Business Compliance Governance in Service-Oriented Architectures

Florian Daniel, Fabio Casati, Vincenzo D'Andrea, Steve Strauch,
David Schumm, Frank Leymann, Emmanuel Mulo, Uwe Zdun,
Schahram Dustdar, Samir Sebahi, Fabien de Marchi, Mohand-Said Hacid

University of Trento, Italy
{daniel, casati, dandrea}@disi.unitn.it

Universität Stuttgart, Germany
{strauch, schumm, leymann}@iaas.uni-stuttgart.de

Vienna University of Technology, Austria
{e.mulo, zdun, dustdar}@infosys.tuwien.ac.at

Université Claude Bernard Lyon 1, France
{samir.sebahi,fabien.demarchi,mohand-said.hacid}@liris.cnrs.fr

BIBTEX:

```
@inproceedings{DanielCDS09,
  author    = {Florian Daniel and Fabio Casati and Vincenzo D’Andrea and
               Steve Strauch and David Schumm and Frank Leymann and
               Emmanuel Mulo and Uwe Zdun and Schahram Dustdar and Samir Sebahi
               and Fabien de Marchi and Mohand-Said Hacid},
  title     = {Business Compliance Governance in Service-Oriented
               Architectures},
  booktitle = {Proceedings of the 23rd International Conference on
               Advanced Information Networking and Applications, AINA '09.,
               26-29 May 2009, Bradford, United Kingdom},
  year      = {2009},
  pages     = {113--120},
  doi       = {10.1109/AINA.2009.112},
  publisher = {IEEE Computer Society}
}
```

Business Compliance Governance in Service-Oriented Architectures

Florian Daniel, Fabio Casati,
Vincenzo D'Andrea
University of Trento, Italy
{casati,daniel,dandrea}@disi.unitn.it

Steve Strauch, David Schumm, Frank Leymann
Universität Stuttgart, Germany
{steve.strauch,david.schumm,leymann}@
iaas.uni-stuttgart.de

Emmanuel Mulo, Uwe Zdun,
Schahram Dustdar
Vienna University of Technology, Austria
{e.mulo,zdun,dustdar}@infosys.tuwien.ac.at

Samir Sebahi, Fabien de Marchi, Mohand-Said Hacid
Université Claude Bernard Lyon 1, France
{samir.sebahi,fabien.demarchi,mohand-said.hacid}@
liris.cnrs.fr

Abstract — **Governing business compliance with regulations, laws, best practices, contracts, and the like is not an easy task, and so far there are only limited software products available that help a company to express compliance rules and to analyze its compliance state. We argue that today's SOA-based way of implementing and conducting business (e.g., using Web services and business process engines) lends itself very well to the development of a comprehensive compliance government solution that effectively aids companies in being compliant. In this paper, we contextualize the compliance problem in SOA-based businesses, we highlight which are the most salient research challenges that need to be addressed, and we describe our approach to compliance governance, spanning design, execution, and evaluation concerns.**

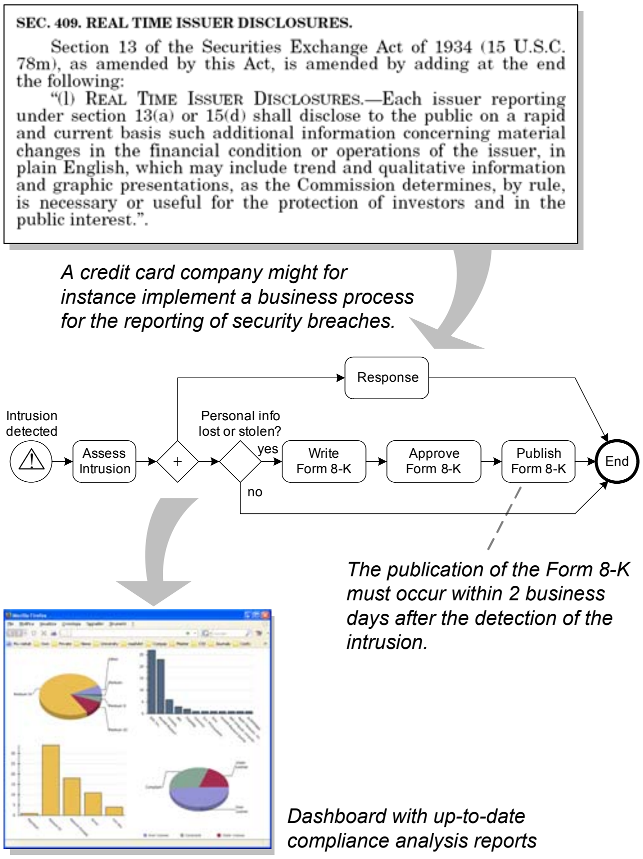
Keywords* — *Compliance, Compliance governance, SOA, Business process management, Business process fragments, View-based modeling

I. Introduction

Business compliance, i.e., the conformance of a company's activities and business practices with existing regulations (as well as laws, best practices, contracts, agreements, policies, and so on), is a major concern of today's business community. For instance, as response to the unexpected cracks of companies such as Enron and Worldcom in the U.S. or Parmalat in Italy, over the last years governments have been increasing the legislative pressure and auditing requirements for companies. The aim of such particular measures is to enhance the transparency of business decisions and augment the accountability of responsibilities. The final goal is protecting investors and stakeholders from fraud, corruption or corporate misconduct. In general, however, companies may want to be compliant for a variety of different aims, e.g., to obtain quality certificates, to take advantage of industry best practices, for internal controlling purposes, etc.

If, instead, we look at how every-day business is being conducted at an operative level, we note that technologies like Web services and business process management systems have largely proved their viability for organizing work and assisting and orchestrating also human actors involved in business processes. The adoption of the so-called service-oriented architecture (SOA) to conduct business (eased by technologies such as SOAP, WSDL, and HTTP) has further affirmed the analogy between Web service technologies and common business practices. As a consequence, today's business reality is characterized by a high level of automation and IT support, and the SOA, as architectural paradigm, is increasingly spreading.

We argue that it is time the two worlds, i.e., compliance governance and service-oriented architectures, meet. Up to now, business compliance has typically been achieved through manual controls performed by a "compliance expert" without any automated assistance and without taking advantage of the many possibilities offered by current technologies. In this paper, we propose a comprehensive, SOA-based compliance governance solution and show how both companies and auditors may benefit from such kind of software support. Specifically, we stress that compliance starts from the very beginning of a business, i.e., from the design of its business processes and practices; for the design of compliant business processes, we propose a model-driven development approach aimed at easing the specification of compliance-specific concerns. We show how such enriched process specifications can be automatically enacted and instrumented in order to produce evidence of the fulfillment of regulatory constraints. Starting from such evidence, we then propose the idea of compliance reporting dashboard, along with compliance control and process mining algorithms, so as to assess past process executions and enable root-cause analyzes for detected violations.



SEC. 409. REAL TIME ISSUER DISCLOSURES.

Section 13 of the Securities Exchange Act of 1934 (15 U.S.C. 78m), as amended by this Act, is amended by adding at the end the following:

"(l) REAL TIME ISSUER DISCLOSURES.—Each issuer reporting under section 13(a) or 15(d) shall disclose to the public on a rapid and current basis such additional information concerning material changes in the financial condition or operations of the issuer, in plain English, which may include trend and qualitative information and graphic presentations, as the Commission determines, by rule, is necessary or useful for the protection of investors and in the public interest.".

A credit card company might for instance implement a business process for the reporting of security breaches.

The publication of the Form 8-K must occur within 2 business days after the detection of the intrusion.

Dashboard with up-to-date compliance analysis reports

Figure 1 A SOX-409-compliant intrusion reporting process in a credit card company.

Next we introduce an example scenario, our compliance management life cycle, and the research challenges we identify in this area. Then we focus on the design for compliance, on compliant business process executions, and on compliance evaluation. After that, we discuss some related works, and, finally, we draw our conclusions and discuss our future work.

II. SCENARIO AND PROBLEM DEFINITION

Let's, for instance, consider a U.S. credit card company that wants to comply with Section 409 ("Real Time Issuer Disclosures") of the US Sarbanes-Oxley Act (SOX [1]), which requires that a publicly traded company discloses information regarding material changes in the financial condition of the company in real-time. Changes in the financial condition of a company that demand for disclosure are, for example, bad debts, loss of production capacity, changes in credit ratings for the company or large clients, mergers, acquisitions, major discoveries, and similar. The requirement that relevant information must be disclosed in "real-time" has so far commonly been interpreted as "within 2-4 business days" (unlike in IT contexts, where "real-time" has a much more responsive flavor).

In the case of the credit card company, the change in the financial condition we are considering as references scenario regards security breaches in the company's IT system, where personal information about customers might get stolen or lost. The business process in the middle part of Figure 1 shows a possible practice that the company may internally follow to react to a detected intrusion. After an initial assessment of the severity of the intrusion, the company immediately starts the necessary response action to mitigate risks and prevent similar future intrusions. In parallel to the response, if personal information got lost or stolen, the disclosure procedure is started. As detailed in the process, the actual disclosure of the intrusion and its potential impact on the company's financial assets is performed by filing a so-called Form 8-K report of unscheduled material events that are important to shareholders. The Form 8-K report is a specific form used to notify investors and the U.S. Securities and Exchange Commission (who is in charge of controlling compliance with SOX). As annotated in the figure, we assume that the company's internal policy is to publish the Form 8-K within 2 business days.

Note that full compliance with Section 409, of course, requires that all business practices in the company are compliant; the case of stolen or lost personal information represents only one out of multiple business practices in the credit card company that are subject to Section 409 of SOX. Hence, the compliance of the sole intrusion detection and handling process does not yet imply the compliance of the overall business.

Regarding the process depicted in Figure 1, we assume that a suitable intrusion detection system is installed in the company and that the execution of the described process is automatically assisted. Again, the mere implementation of the compliant business process does not yet guarantee compliance: failures during process execution may happen, e.g., due to human errors, system failures or the like, and the preparation and publication of the Form 8-K might be delayed, erroneous, or even forgotten. Awareness of such problems is of utmost importance to the company, in order to be able to react timely and, hence, to assure business continuity. In this regard, periodical and up-to-date reports about the compliance state of the company and, possibly, the ability to perform root-cause analyses to understand the reason for specific non-compliance problems starting from the monitoring of the company's IT infrastructure and business processes are required, so as to allow the company to improve its responsiveness and level of compliance. For instance, in Figure 1 we envision the availability of a compliance governance dashboard, assisting the company in assessing and interpreting its compliance state in an intuitive, visual fashion.

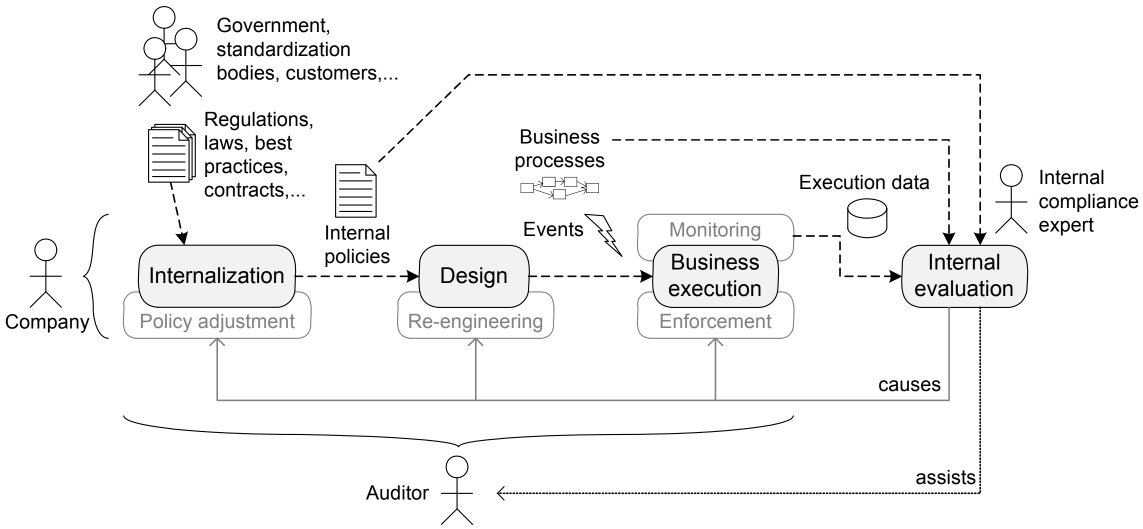


Figure 2 The compliance management life cycle with phases, products, and actors.

A. The compliance management life cycle

In Figure 2 we generalize the previous scenario and depict our interpretation of the *compliance management life cycle*, articulated into *phases*, *products* and *actors*: In its everyday business, a company is typically subject to a variety of different regulations, laws, best practices, contracts, etc. originating for instance from the government, standardization bodies, customers and the like. It is up to the company to understand and "internalize" those sources of compliance rules that directly affect its business, thus producing what we call its internal policy (*Internalization* phase). The internal policy then drives the design of the company's business practices, yielding a set of business processes that are compliant by design (*Design* phase), meaning that they are designed to respect the internal policies. Along with the definition of the actual business processes, the company also defines a set of events (often also called "controls" or "control points") that provide sufficient evidence of the (hopefully) compliant execution of the business processes.

Process and event definitions are consumed during the *Business execution* phase, where the company's employees perform the tasks and duties specified in the process models. Ideally (but not mandatorily), this execution of the work is assisted by software tools such as workflow management or business process execution systems, also able to collect compliance-specific evidence and to generate respective execution events (for monitoring), which can be stored in an audit trail or log file for later evaluation.

The *Internal evaluation* serves a twofold purpose: First, it is the moment where a company-internal expert may inspect and interpret the tracked evidence, in order to assess the company's level of compliance. Second, it is the point where collected data can be automatically analyzed in order to detect compliance violations. Indeed, designing compliant business processes is not enough to assure compliance, as in practice there are a multitude of reasons for which deviations from an expected business process might happen (e.g., human factors, system downtimes, software failures). Some of such problems can be detected during runtime, resulting in the generation of a respective event; some of them can only be detected after execution by means of, e.g., data mining or root-cause analysis techniques applied to tracked runtime data. The result of the internal evaluation might be the enforcement of corrective runtime actions (e.g., sending an alert), the re-engineering of process designs (e.g., to take into account design flaws) or the adjustment of the internal policies (e.g., to cope with inconsistent policies).

Note that the internal evaluation, however, does not yet certify a company's level of compliance; it rather represents an internal control mechanism by means of which the company is able to self-assess and govern its business. For the certification of compliance, an external auditor, e.g., a financial auditor, physically visits the company and controls (i) whether the company has correctly interpreted the existing legislation, (ii) whether business processes have been correctly implemented, and, finally, (iii) whether the business has been correctly executed.

Ideally, in a fully trusted environment, the auditor would only look at the internal evaluation tool, in order to assess the company's level of compliance, but as we are still in a very early stage of research in this area, this is not yet a realistic short-term objective. In practice, the control of the business execution is typically still based on statistical checks of documents, neither performed on the whole body of available documents nor on electronic documents only. Then, legislation is subject to interpretation; therefore, the auditor's assessment contains a subjective component that also leaves room for negotiation of the final judgment. In this regard, a company's compliance evaluation infrastruc-

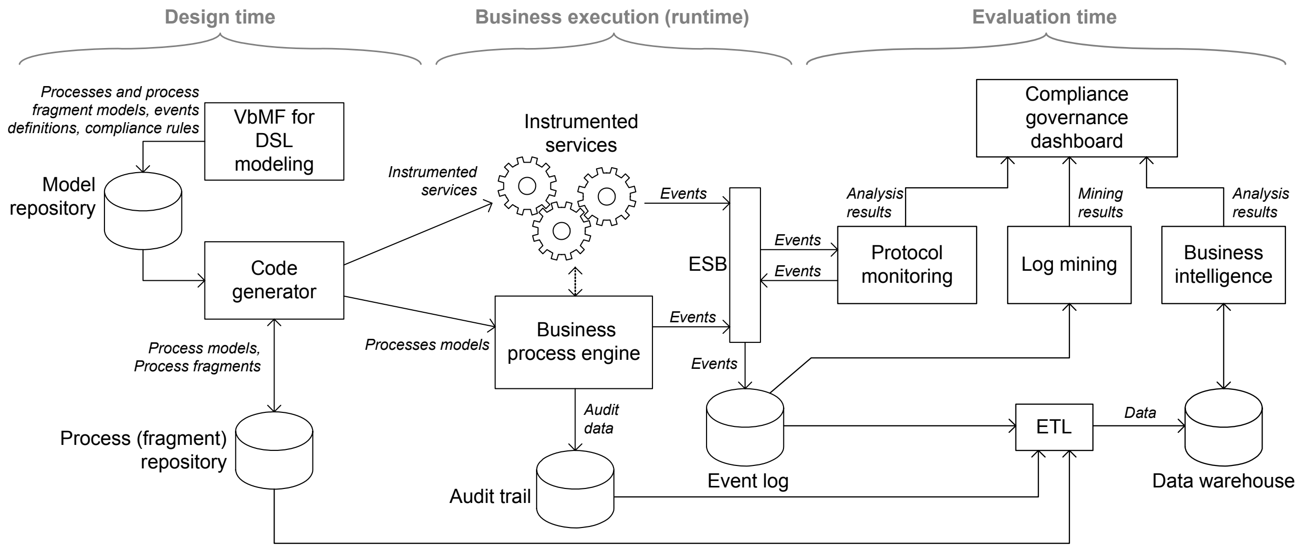


Figure 3 Overview of the COMPAS architecture for compliance governance in service-oriented architectures.

ture might assist the auditor in his/her work (e.g., via agreed upon reports and analyzes) and thereby positively affect the final assessment.

B. Research challenges

From an IT perspective, the described problem poses several interesting research challenges, especially as for what regards the Design, Business execution and internal evaluation phases. As main challenges, we identify:

- The development of *models, languages, or annotations* to formally express compliance concerns at different levels of abstraction (e.g., targeting business people and IT people);
- The development of a *generic instrumentation approach* for services and processes in order to generate the evidence that is necessary to document and evaluate compliance;
- The *assessment* of and *reporting* on the compliance state of a company by looking at the produced evidence and the specified compliance concerns;
- The identification of *hidden process patterns* and of *root causes* of recurrent non-compliance situations, enabling a process modeler to improve critical process models.

In the following we discuss the COMPAS proposal for compliance in service-oriented architectures, and we show how we address the above challenges at design time, runtime, and evaluation time.

III. A COMPLIANCE GOVERNANCE FRAMEWORK

In Figure 3 we illustrate a simplified version of the architecture of our compliance governance framework. The figure distinguishes between components that are used at design time, at runtime, and at evaluation time. We here introduce the architecture, while in the following sections we deepen its most important aspects.

At *design time*, we provide a set of domain-specific languages (DSLs) that can be used to describe the compliance concerns a company is subject to, its services and business processes, and the compliance rules that must be satisfied by them. In addition to rules, we propose the use of so-called process fragments for the specification of complex, process constraints (more details in the next section). Specifications are stored in a Model repository, which feeds a Code generator able to produce executable, instrumented services and process or process fragment models. The latter are stored in a dedicated Process (fragment) repository.

At *runtime*, a Business process engine instantiates deployed processes and interacts with the Instrumented services. Both engine and services may emit events (via a dedicated Enterprise Service Bus), communicating evidence for compliance evaluation. In addition, the engine maintains its own internal Audit trail. All published events are tracked and stored in an Event log.

At *evaluation time*, we propose three different solutions, each supporting a different feature. First, the Protocol monitoring component checks whether a process is being executed according to its specification and whether modeled process fragments are respected. Then, a Business intelligence solution generates compliance reports and computes compliance indicators out of a Data warehouse that stores all the evidence generated during Business execution (possibly also data from the Audit trail). The Data warehouse is periodically loaded by the ETL (Extract-Transform-Load) procedures that extract events from the log and transform them, taking into account process and process fragment models. Via Log mining, we provide for the discovery of recurrent process patterns and root cause analysis. Finally, the results of the three analyses are displayed in the Compliance governance dashboard, an intuitive Web interface for business and IT people alike.

IV. Design of compliant processes/services

In order to tackle the mentioned challenges regarding the design for compliance, we employ a Model-Driven Software Development (MDSD) paradigm [10]. In particular, we use domain-specific languages (DSLs), one of the bases MDSD is building on, in order to express compliance concerns. DSLs are languages that are tailored to be particularly expressive in a specific problem domain. They describe knowledge via a graphical or textual syntax that is tied to domain-specific modeling elements through a precisely specified language model. That is, DSL elements are defined in terms of a model that can be instantiated in concrete application models.

DSLs can then be used to automatically generate application and configuration code. Code generation is performed by applying to the DSL documents transformation rules and templates that specify how code is generated. The ultimate goal of the transformations is to generate code in executable languages, such as programming languages or process execution languages. The MDSD paradigm specifically aims to generate those parts of code that are schematic and recurring, and whose production hence can be automated.

The left-hand side of Figure 3 illustrates our MDSD environment that enables the rapid development of the necessary business compliance artifacts. It is composed of a View-based Modeling Framework (VbMF [11]), DSLs, transformation templates, and the Code generator. The VbMF is built on the premise that business processes comprise various concerns that require modeling support. The framework enables one to separate the various concerns into different model views. The framework provides a core model that can be extended to create views of a process, which are specific to the different concerns. Extended views then address a specific aspect of the business process that one would like to focus on, for example, process execution (control) flow, data access, or collaborations in a business process.

Figure 4 shows the hierarchy of models in the VbMF – with a special focus on compliance-oriented views and models. As can be seen in the Figure, we in particular focus on compliance regarding (i) regulatory compliance (e.g., SOX), (ii) QoS compliance (e.g., not exceeding given response times), and (iii) license compliance (e.g., software licenses). Each of these compliance concerns is provided with an own view and model to annotate business processes, which allows us to generate application code, process execution (control) code, configuration files, and service description code. That is, the compliance views allow us to instrument business processes for compliance evaluation. Instrumenting a business process mainly means extending the process, the process' services, or the system that runs the process, so as generate events or alerts during runtime that provide the evidence that is necessary to perform compliance checks.

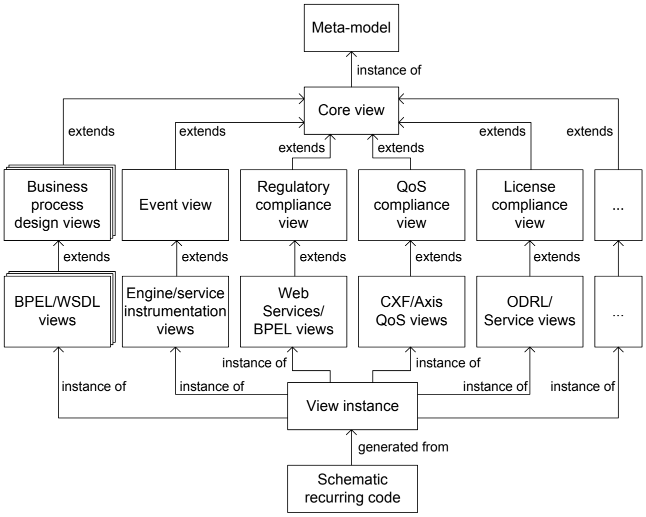


Figure 4 The View-based Modeling Framework (VbMF).

For this purpose, we define a DSL that can be used to create event models. With such a DSL, we can create event models based on the events that an organization wishes to monitor (e.g., events related to license violations). Event models contain event definitions and the conditions under which the events may occur. The models enable the generation of the instrumentation code for services and processes able to generate the events.

There are two possible ways to annotate a process model with compliance information (see Figure 5): (i) *compliance rules* in textual form, (ii) *process fragments*. We next discuss the two annotation features, while in the next section we focus on how events are generated during runtime, starting from annotations.

A. *Compliance rules*

Textual compliance rules allow us to express simple compliance requirements that do not require any process logic, e.g., a separation of duty constraint can easily be expressed as rule over the actors of two process activities. In general, such compliance constraints can be stated in a variety of rule language, such as RuleML [8], logic-based languages such as deontic logic [9], WS-Policy [7], and so on. In some cases it might also be feasible to use natural language to describe, for example, a problem domain for a process model. The exact formalism to be used to express compliance rules depends on the specific DSL adopted for each compliance concern.

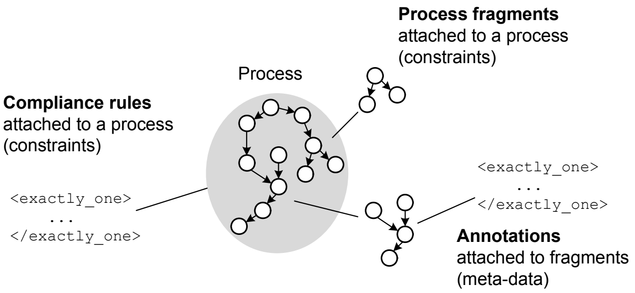


Figure 5 Annotating business processes with compliance rules and process fragments.

B. *Process fragments*

The second compliance annotation mechanism is the modeling of compliance constraints as process fragments, which allows us to describe desired behaviors in terms of control flows of a process model. A process fragment [2] is a connected sub graph of a process graph. It can also contain additional artifacts, such as variables, references to related processes, annotations, etc. Some parts of a fragment may be explicitly stated as *opaque*, in order to mark points of variability and degree of freedom for reuse. Therefore, a process fragment is not necessarily executable. We advocate the use of process fragments for two distinct purposes: (i) to express *compliance requirements* and (ii) to *reuse* existing, compliant parts of business processes.

Process fragments can be attached to a process, for example, to state that the process implements the functionality modeled by the fragment, but possibly in a different manner. That is, process fragments can be used to describe the desired behavior of a process. The information contained in the fragment can then be used for monitoring after according transformations. Fragments used for describing the behavior of a process can be annotated with textual constraints themselves, in order to express further characteristics of the fragment, as illustrated in Figure 5.

For the application of process fragments to perform compliance checks we can formalize annotations as follows: Let *PM* be a process model and the set $PF = \{pf1, pf2\}$ the association of two process fragments to this process model. The process fragments, in this case *pf1* and *pf2*, have themselves annotations represented by the set $PFA = \{a_{pf1}, a_{pf2}\}$. For example, a_{pf1} may state that the fragment *pf1 must* be executed in any case, i.e., it *must* appear in the execution history. The annotation of *pf2* (a_{pf2}) may state that this fragment *must not* be executed, i.e., that it *must not* appear in the execution history.

Those are two extremes; arbitrary levels in between *must* and *must not* are conceivable, but for the illustration of the concept those two levels are already sufficient. The main challenge in the usage of process fragments for monitoring purposes is their interpretation (along with their respective annotations) in terms of events registered during business execution and the conformance check of business execution with process fragments.

A process fragment can also be employed for the reuse of functionality (in our case business logic in form of Web service orchestrations), using common techniques from approaches for reuse, such as abstraction or parameterization. Yet, a process fragment is not necessarily self-contained, as it cannot be integrated into a process as "easily" as a subprocess [4]. Many customization steps are necessary when fragments shall be integrated into a process. Related works use the term "process merging problem" in this regard [5].

As for the specification of process fragments, initial work has been done in the context of BPEL 2.0 [4]: abstract processes have been specified for templating purposes and for modeling the visible behavior of processes that take part in a globally coordinated choreography.

V. Business process execution

Regarding process execution, events generated by the execution of services and processes are of fundamental importance for the monitoring of running processes and, hence, compliance governance. The execution of services and business processes creates events that provide information about the concrete execution of services and process instances. Generated events typically include the *type* of the event (e.g., deadline expiration event), its *source* (e.g., service A), its *destination* (e.g., service B), runtime information like *timestamps* when process steps are executed, and further possible properties. Events must carry enough data so that during compliance evaluation it is possible to reconstruct which service or process an event is associated with.

As show in Figure 3, events are generated by the Business process engine and by the instrumented services. The engine provides data about the current status of the execution of processes by emitting execution events that indicate steps within a running process instance (e.g., the start or end of an activity). The services that are part of the process and that are instrumented produce events related to the action that has been invoked on this service. Business process engine and services are configured by means of the code that is generated from the compliance requirements expressed in the DSLs described earlier.

Events are published via a central Enterprise Service Bus (ESB) [6] representing a unified communication channel between components. The ESB provides publish/subscribe support for events. The Event log and Protocol monitoring component consume events after subscribing to them. The Event log component is subscribed to any event type that is relevant for compliance evaluation; the Protocol monitoring component may also generate new events in response to the fulfillment or violation of a process fragment. Additional process execution data (e.g., data that is not carried with events, such as engine-internal logs) are stored in a dedicated Audit trail.

VI. Compliance evaluation and auditing

A. *Assessing compliance*

Evaluating the compliance state of a company means looking at how business has been performed, more than looking at how it has been designed. That is, we need to check whether the traces and evidence produced by the execution of processes and services conform to the compliance rules and process fragments defined. The two ways of expressing compliance requirements demand for two ways of checking compliance.

First, *process fragments* are evaluated during runtime by the Protocol monitoring component (see Figure 3). Fragments are interpreted as business protocols, i.e., in terms of visible messages exchanges among services. Business protocols offer automatic reasoning mechanisms with many applications, such as correctness verification, compatibility testing, monitoring, etc. The Protocol monitoring component verifies (i) whether a process instance is effectively executed according to its process model and (ii) whether processes that are annotated with a process fragment correctly adhere to the behavior described by the fragment. If a violation is detected, a violation event is published.

Second, *compliance rules* are evaluated during ETL time, when events and business data are loaded from the Event log and the Audit trail into the data warehouse. Rules are evaluated by looking at the parameters carried by the events. Rule and fragment violations are loaded into the warehouse for analysis.

B. *Warehousing compliance data*

In order to report on compliance, we maintain a Data warehouse that stores business process related events and compliance violations in a form that enables OLAP-style analysis. The main challenge in this context is the reconstruction of process, activity, and service instances from logged events, in order to provide the users of the compliance governance dashboard with data that are at the right level of abstraction.

C. *Mining protocols from event logs*

As mentioned above, protocol monitoring allows us to check the conformance of a set of events to a process model or a process fragment. However, in order to fully understand the behavior of an implemented system, we are also interested in identifying interaction patterns that are not specified at design time, yet occur frequently. Potential reasons for not defining all system behaviors at design time typically include lack of time, uncontrolled evolution, or, simply, unawareness of the behavior.

A solution is to infer protocols from the conversation logs. Direct applications are re-engineering issues, such as implementation correctness checking, service evolution, or monitoring. Discovering service protocols includes many technical challenges: cleaning logs from noise, identifying the different conversations, defining assessable models, developing and refining tools for an interactive extraction, etc. Algorithms dealing with specified constraint protocols extraction are required.

D. *Reporting on compliance*

Finally, starting from the Data warehouse, the Compliance governance dashboard provides visibility into compliance concerns, such as regulations, QoS requirements, and licensing concerns across related business functions. The dashboard allows users to navigate and analyze compliance together with performance (e.g., via KPIs) and quality of business processes. To enable users to quickly and comprehensively access this critical information, we implement a Web-based user interface to manage business activity data related to interacting processes. Graphical indicators and charts facilitate at-a-glance business monitoring, and reporting of real-time KPIs against historical trends, and users can perform in-depth analysis of business performance, assisted by the Business intelligence component. The output of the Compliance governance dashboard is meant for human consumption (e.g., for auditing purposes) by business or IT people.

VII. Related work

Most of the works in literature or products on the market focus on one or two aspects of the overall compliance problem, e.g., on design, on runtime, or on both. For instance, Accorsi [22] concentrates on the problem of identity management for privacy control. In [14] the authors extend the definition of a policy language (the Extended Privacy Definition Tool language, ExPDT) toward the formulation of generic compliance policies for business processes. The language enables the specification of compliance constraints of business processes with regulations and their validation without sacrificing business agility. In [20] the authors propose a semantics-based approach to the modeling of compliance concerns.

Similar to our approach, the authors of [12] propose a model-driven generation of rules for compliance monitoring of systems. Their work uses meta-models to define and formalize compliance regulations into formal policies. These policies are then used to drive the generation of the monitoring logic. The runtime monitoring is based on event streams that are analyzed by event-oriented middleware to identify violations of the compliance regulations. In [21], instead, the authors focus on leveraging standard database technology (instead of events) to guarantee compliance.

The work described in [13] proposes solution for ensuring compliance throughout a complete business process lifecycle, that is, both at design time and at runtime. The work proposes a formal language for specifying a subset of business rules, called semantic constraints, and the necessary mechanisms for parsing the constraints and ensuring compliance of process management systems at runtime.

Finally, there is a number of industry solutions [15][16] [17] that address the issue of business compliance. Typically, these solutions focus on the monitoring of compliance at runtime, where the targets of the monitoring are the IT assets of an organization, such as directory servers, file servers, desktops, databases, firewalls, and similar. Assets are monitored with respect to the state of system controls, for example, user privilege configurations that are put in place to ensure compliance to certain regulations. Most of these (and similar) solutions on the market have a strong focus on compliance with specific regulations, such as PCI DSS [18], ISO 27002, Basel II [19], SOX [1].

VIII. Discussion and future work

In this paper we described our infrastructure for compliance management in SOA environments, in which business processes are implemented as Web service compositions. The aim of the infrastructure is to cover the whole compliance management life cycle, covering design, execution, and assessment and evaluation. For the design of compliant business practices (i.e., processes, services, and policies) we propose a View-based Modeling Framework, which allows designers to separate the multiple concerns (e.g., the process logic, the event generation logic, compliance rules, and business process fragments) that characterize the compliance modeling scenario. To assist the execution and monitoring of processes and services we propose instrumenting them with event generation logic by automatically generating the necessary code from the designs. To evaluate compliance we propose three mechanisms, each addressing different reporting needs: online protocol monitoring for runtime assessment, analysis of execution data to report on compliance problems with specific process or service instances, and log mining to identify possible design flows. Special attention is given to the problem of how to tell business people, compliance experts, and auditors what the current compliance status of a company is, an aspect that has mostly been neglected so far.

The described system represents an ambitious goal, yet we are convinced that current technologies and business practices are mature enough to adequately support compliance governance via automated means. We are still at the beginning of our endeavor, but the ideas conceived and results achieved so far are promising, and the need for automated compliance governance support is self-evident.

Acknowledgment

This work was supported by funds from the European Commission (contract N° 215175 for the FP7-ICT-2007-1 project COMPAS).

References

[1] Congress of the United States. Public Company Accounting Reform and Investor Protection Act (Sarbanes-Oxley Act), 2002. Pub. L. No. 107-204, 116 Stat. 745.

[2] Z. Ma and F. Leymann. A Lifecycle Model for Using Process Fragment in Business Process Modeling. BPDMS'08.

[3] A. Alves et al. OASIS Web Services Business Process Execution Language Version 2.0. April 2007.

[4] M. Kloppmann et al. WS-BPEL Extension for Sub-processes – BPEL-SPE. White Paper, September 2007.

[5] J. Küster et al. Improving Business Process Models with Reference Models in Business-Driven Development. BPM'06.

[6] D. A. Chappell. Enterprise Service Bus: Theory in Practice. O'Reilly, 2004.

[7] S. Bajaj et al. Web Services Policy 1.2 - Framework (WS-Policy), W3C Member Submission, 25 April 2006.

[8] RuleML. The Rule Markup Initiative, 26th February 2005. http//www.ruleml.org

[9] J. Carmo and A.I.J. Jones. Deontic Logic and Contrary-to-Duties. In Handbook of Philosophical Logic, 2nd Ed. Volume, 8, 265-343, Kluwer, 2002.

[10] T. Stahl and M. Völter. Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

[11] H. Tran et al. View-based Integration of Process-driven SOA Models at Various Abstraction Levels. MBSDI'08, Berlin, Germany, pp. 55-66, Springer, 2008.

[12] C. Giblin et al. From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. IBM Research Report RZ 3662, Zurich, October, 2006.

[13] L. T. Ly et al. Compliance of Semantic Constraints - A Requirements Analysis for Process Management Systems. GRCIS'08, Montpellier, France.

[14] S. Sackmann and M. Kähmer. ExPDT: A Policy-based Approach for Automating Compliance. Wirtschaftsinformatik (WI), 50(5), pp. 366-374, 2008.

[15] D. J. Langin. Basel II compliance with Tripwire. Tripwire. February 2009 http://www.tripwire.com/europe/resources/white-papers/?tid=Basel II

[16] Application Security Inc. http://www.appsecinc.com/products/appdetective/

[17] IBM. IBM Rational AppScan: enhancing Web application security, March 2009. http://www-01.ibm.com/software/awdtools/appscan/

[18] PCI Security Standards Council. Payment Card Industry (PCI) Data Security Standard, Requirements and Security Assessment Procedures. V.1.2, October 2008. https://www. pcisecuritystandards.org/security_standards/pci_dss.shtml

[19] Basel Committee on Banking Supervision. Basel II: International Convergence of Capital Measurement and Capital Standards: a Revised Framework. , June 2004, http://www. bis.org/publ/bcbs107.htm

[20] K. Namiri and N. Stojanovic. A Semantic-based Approach for Compliance Management of Internal Controls in Business Processes. CAISE'07, pp. 61-64.

[21] R. Agrawal et al. Taming Compliance with Sarbanes-Oxley Internal Controls Using Database Technology. ICDE'06, pp. 92.

[22] R. Accorsi. Automated Privacy Audits to Complement the Notion of Control for Identity Management. IFIP Conference on Policies and Research in Identity Management, Springer, Berlin, pp. 39–48, 2008.