, we exploit this important capability of VbMF and extend VbMF with a compliance metadata model that serves as a bridge among compliance DSLs designed by business and compliance experts, the compliance requirements and compliance sources, and the business functionality of software and systems. Besides, we devise a number of 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 documentations for auditing and compliance demonstration purposes. Our approach aims at supporting the systematic and automatic 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 exemplify our approach for process-driven SOAs – a particular kind of SOAs utilizing processes to orchestrate services – because enterprises increasingly use 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 tackle all kinds of 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 have been developed for other concerns, such as licensing and security (not presented in detail in this

article).

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 SOA

SOAs are typically realized as layered architectures [12]. Based on a communication layer, which abstracts from platform and communication protocol details, a remoting layer 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.

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.

Figure 1 illustrates a small-scale process-driven SOA. 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.

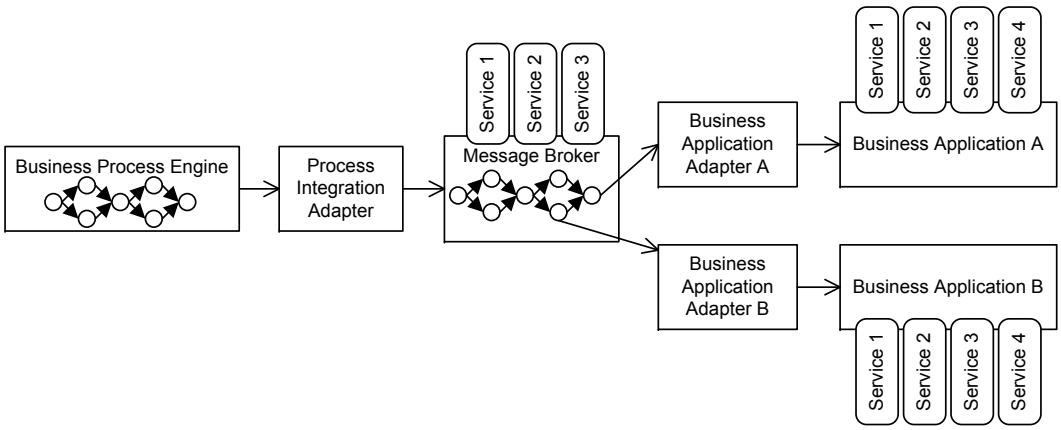


Figure 1: Illustrative small-scale SOA

We limit the scope of our compliance approach to these process-driven SOAs. Next we discuss the kinds of compliance concerns that occur in a process-driven SOA.

2.2. Compliance 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.

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 to an organization to implement any relevant such compliance concern in process-driven SOA in traceable, changeable, understandable, and reusable fashion.

3. Model-driven approach to supporting compliance in SOAs

3.1. Approach overview

Figure 2 gives – on the left hand side – 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 manual controls, reports, or automated controls.

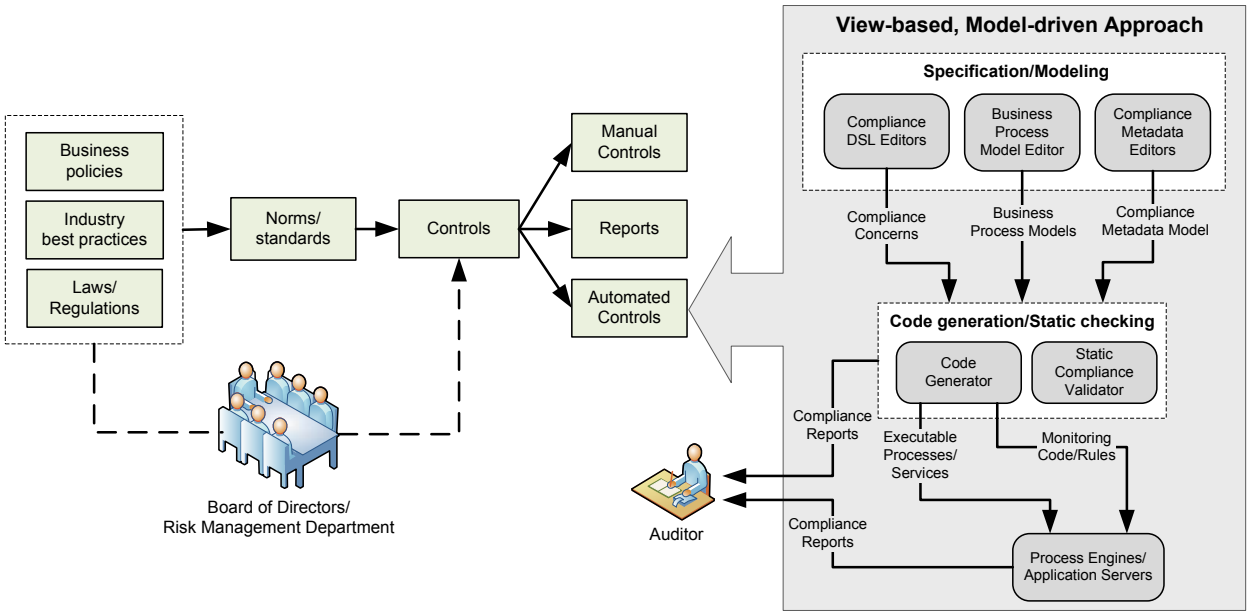


Figure 2: Overview of the view-based, model-driven approach for supporting compliance in SOAs

Our work focuses on *automated controls* in the area of process-driven SOAs (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 automatic 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 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 documentations and reports for auditing and demonstrating purpose are also automatically generated.

3.2. Model-driven development (MDD)

To address the problems of assuring compliance sketched above, we propose to use the model-driven development (MDD) paradigm [11, 18, 41] to enable companies to develop and then evolve and maintain a customized business compliance framework. Domain-specific languages (DSLs) are specification languages which are tailored to be particularly expressive in a certain problem domain. In our approach, DSLs are based on the MDD paradigm to involve

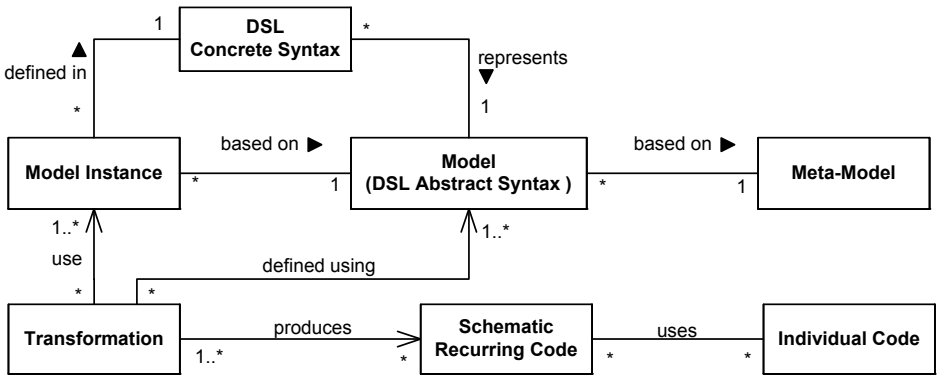


Figure 3: MDD artifacts overview

the different stakeholders into the SOA engineering process. Figure 3 shows an overview of the artifacts in the MDD paradigm.

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.

A DSL describes knowledge via a graphical or textual syntax (called the concrete syntax of the DSL), which is tied to the domain-specific modeling elements through a precisely specified language model (called the abstract syntax of the DSL). The abstract syntax, which represents the language model, defines the elements of the domain and their relationships without considering their notations. That is, the DSL elements are defined in terms of a language model that can be instantiated in concrete model instances. The model instances are defined in the DSL's concrete syntax. The concrete syntax describes the representation of the domain elements and their relationships in a form suitable for the stakeholders using the DSL. Abstract and concrete syntax enable DSL users to define model instances with a familiar notation to represent particular problems of the domain.

An MDD tool introduces some ways to specify transformations. There are different kinds of transformations, such as model-to-model transformations or model-to-code transformations. There are also different ways to specify transformations, such as transformation rules, imperative transformations, or template-based transformations. In any case, the ultimate goal of all transformations in MDD tools is to generate code in executable languages, such as programming languages or process execution languages. The MDD tools are used to generate all those parts of the executable code which are schematic and recurring, and hence can be automated.

3.3. 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 the compliance concerns and service compositions, the compliance framework shall augment business process specifications, such as the Business Process Execution Language (BPEL), 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 [16, 29], BPMN [32], and UML activity diagrams [31]), 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 [13, 43, 44]). 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.

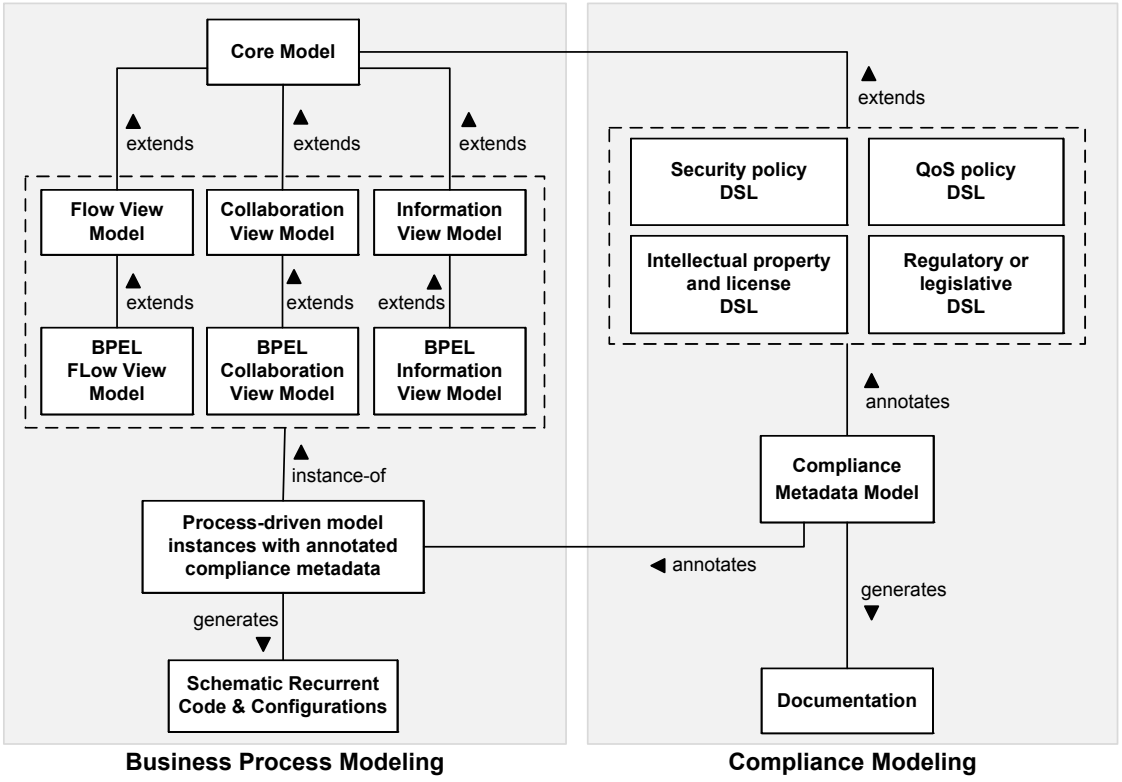


Figure 4: View-based approach for modeling processes and business compliance

As we mentioned in Section 2.2, there are many different kinds of business compliance that companies have to consider, for instance, jurisdictional policies, business policies, QoS policies, intellectual property and licenses, and security policies. 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.

In Figure 4, we illustrate the modeling of process-driven systems and the compliance concerns exemplified in this article using our view-based approach. On the left hand side, the modeling framework provides fundamental view models for describing the functionality of software and systems in terms of business processes. As stated in [43], three view models, the Flow, Collaboration, and Information view models, represent the basic concerns of a business process. Other concerns, such as transactions, human integration, event handling, etc., are also developed and plugged into VbMF accordingly thanks to its extensibility [13, 43]. For the sake of simplicity and concentration on compliance modeling, these other view models are not presented in Figure 4.

On the right-hand side, VbMF provides capabilities for specifying compliance concerns in terms of appropriate DSLs, such as DSLs for representing security policy, QoS policy, intellectual property and licenses, and regulatory or legislative provisions. Each DSL is correspondent to an extension view 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 [43], the framework can be extended to other kinds of compliance concerns. More details on extending the framework with additional compliance DSLs are elaborated in the subsequent sections. In this article, we will illustrate these capabilities of VbMF for one of these DSLs: the DSL for specifying QoS-related compliance concerns.

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 that has a special purpose: To annotate the controls defined both in the basic VbMF views as well as the controls defined in the four extensional

DSLs with compliance metadata. 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. In other words, the Compliance Metadata model allows us to not only model "how" compliance is achieved, but also "why".

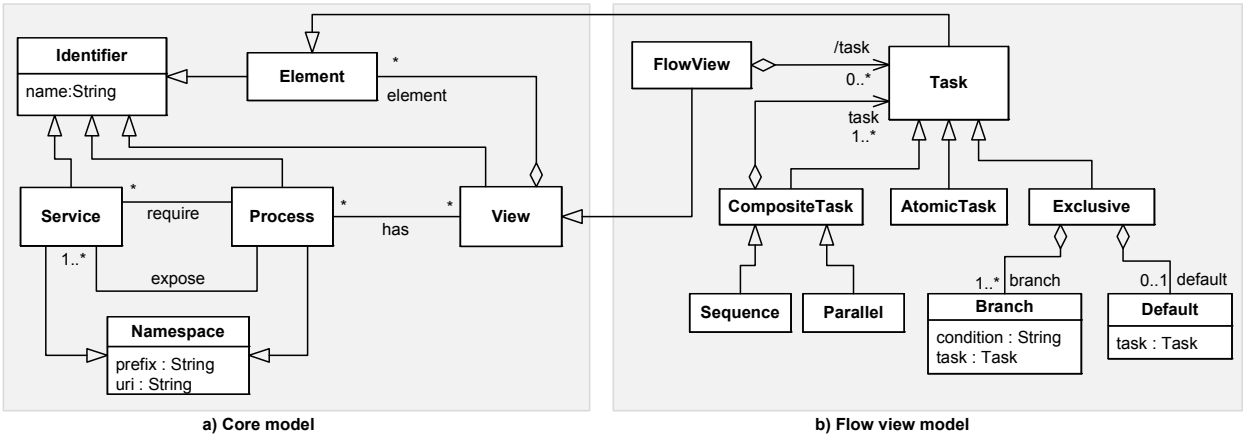


Figure 5: The Core model (left) and the Flow view model (right)

Using low-level models for describing the technology specifics of the models and the DSLs (see Section 3.4), the process models are readied for producing schematic process code as well as configurations via code generators needed to deploy and monitor the execution of the business process.

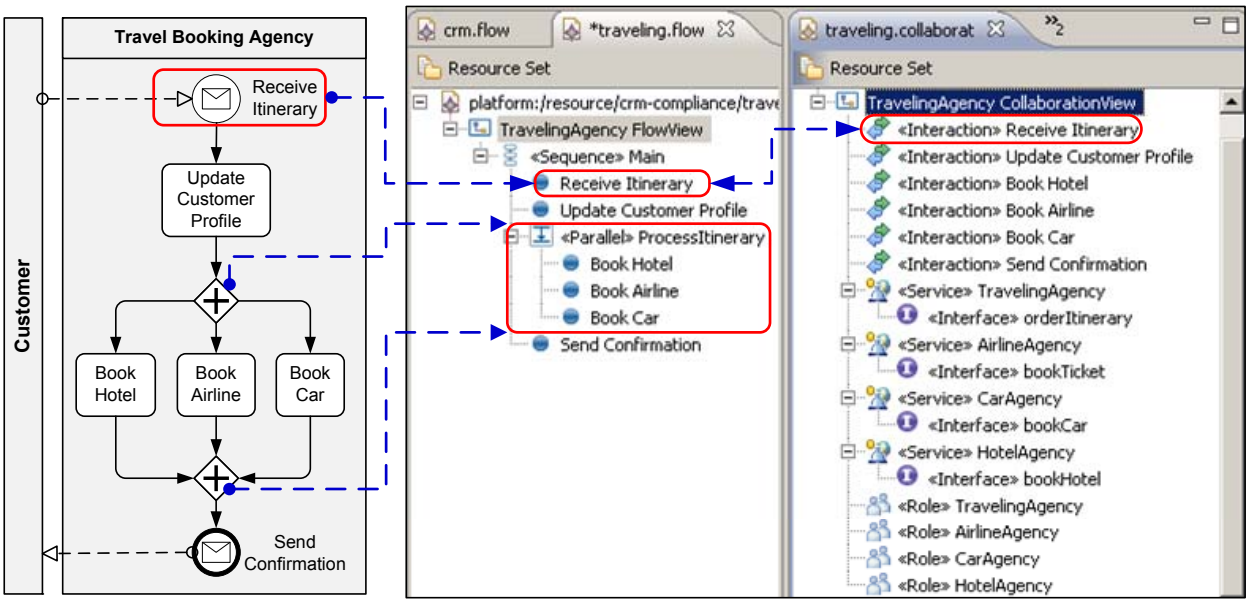
Our view-based approach is not limited to these above named concerns, but can be extended to cover other concerns. Examples of existing extensions are views for data object access [25] and human interactions [13]. An extension is accomplished using the following approach: A new concern can be integrated into VbMF by devising a new view model that extends the basic concepts of the Core model and defines additional concepts of that concern. By adding new view models for additional process concerns, we can extend the view-based approach along the *horizontal dimension*, i.e., the dimension of process concerns, to deal with the complexity caused by the tangling of process concerns [43]. Along the *vertical dimension*, i.e., the dimension of abstraction, our view-based approach provides two essential layers: the abstract layer and technology-specific layer. 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. A technology-specific model can be either directly derived from the Core model or as an extension of an abstract view model by using the model extension mechanism provided in our view-based approach [43]. Figure 5 shows the Core model, which is the basis for creating the other view models, and the Flow view model, which derives and extends the basic concepts of the Core model to represent the control flows of business processes.

Figure 6 presents a small example of modeling the business process of a Travel Booking agency [15] using VbMF. The Travel Booking process (Figure 6(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 aforementioned description of the Travel Booking process is modeled by using VbMF's views. The Travel Booking Flow view (Figure 6(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 views of the process concerns. For instance, the task `Receive Itinerary` waits for the customer's request, and therefore, is described in the Travel Booking Collaboration view (Figure 6(c)).

3.4. DSLs for compliance concerns

To offer expressive and convenient languages for the different stakeholders, our approach provides a separation of DSLs into multiple sub-languages, where each sub-language is tailored for the appropriate stakeholders – see



a) Travel Booking Process b) Travel Booking Flow view c) Travel Booking Collaboration view

Figure 6: Modeling business process with VbMF: A Travel Booking process example

[30] for further details. Hence, it is possible to provide multiple different levels of abstractions where each level of abstraction is tailored for the designated stakeholders. The number of different levels of abstractions depends on the problem domain, as well as on the number of the different stakeholders. Hence, we can better provide tailored and user-friendly representations for each group of the stakeholders.

The following Sections 3.4.1 and 3.4.2 exemplify our approach of providing high-level and low-level DSLs by an example. The example illustrates (1) how compliance concerns – especially 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.

3.4.1. The high-level QoS DSL

The high-level QoS DSL has the requirement that domain experts should be able to model which QoS compliance concerns have to be measured for a certain business process to fulfill some Service-Level Agreements (SLAs), as well as the actions to be taken when the SLAs are violated. The high-level DSL should 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 concentrate on the specification of runtime and performance related QoS measurements as specified in [35]. Furthermore, it should be possible to define actions which should be performed if an SLA is violated. An example of a high-level specification of the QoS compliance concerns is: "*In case the availability of a service is less than 99%, an e-mail should be sent to the system administrator*".

The language model of the high-level QoS DSL.

Figure 7 illustrates VbMF's Core model and demonstrate how QoS compliance concerns can be correlated with VbMF process models. In the same way, other compliance concerns can be woven into the view-model-based compliance framework as illustrated in Figure 4.

`Identifiers` – `Services` or `Processes` – of the VbMF Core model can be augmented with `QoSMeasurements`. The model provides the possibilities for specifying `Performance` and `Runtime` related QoS measurements. Each

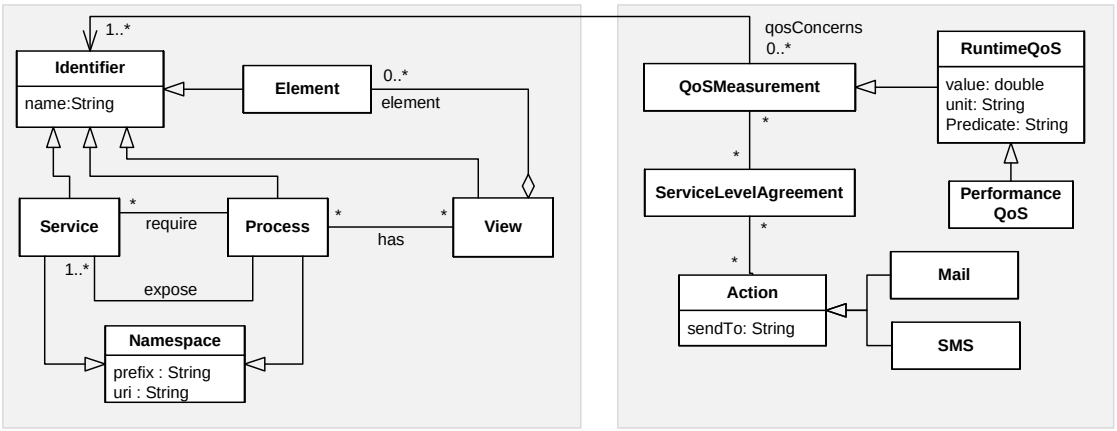


Figure 7: An example of annotating processes or services with QoS compliance concerns

QoS measurement has relations to contractually negotiated `ServiceLevelAgreements` which are in relation with different `Actions` that should be performed if an SLA is violated. In future works we have planned to extend the modeling of SLAs, such as the modeling of the involved parties.

An example of using the high-level QoS DSL.
To demonstrate an example of the high-level QoS DSL, let us consider that the task `Receive Itinerary` of the Travel Booking process example in Figure 6 must have a latency of less than 4 days and an availability of more than 99%. Figure 8 illustrates how the high-level QoS DSL can be used for annotating processes and services with QoS compliance concerns.

```
## 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 8: An example of using the high-level QoS DSL

First, the user of the high-level DSL – the domain expert – has to specify the required QoS compliance concern. As illustrated, a required latency and a required availability are specified. The `ItineraryLatency` instance is specified as an instance of the `PerformanceQoS` class. Then, the values of the attributes, such as the `predicate` attribute, are set. We can do the same for the availability. An instance of the `RuntimeQoS` class – `ItineraryAvailability` instance – is created and the attributes are set. Afterwards, the defined QoS requirements are assigned to the services. In this case, the `ReceiveItineraryAvailability` and `ItineraryLatency` instances are assigned to the `ReceiveItinerary` service of the Travel Booking process.

3.4.2. The low-level QoS DSL

Technical experts need a language for specifying how the different QoS values are measured in the used Web service framework, as well as how the defined actions are executed or performed. We decided to use the open-source Apache CXF Web service framework[1] in our prototype.

[1] http://cxf.apache.org

The language model of the low-level DSL extends the high-level language model for modeling the technological aspects to generate a running system from the model instances described by using the DSLs. Similar to the high-level DSLs, the provided constructs and expressions of the low-level DSL are named similar to the appropriate technology to enable technical experts to easily make the link to the technology. The expressions of the low-level QoS DSL depend on the technology on which the DSL is based.

The technical requirements on the low-level QoS DSL are as follows: The message-flows between the service client and the service provider are based on chains. 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 phases, in 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 or for performing the required actions. In this case, interceptors do not need to be specified within the models because just by specifying the QoS measurements and the corresponding phases, the interceptors can be generated automatically.

The language model of the low-level QoS DSL.

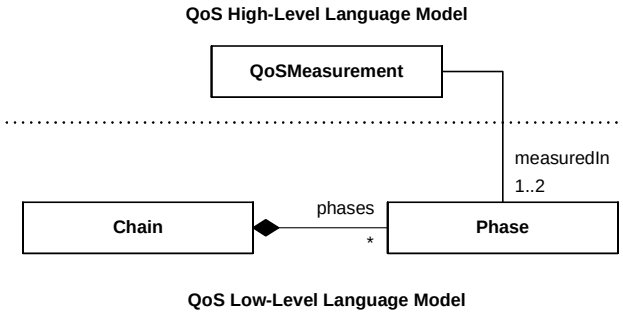


Figure 9: Extending the high-level QoS language model with additionally needed technical details

Figure 9 presents the low-level language model and how it extends the high-level language model. Following the technical requirements mentioned above, the message-flow between the service client and the service provider is based on `Chains`. Each chain – incoming and outgoing – consists of a number of `Phases`. Runtime QoS concerns, such as the `ResponseTime`, can be measured within the corresponding phases by automatically generated interceptors.

An example of using the low-level QoS DSL.
Similar to the high-level DSL, the low-level DSL provides a textual syntax. Figure 10 depicts an example of how technical experts can use the low-level DSL to extend the high-level domain concepts with the additionally needed technical aspects.

```
# 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 10: An example of using the low-level QoS DSL

In this example, the technical expert has to specify in which phases the QoS values should be measured. First, the phases are defined – `OutSetup` and `OutSetupEnding`. Then, it is defined that the `Latency` – defined as an instance of the high-level `PerformanceQoS` class – has to be measured between the two phases. This is defined by using the `measuredInPhases` relationship between the `QoSMeasurement` and `Phase` classes.

3.5. Compliance Metadata model

In this section we propose 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 as well as compliance sources. As described in Section 3.3, the Compliance Metadata model correlates process view models with compliance metadata such as compliance documents, requirements, risks, and so on. On the one hand, such annotations can be used for facilitating automatic compliance *controls* (cf. 3.1). That is, services or processes are distinguished by annotation to implement and realize automatic controls. On the other hand, the metadata serves as an information source for automatically generating reports and documentations 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.

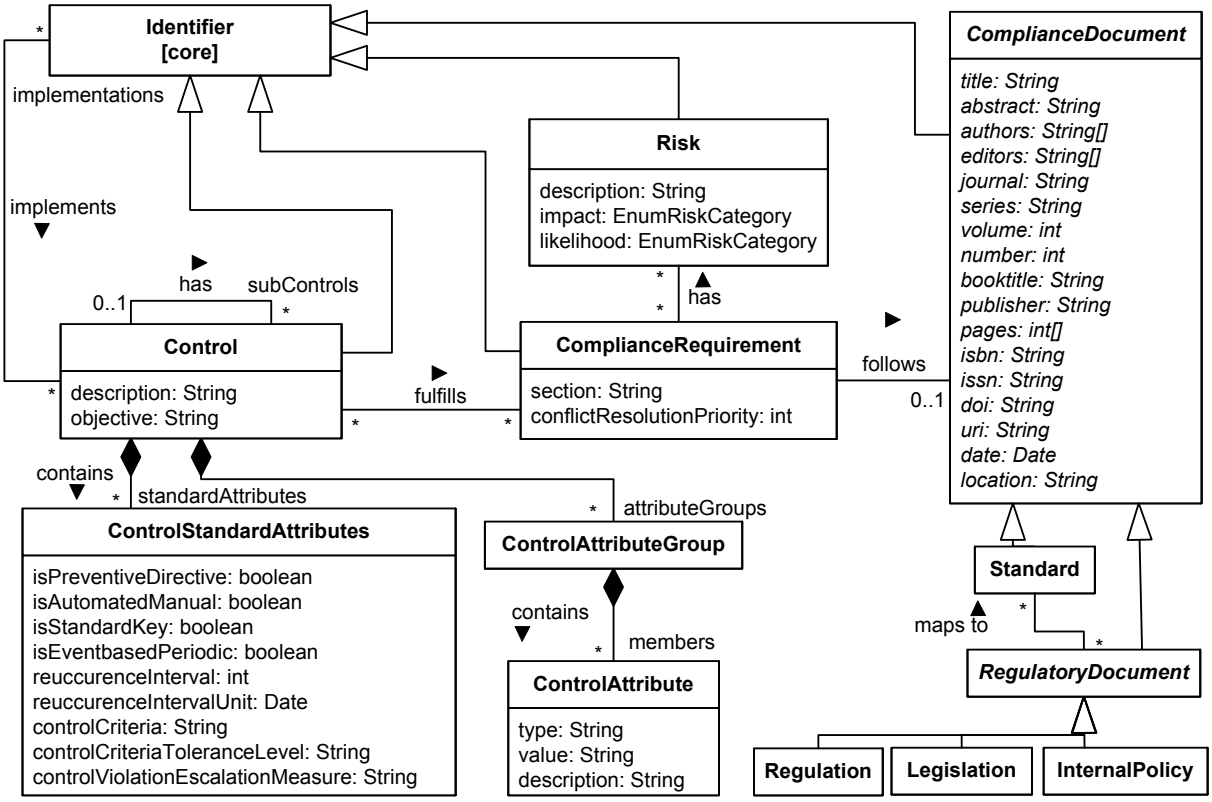


Figure 11: The Compliance Metadata model

Figure 11 shows our proposed model for expressing compliance metadata. Annotation of specific SOA elements with compliance metadata is done using compliance `Controls` that relate to concrete `implementations` such as a process or service (cf. the Core model of Figure 5). Note, that besides processes and services also DSLs for compliance concerns such as the QoS DSL are integrated into the framework using `Identifiers` (see also Figure 7). As such they can be related to from the compliance metadata (see also Figure 4).

A `Control` can have `subControls`. This way compliance controls can be grouped and combined. `Controls` fulfill `ComplianceRequirements` that relate to `ComplianceDocuments` such as a `Regulation`, `Legislation`, or `InternalPolicy`. Such `RegulatoryDocuments` can be mapped to `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.

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 allows for the 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.

For documentation purposes and for the implementation of compliance controls the `ControlStandard-Attributes` help to specify general metadata for compliance controls, e.g., if the control is automated or manual (`isAutomatedManual`). Besides these standard attributes, individual `ControlAttributes` can be defined for a compliance control within `ControlAttributeGroups`.

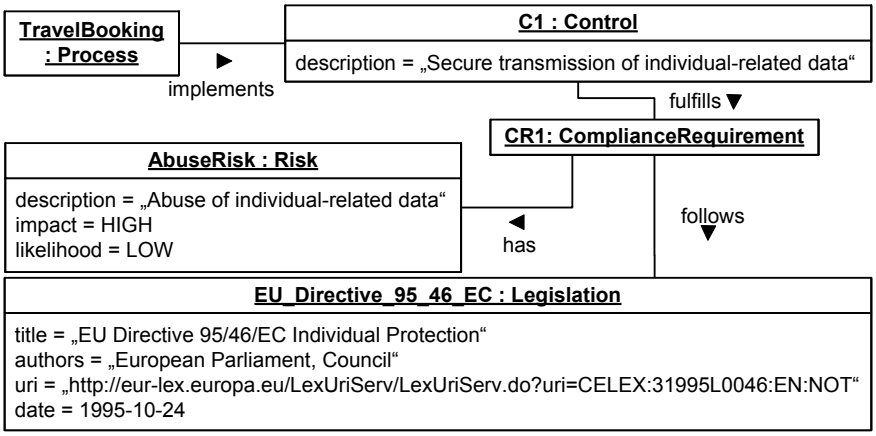


Figure 12: Example for a Compliance Metadata model instance

Figure 12 shows a model instance for the compliance metadata that contains a directive from the European Union on the protection of individuals with regard to the processing of personal data. 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*[1] *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.

3.6. Model validation and generation

To ensure the correctness of the defined model instances, different kinds of design time validations can be performed. Figure 13 illustrates the validation and generation process of our approach using an example. As described in the previous sections, various DSLs exist for creating instances of the compliance concerns' models. Also, existing tools can be used to create process instances, such as process modeling tools. All model instances are provided to the model validator that checks whether the models can be properly integrated and whether all static OCL-like constraints hold. The upper box, which is labeled with **Constraints**, shows some constraints which were defined using the Frag Constraint Language (FCL) [48] – 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.

[1] http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT

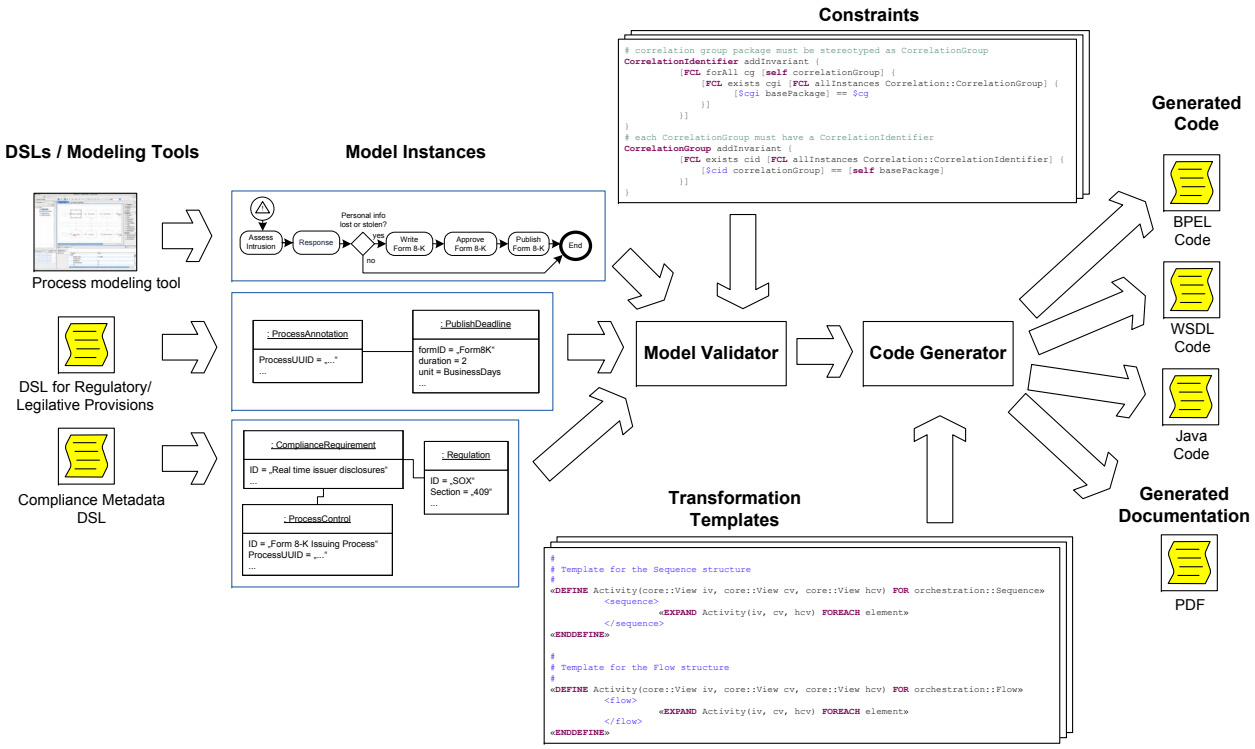


Figure 13: Generation and validation example

After the static 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 instance into code in various executable languages, such as BPEL, WSDL, and Java service code. The code generator also generates the compliance documentation from the Compliance Metadata models. We use the template-based transformation technique provided by openArchitectureWare's XPand language [34]. 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 documentations. 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 supports compliance at runtime via model to code transformations that translate concepts and rules described in compliance DSLs into components or directives. These components and directives can be deployed at process engines and application servers 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[1]. The process is designed using BPMN[2] and implemented using process-driven SOA technology: BPEL[3]

[1] http://sembiz.org/attach/D4.1.pdf
[2] http://www.omg.org/spec/BPMN/1.1/
[3] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

and WSDL[1]. BPMN, BPEL, and WSDL are used for exemplification because these are widely adopted in research and industry today. Nevertheless, our approach is not limited to those technologies but is generally applicable for other process-driven SOA technologies.

In the context of the CRM Fulfillment process (see Figure 14), 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.

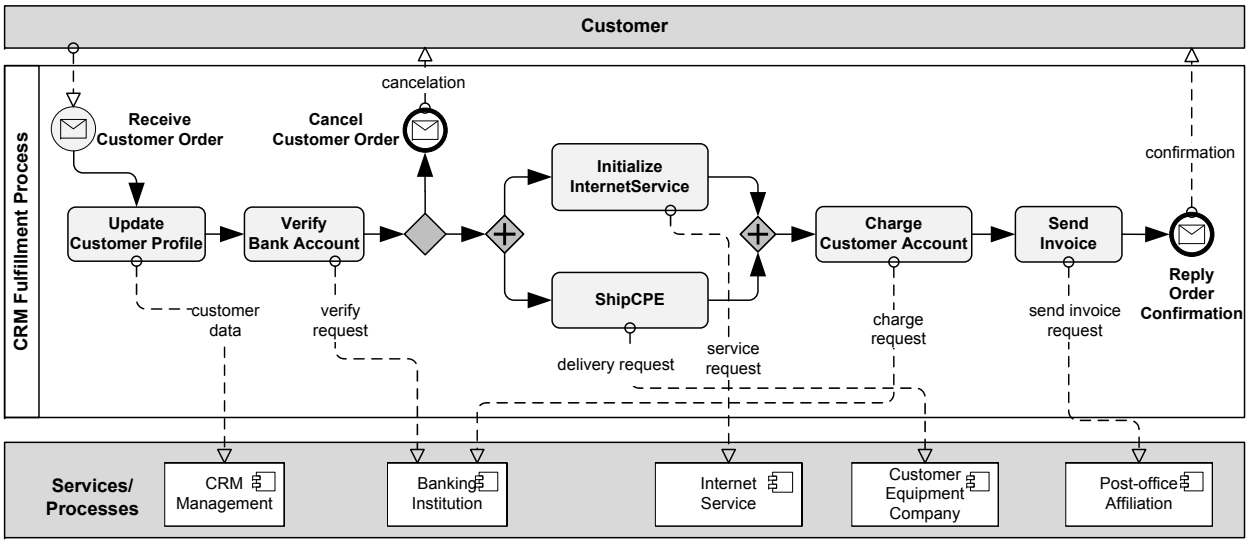


Figure 14: Case study: A CRM Fulfillment process

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

[1] http://www.w3.org/TR/wsdl

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 `Initialize InternetService` 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

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.

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

4.3. Modeling the CRM Fulfillment process

The View-based Modeling Framework [43] supports stakeholders in modeling and developing business processes by using the notion of views to separate the various process concerns. Figure 15 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

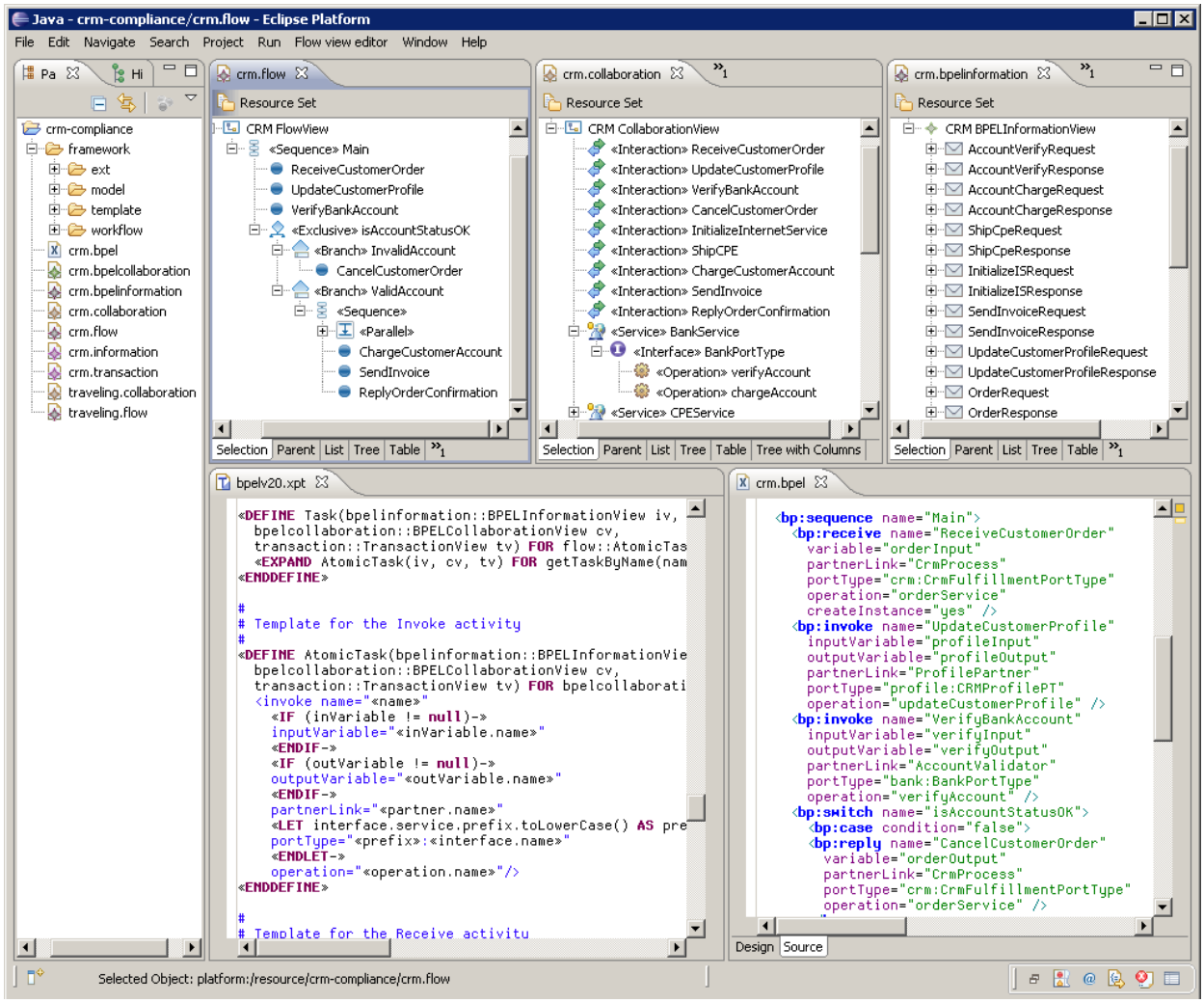


Figure 15: Using VbMF to model the CRM Fulfillment process

The stakeholders, 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. The integration of views are accomplished by using view integration mechanisms provided in VbMF [43]. These process views are then passed to the VbMF code generator in the last step described in Section 3.6 for generating process executable code in BPEL and process service interface in WSDL. In the lower part of Figure 15, 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 have to be integrated into the business process through the Compliance Metadata model. 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 3.4, in this section we want to illustrate the usage 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

Figure 16 illustrates the annotation of the CRM Fulfillment process with QoS compliance concerns by using the implemented high-level QoS DSL. The modeling of the two controls **C6** and **C7**, described in Table 1, is shown.

```
## 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 16: Specifying the required QoS compliance concerns by using the high-level QoS DSL

First, the business expert has to `import` the CRM Fulfillment process which was modeled in the VbMF. Now, the QoS compliance concerns have to be specified and assigned to the process and its activities or services. As shown, the CRM Fulfillment process must have a latency of maximum 3 hours. Also, the banking service must have an availability of 99%.

4.4.2. Using the low-level QoS DSL

Figure 17 depicts an excerpt of using the low-level QoS DSL regarding the CRM case study. The technical expert has to specify during which phases of the Apache CXF Web service framework the `Latency` has to be measured. As shown, the low-level specifications in Figure 17 are the same as in Figure 10.

```
# 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 17: Specifying the additionally needed technical aspects by using the low-level QoS DSL

Now, the advantages of our separation into high- and low-level languages can be seen. 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

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. 3.4.2). Figure 18 illustrates excerpts of the generated interceptors for measuring the required latency of the banking service.

```
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 18: The generated interceptors for measuring the required QoS values

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 is 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.

Figure 19 depicts an instance of the compliance metadata model shown in Figure 11. The model instance describes the control `C1` of Table 1. The `CRMFulfillment_Process` implements the control `C1` which fulfills the `ComplianceRequirement`. The compliance requirements originate from the `Basel_II` compliance document and have an `AbuseRisk`. The `impact` and `likelihood` of the risk are `HIGH`.

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 metamodel for accommodating a relevant control with the current requirements and compliance concerns. That is, a new requirement is

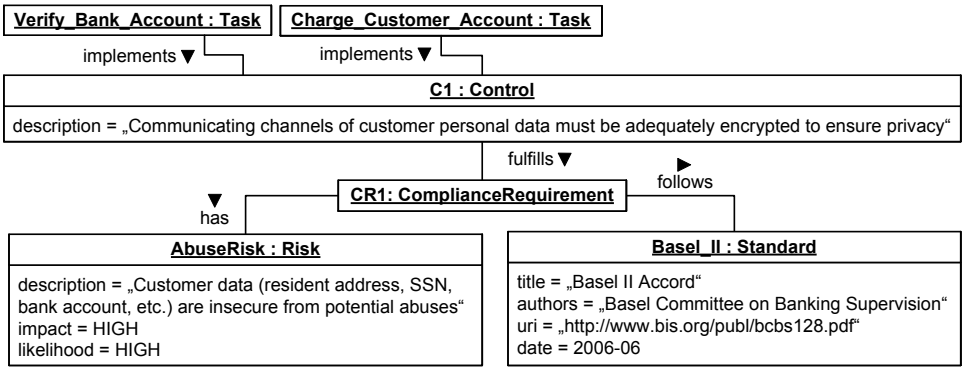


Figure 19: Compliance Metadata view for the Information Security compliance requirement

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 documentations. Such documentations visualize 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.

```
<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: XPand template for generating the compliance risk-control matrix

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.

Figure 21 shows a 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

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 [38]. 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.

REO [2] is a channel-based, formal coordination model that can be used to model compliance for behavioral models, including business processes represented in, for instance, BPMN and BPEL. 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 [6] 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. [23, 24] present 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. [28] 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. [3] present 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.

Liu et al. [19] 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 Linear Temporal Logic (LTL). Static compliance checking is accomplished by the model-checker NuSMV2 [5] 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 [20] 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 [10, 26]. Both approaches use the same formal representation, namely, Formal Contract Language (FCL), for expressing the compliance constraints derived from business contracts. In [10], 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 [26] 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.

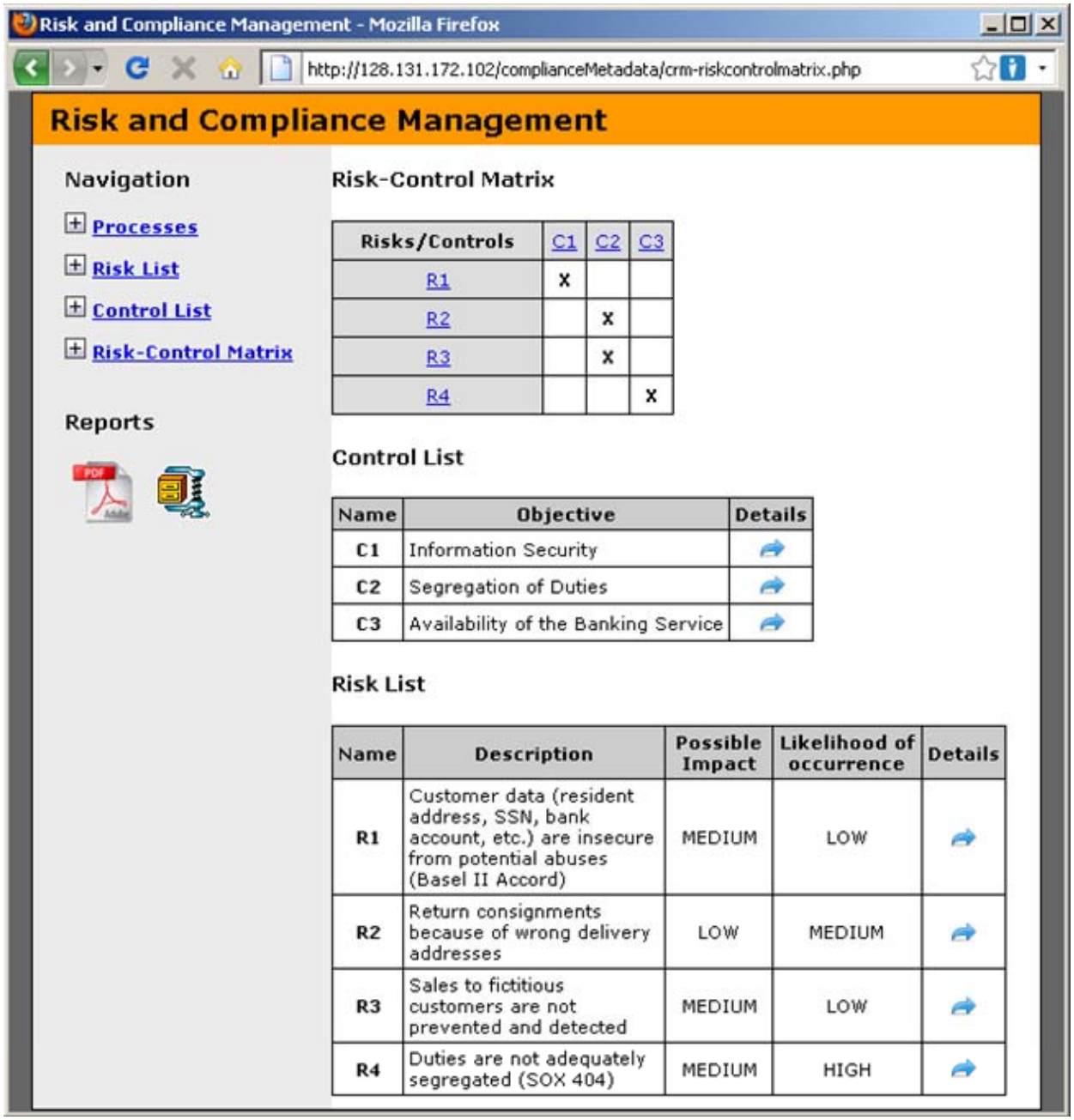


Figure 21: Compliance risk-control matrix

The compliance aware business process design framework proposed by Sadiq et al. [9, 21, 22, 39, 40] 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 [10]. The framework described in [17, 37] 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. [7, 8] 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.

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. [47] 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. [36] 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 [1], 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 [19, 40]) 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 [20]), 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, documentations of the implementation of relevant compliance requirements in business processes are crucial evidence for compliance auditing. Moreover, the documentations 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 4), can be used to generate documentations 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. [3]	Not supported	External model checking validates queried process models against compliance constraints n PLTL expressions	Not supported
ExPDT [17, 37]	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 [8]
Contract compliance [10, 26]	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. [6]	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. [19]	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. [23, 24]	Not supported	Process models are verified against semantic constraints for their correctnesses	Not supported
MASTER [20]	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. [28]	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 [7, 8]	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 [2]	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 [1, 36, 47]	Not supported	Not supported	Process models formalized and verified against the events producing during process executions to detect the non-compliant behaviors
Sadiq et al. [9, 21, 22, 39, 40]		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 [33]	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.

Table 2: Comparing compliance solutions

	Supported compliance concerns	Extensibility options	Support for involving domain experts	Documentation of compliance
Awad et al. [3]	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 [10, 26]	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

(Continued on next page)

	Supported compliance concerns	Extensibility options	Support for involving domain experts	Documentation of compliance
ExPDT [17, 37]	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. [6]	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. [19]	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. [23, 24]	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 [20]	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. [28]	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 [7, 8]	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 documentations for regulatory associated with policy rules via the metadata
REO [2]	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 [1, 36, 47]	Mainly focus on the behavioral concerns	Not supported	Process models are represented in high level and intuitive formalisms, e.g., Petri-net	Not supported
Sadiq et al. [9, 21, 22, 39, 40]	Based on FCL, same as [10, 26]	Same as [10, 26]	[40] provides annotated process visualizations for domain experts	Potentially supported but not mentioned
SoaML [33]	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 documentations can be generated	Compliance Metadata model: reports, visualizations, and documentations can be generated

Table 3: Comparing compliance solutions (cont'd)

6. Conclusions

In this article, we have presented a novel approach and associated architecture for dealing with compliance in process-driven SOAs. In contrast to the related work, our approach can support stakeholders to deal 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.

Acknowledgment

This work was supported by the European Union FP7 project COMPAS, grant no. 215175.

References

[1] R. Accorsi. *Automated Counterexample-Driven Audits of Authentic System Records*. PhD thesis, University of Freiburg, Germany, 2008.

[2] F. Arbab, N. Kokash, and S. Meng. Towards using reo for compliance-aware business process modeling. In T. Margaria and B. Steffen, editors, *Proc. of the Third Intl. Sym. on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2008)*, volume 17 of *CCIS*, pages 108–123. Springer, 2008.

[3] A. Awad,, G. Decker,, and M. Weske,. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In *BPM '08: 6th Intl. Conf. Business Process Management*, pages 326–341, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] 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, June 2004. [Accessed 2009/02/01].

[5] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *14th Intl. Conf. Computer Aided Verification (CAV'02)*, volume LNCS 2404, pages 241–268. Springer, July 2002.

[6] A. Ghose, and G. Koliadis,. Auditing business process compliance. In *ICSOC '07: 5th Intl. Conf. on Service-Oriented Computing*, pages 169–180, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] C. Giblin, A. Y. Liu, and X. Zhou. Regulations expressed as logical models (REALM). In A. I. O. S. Press, editor, *Proc. of the 18th Annual Conference on Legal Knowledge and Information Systems (JURIX 2005)*, pages 37–48, 2005.

[8] C. Giblin, S. Müller, and B. Pfitzmann. From regulatory policies to event monitoring rules: Towards model-driven compliance automation. Technical Report RZ 3662, IBM Research, 2006.

[9] G. Governatori, J. Hoffmann, S. Sadiq, and I. Weber. Detecting regulatory compliance for business process models through semantic annotations. In *BPD-08: 4th Intl. Workshop on Business Process Design*, Sept. 2008.

[10] G. Governatori, Z. Milosevic, and S. W. Sadiq. Compliance checking between business processes and business contracts. In *EDOC*, pages 221–232, 2006.

[11] J. Greenfield, K. Short, S. Cook, and S. Kent. *Software Factories: Assembling Applications with Patterns, Frameworks, Models & Tools*. J. Wiley and Sons Ltd., 2004.

[12] C. Hentrich and 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, July 2006.

[13] T. Holmes, H. Tran, U. Zdun, and S. Dustdar. Modeling human aspects of business processes – A view-based, model-driven approach. In *Fourth European Conference on Model Driven Architecture Foundations and Applications (ECMDA'08)*. Springer LNCS, June 2008.

[14] IASB. International Financial Reporting Standards (IFRSs). http://www.iasb.org/IFRS+Summaries/, 2007. [Accessed 2009/02/01].

[15] IBM. Travel Booking Process. http://publib.boulder.ibm.com/bpcsamp/scenarios/travelBooking.html, 2006. (accessed 2009/01/05).

[16] IBM, BEA Systems, Microsoft, SAP AG, and Siebel Systems. Business Process Execution Language for Web Services. ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, May 2003.

[17] M. Kähmer, M. Gilliot, and G. Muller. Automating Privacy Compliance with ExPDT. In *CEC/EEE*, pages 87–94, 2008.

[18] S. Kelly and J. P. Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. John Wiley & Sons, 2008.

[19] Y. Liu,, S. Müller,, and K. Xu,. A static compliance-checking framework for business process models. *IBM Syst. J.*, 46(2):335–361, 2007.

[20] V. Lotz, E. Pigout, P. M. Fischer, D. Kossmann, F. Massacci, and A. Pretschner. Towards systematic achievement of compliance in service-oriented architectures: The MASTER approach. *WIRTSCHAFTSINFORMATIK*, 50(5):383–391, Oct. 2008.

[21] R. Lu, S. W. Sadiq, and G. Governatori. Compliance aware business process design. In *Business Process Management Workshops*, pages 120–131, 2007.

[22] R. Lu, S. W. Sadiq, and G. Governatori. Measurement of compliance distance in business processes. *IS Management*, 25(4):344–355, 2008.

[23] L. T. Ly, K. Gser, S. Rinderle-Ma, and 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.

[24] L. T. Ly, S. Rinderle, and P. Dadam. Integration and verification of semantic constraints in adaptive process management systems. *Data Knowl. Eng.*, 64(1):3–23, 2008.

[25] C. Mayr, U. Zdun, and S. Dustdar. Model-Driven Integration and Management of Data Access Objects in Process-Driven SOAs. In *ServiceWave '08: Proc. of the 1st European Conf. on Towards a Service-Based Internet*, pages 62–73, 2008.

[26] Z. Milosevic, S. W. Sadiq, and M. E. Orlowska. Translating business contract into compliant business processes. In *EDOC*, pages 211–220, 2006.

[27] Ministre de l'conomie, des finances et de l'industrie. Loi de Sécurité Financière (LSF). http://www.senat.fr/leg/pjl02-166.html, Aug. 2003. [Accessed 2009/02/01].

[28] K. Namiri and N. Stojanovic. Pattern-Based Design and Validation of Business Process Compliance. In *OTM Conferences (1)*, pages 59–76, 2007.

[29] OASIS. Business Process Execution Language (WSBPEL) 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf, May 2007.
[30] E. Oberortner, U. Zdun, and 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*, pages 20–25, Vancouver, BC, Canada, 2009.
[31] OMG. Unified Modelling Language (UML) 2.0. http://www.omg.org/spec/UML/2.0/, July 2005. [Accessed 2009/02/01].
[32] OMG. Business Process Modeling Notation (BPMN) 1.1. http://www.omg.org/spec/BPMN/1.1/, Jan. 2008.
[33] OMG. Service-Oriented Architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services. Technical report, OMG, 2008.
[34] openArchitectureWare.org. openArchitecture Model-driven Framework. http://www.openarchitectureware.org, Aug. 2002. [Accessed 2009/02/01].
[35] S. Ran. A model for web services discovery with qoS. *SIGecom Exch.*, 4(1):1–10, 2003.
[36] A. Rozinat, and W. M. P. van der Aalst,. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.
[37] S. Sackmann and M. Kähmer. ExPDT: A Policy-based Approach for Automating Compliance. *WIRTSCHAFTSINFORMATIK*, 50(5):366–374, Oct. 2008.
[38] S. Sackmann, M. Kahmer, M. Gilliot, and L. Lowis. A classification model for automating compliance. In *10th IEEE Conf. EEE/CEC*, pages 79–86, July 2008.
[39] S. Sadiq and G. Governatori. *A methodological framework for aligning business processes and regulatory compliance*. Springer, Handbook of Business Process Management edition, 2009.
[40] S. W. Sadiq, G. Governatori, and K. Namiri. Modeling control objectives for business process compliance. In *BPM*, pages 149–164, 2007.
[41] T. Stahl and M. Völter. *Model-Driven Software Development*. John Wiley & Sons, 2006.
[42] The Netherlands Corporate Governance Committee. The Dutch corporate governance code. http://www.commissiecorporategovernance.nl/page/downloads/CODE DEF ENGELS COMPLEET II.pdf, Dec. 2003. [Accessed 2009/02/01].
[43] H. Tran, U. Zdun, and 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)*, volume 116 of *LNI*, pages 105–124, Sept. 2007.
[44] H. Tran, U. Zdun, and S. Dustdar. View-Based Reverse Engineering Approach for Enhancing Model Interoperability and Reusability in Process-Driven SOAs. In H. Mei, editor, *10th Intl. Conf. on Software Reuse (ICSR'08)*, LNCS, pages 233–244. Springer, May 2008.
[45] U.K. Financial Services Authority. Markets in Financial Instruments Directive (MiFID), Nov. 2007. [Accessed 2009/02/01].
[46] 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, Jan. 2002. [Accessed 2009/02/01].
[47] W. M. P. van der Aalst, H. T. de Beer, and 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*, pages 130–147. Springer, 2005.
[48] U. Zdun. Frag. http://frag.sourceforge.net/, 2005.