SOCA
DOI 10.1007/s11761-013-0129-3

ORIGINAL RESEARCH PAPER

SOA-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes

Carlos Rodríguez · Daniel Schleicher · Florian Daniel · Fabio Casati · Frank Leymann · Sebastian Wagner

Received: 24 February 2012 / Revised: 26 October 2012 / Accepted: 11 January 2013

Abstract Facilitating *compliance* management, that is, assisting a company's management in conforming to laws, regulations, standards, contracts, and policies, is a hot but non-trivial task. The service-oriented architecture (SOA) has evolved traditional, manual business practices into modern, service-based IT practices that ease part of the problem: the systematic definition and execution of business processes. This, in turn, facilitates the online monitoring of system behaviors and the enforcement of allowed behaviors—all ingredients that can be used to assist compliance management on the fly during process execution. In this paper, instead of focusing on monitoring and runtime enforcement of rules or constraints, we strive for an alternative approach to compliance management in SOAs that aims at *assessing and improving* compliance. We propose two ingredients: (i) a *model and tool* to design compliant service-based processes and to instrument them in order to generate evidence of how they are executed and (ii) a *reporting and analysis suite* to create awareness of a company's compliance state and to enable understanding *why* and *where* compliance violations have occurred. Together, these ingredients result in an approach that is close to how the real stakeholders—compliance experts and auditors—actually assess the state of compliance in practice and that is less intrusive than enforcing compliance.

Keywords Service-based compliance governance · Compliance assessment · Signaling instrumentation · Key indicators · Root cause analysis · Reporting dashboard

C. Rodríguez (✉) · F. Daniel · F. Casati
Department of Information Engineering and Computer Science, University of Trento, Via Sommarive 14, 38123 Povo, TN, Italy
e-mail: crodriguez@disi.unitn.it

F. Daniel
e-mail: daniel@disi.unitn.it

F. Casati
e-mail: casati@disi.unitn.it

D. Schleicher · F. Leymann · S. Wagner
Institute of Architecture of Application Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany
e-mail: daniel.schleicher@iaas.uni-stuttgart.de

F. Leymann
e-mail: frank.leymann@iaas.uni-stuttgart.de

S. Wagner
e-mail: sebastian.wagner@iaas.uni-stuttgart.de

1 Introduction

Compliance management [34] is an important, costly, and complex problem: It is *important* because there is increasing regulatory pressure on companies to meet a variety of requirements in terms of regulations, laws, and similar (e.g., Basel II, MiFID, SOX). This increase has been to a large extent fueled by high-profile bankruptcy cases (e.g., Parmalat, Enron, WorldCom), safety mishaps (the April 2009 earthquake in L'Aquila, Italy, has already led to stricter rules and certification procedures for buildings and construction companies), or the recent financial crisis. Failing to meet these requirements may imply safety risks, hefty penalties, loss of reputation, or even bankruptcy or jail [35].

Managing and auditing/certifying compliance is a very *expensive* endeavor. In their 2008–2009 Governance, Risk Management, and Compliance Spending Report [13], AMR Research estimated that companies would spend US$ 32B only on governance, compliance, and risk in 2008 and more than US$ 33B in 2009. In addition, audits are themselves expensive and invasive activities, costly not only in terms of auditors' salaries but also in terms of internal costs for preparing for and assisting the audit.

Published online: 02 February 2013

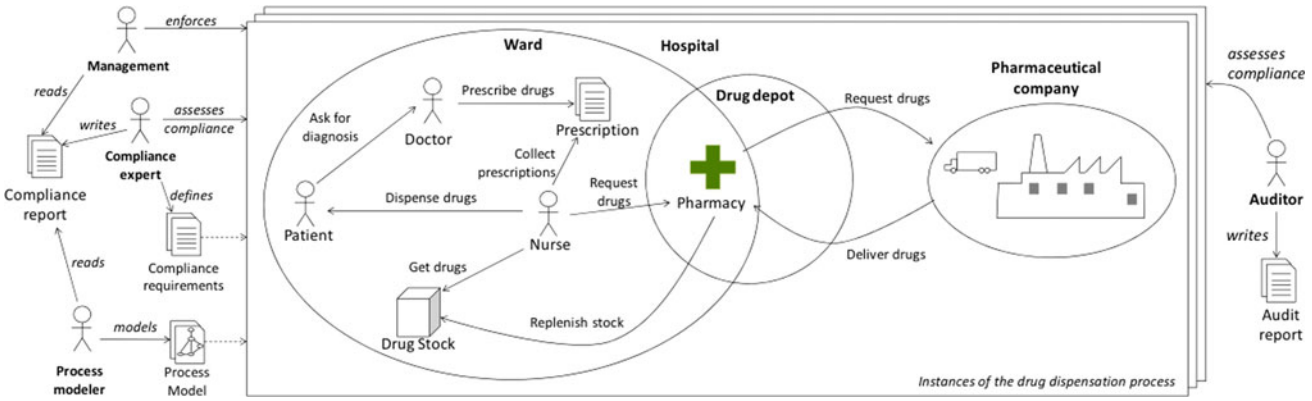

Fig. 1 Outpatient drug dispensation in a hospital: modeling compliance requirements and assessing compliance

Finally, the problem is *complex* because compliance requirements are often pervasive in that they span across many segments of a company and many processes. They are also sometimes only vague and informally specified. Yet, compliance management requires understanding and interpreting requirements and then implementing and managing a typically large number of controls on a variety of procedures across the business units of a company. Each compliance requirement and procedure may demand for its own control mechanism and its own set of assessment metrics to adequately capture the state of compliance.

Today, compliance is to a large extent managed by the various business units in rather ad hoc ways and with little or no IT support. As a result, today it is very hard for any CFO or CIO to answer the following questions: Which requirements does my company have to comply with? Which processes should obey which requirements? Which processes are following a given regulation? Where do violations occur? Which processes do we have under control? Even more, it is hard to do so from a perspective that not only satisfies the company but also the company's auditors, which is crucial as the auditors are the ones that certify the company's capabilities to control compliance.

Yet, business processes are indeed supported by IT. Technologies like web services and business process management systems have demonstrated, although more slowly than initially thought, their viability for organizing work and assisting and orchestrating also human actors involved in business processes. Interestingly, however, the automated operation of business processes has not yet lead to a significant facilitation of compliance management practices.

1.1 Reference scenario: outpatient drug dispensation in a hospital

Let us consider, for example, the management and assessment of the outpatient *drug dispensation* process summarized in Fig. 1. The process—and this paper—originates in the EU project MASTER (http://www.master-fp7.eu) where we cooperate on compliance management with Hospital San Raffaele (Milano, Italy), which runs the described distributed business process. The process is part of a bigger procedure known as the *outpatient drug reimbursement*, which implements the steps required for refunding hospitals for the drugs dispensed to patients that are not hospitalized. The overall process is regulated by the Italian Healthcare Authority, which dictates regulations on the dispensation and reporting requirements for the reimbursement of drug expenses, such as the ones concerning privacy in personal information processing.

The core process, shown in Fig. 1, starts with the patient's visit to the doctor in the hospital's ward. Depending on the diagnosis, the doctor sends a prescription for drugs to the nurse, who dispenses the necessary drugs to the patient if the requested quantity is available. If the available drug quantity is insufficient, she requests the drug to the hospital-internal pharmacy, which is then in charge of replenishing the nurse's drug store. If, in turn, the pharmacy is running out of stock, it orders the necessary drugs from the pharmaceutical company.

The drug dispensation process is supported by several web service-based information systems that interact inside a SOA that is distributed over the hospital, the pharmacy, and the pharmaceutical company. For instance, there are web services for issuing drug requests in the various dependencies of the institute, and the pharmaceutical company the hospital cooperates with accepts drug requests through web service interfaces. To retrieve the data necessary to assess the hospital's state of compliance, during the execution of the process, suitable data (e.g., events) that can be logged and later analyzed are produced by all cooperating parties.

By law, the hospital must guarantee that all *patient data are anonymized* throughout the process (and in the generated events), and the hospital's internal policy states that *drug replenishment must occur within maximum two business days* and that the person who prescribes a drug cannot

also dispense the drug (*separation of duties*). While this latter requirement is monitored internally by the hospital's own compliance expert, the former requirement is subject to yearly audit by an official security *auditor*, who certifies (or not) the hospital's compliance with the laws the hospital is subject to. Passing this audit is crucial for the hospital's business continuity.

The compliance requirements that apply to the hospital are identified and specified by the *compliance expert*, who knows about the applicable laws and regulations and about the internal policy. Typically, the compliance expert assists the *process modeler* in designing compliant processes, in order to prevent non-compliance by design. Yet, he also checks the execution of the designed processes, as at runtime, non-compliant situations may occur despite a well-designed process model (e.g., due to system failures or manual intervention on a running process instance). Periodically, he then writes an internal compliance report, which is the basis (i) for the *management* to take decisions and enforce compliance and (ii) for the *process modeler* to understand violations and improve process models and controls.

Today, all these activities are typically performed *manually*, and compliance assessment is of *statistical* nature. That is, controls are added to processes in an ad hoc and per-process fashion; process instances are checked by inspecting only a subset of physical documents or log files and estimating compliance levels; the compliance report is written by hand; and analyzing the root causes of violations is hard and time-consuming, given the large amount of data to be correlated. In addition to that, although the overall process is automatically orchestrated and activities have suitable IT support, in practice many tasks are still based on paper forms filled by doctors or nurses during their service and manually input only in a later stage. [1]

1.2 Contributions and structure of the paper

This paper describes an infrastructure and methodology that supports compliance management. Specifically, we provide the following contributions:

- We provide ***a model and a graphical modeling tool*** that eases building processes that are compliant with process-specific compliance requirements. The approach allows one to equip a common business process definition (e.g., BPEL process specification) with a definition of technical *compliance rules* and to *instrument* it in order to generate the necessary evidence for compliance assessment.
- In order to facilitate compliance assessment, we ***extend a state-of-the-art service orchestration engine*** with signaling capabilities that are able to generate compliance-related evidence on process executions.
- We provide a ***suite of reporting and analysis tools*** that facilitates the writing of the compliance report and helps the compliance expert and the process modeler to identify where and why compliance violations happened. The suite is based on a *compliance-oriented warehouse*, *key compliance indicators*, and *root cause analysis algorithms*.
- We implement all reporting and analysis algorithms on top of a ***data model that supports compliance assessment***, which allows us to better reflect the *nature* of the data that are available for analysis and to enable *better business decisions*.

In summary, the main goal of this work is to enable humans to be aware of how compliant business processes are and to understand why problems happen, in order to improve compliance. We propose a methodology and tool for the definition and assessment of compliance rules. While compliance rules are typically domain-specific, our solution is generic and aims to support different regulations at a technical level, limited to those business processes of a company that are supported by web services and that are executed with the help of a business process engine.

The methodologies, prototypes, and demos described in this work have been designed and evaluated with the help from audit experts of Deloitte, Paris, who deal with compliance in a variety of domains at a daily basis.

In the next section, we introduce our approach to compliance management and show how the above contributions fit into an overall methodology. Then, in Sects. 3, 4, 5, and 6, we describe the individual phases of the methodology, that is, (i) design of processes and evidence, (ii) execution of processes and generation of evidence, (iii) assessment of compliance, and (iv) analysis of problems. In Sect. 7, we survey the most related works, and in Sect. 8, we recap the contributions of the paper.

2 Compliance management in the SOA

2.1 Compliance management requirements

Compliance management should enable the company's *management* to know about the state of compliance, assess the risks associated with non-compliant situations, and take business decisions to correct them. Ideally, these decisions are

[1] It is important to note here that we assume all the artifacts needed for compliance management are represented inside the information system. Also, we do not deal with the problem of fidelity regarding the computer representations of real-world artifacts; this is a general modeling that requires adequate domain and modeling knowledge.

based on up-to-date compliance reports, featuring a set of compliance-specific indicators that are easy to interpret and, hence, effective in communicating key information. The *compliance expert*, instead, is interested in knowing the individual compliance violations and understanding their causes, while the *process modeler* is rather interested in how to improve process models as well as control points for future executions.

Typically, this means that we need a dashboard for reporting on compliance that allows navigation across the company's processes and across compliance concerns and associated key indicators at different levels of aggregation and details. It also means that we need to provide a way to model concerns and indicators and to collect evidence for their computation. In terms of modeling, we need a ***formalism and tool to express compliant behaviors***, for example, in the form of process templates that specify compliance requirements and constrain the instantiation of the template. Once we have a definition of a compliant process trace, we can then verify if the actual execution is compliant. We also need a way to define and compute indicators which can typically be based on aggregating information over many process instances (e.g., the total amount of invoices that were handled in a non-compliant manner).

In this paper, we consider as evidence of compliance and as source for the computation of reports the information and events related to the execution of processes. Some of these data and events (e.g., the start of an activity) are commonly produced by business process engines during runtime, but compliance assessment may ask for some specific execution evidence (e.g., a login event with actor information, or information about an invoice). Collecting proper evidence, typically within a data warehouse, requires the ***instrumentation*** of a process or service orchestration engine as well as a way to specify which events should be signaled by the process.

While processes, related evidence, compliance requirements, and indicators differ on a case-by-case basis, the challenge here is to adopt the same formalisms and computation model; otherwise, the approach is not reusable and we would have to develop models and code for each new compliance requirement or new process.

In the case compliance violations have happened, it is of utmost importance to be able to ***understand why and where*** these violations occurred. Violations may stem from problems during process execution (*instance-level* violations) or from badly designed processes (*model-level* violations). To understand instance-level violations, we propose the use of classification by means of decision trees, which allows the correlation of a process instance's compliance state with its business data. In order to understand model-level violations, we propose the use of *protocol discovery*, which allows the comparison of a system's real behavior with its designed behavior. Finally, both instance-level and model-level violations manifest themselves also in compliance indicators. Correlating their values and dynamics over time may thus provide further indications on where violations occurred in a process model and which violations impact on which other violations.

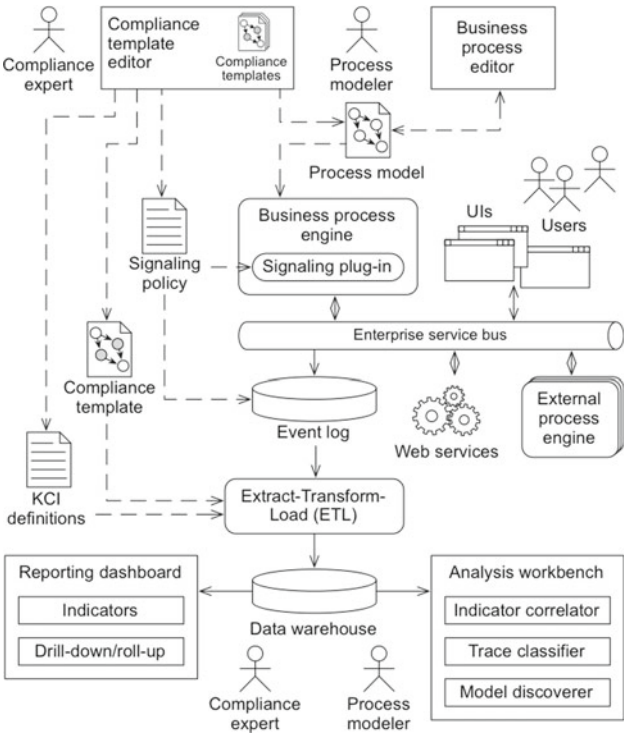

Fig. 2 Event-based compliance management architecture

2.2 System architecture

Figure 2 illustrates how we approach the above requirements from a system architecture perspective. The architecture has been designed leveraging on events as concrete evidence of the runtime behavior of the system, where the necessary events can be either derived for free from service communications in the service-based environment or they can be obtained by instrumenting the system purposefully. Starting from business and compliance requirements, the compliance expert defines *compliance templates* (see Sect. 3.1) and *Key Compliance Indicators* (KCIs), that is, indicators measuring the compliance of process instances (Sect. 3.2), with the help of a dedicated *compliance template editor*. A compliance template describes the compliant behavior of a business process, while the KCIs are key indicators that measure how compliant a company is with respect to its compliance requirements. Based on the compliance template and the specified KCIs, a so-called *signaling policy* is created, which states which execution events are needed to assess compliance.

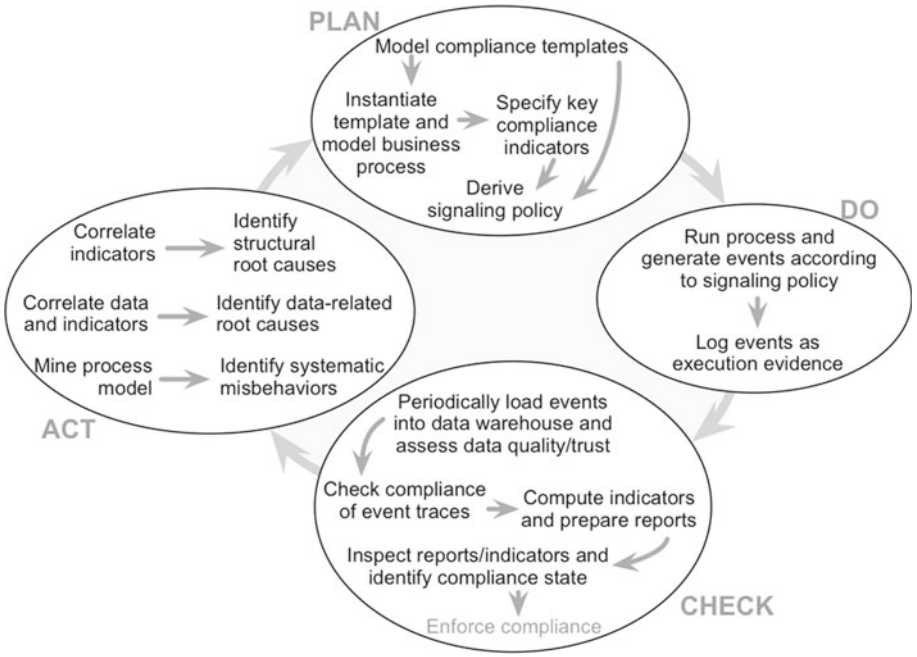

Fig. 3 Assisted compliance management methodology

The process modeler instantiates the templates, designs the *process models*, and generates executable process specifications (in our case, we generate BPEL), which can be inspected and fine-tuned with the help of a common *business process editor*. The engine takes process models as input and instantiates and runs them, establishing this way a communication between *web services*, *human users* (via dedicated user interfaces that allow them to interact with the process), and possible *external business process engines* (in the case of distributed business processes). The *signaling policy* configures the *signaling plug-in* of the purposely extended *business process engine*. Communications and events are sent over a shared *enterprise service bus* (ESB), which allows the easy tracking of events in an *event log*. Out of all the messages that flow through the ESB, the event log only subscribes to the events defined in the signaling policy. Periodically (e.g., during the night), an ETL (*Extract-Transform-Load*) procedure loads tracked events into the *data warehouse* and computes compliance and KCIs. The data in the warehouse can then be inspected by the compliance expert in a *reporting dashboard* that visualizes indicators and supports the necessary drill-down (navigation to finer-grained details) and roll-up (aggregation) features. An *analysis workbench* provides for the analysis of compliance violations.

2.3 Compliance management methodology

Compliance management is *not* a simple issue, a property that manifests itself also in the complexity of the proposed system architecture (see Fig. 2). Compliance management typically requires understanding multiple sources for regulatory compliance requirements (e.g., laws, standards, or similar) and to translate the requirements that affect a given process into technical rules. We aim at supporting this latter aspect, which translates into the architecture and instruments in Fig. 2. For a better understanding of how the joint use of these instruments can aid compliance management, we contextualize them in our ***assisted compliance management methodology***, which is based on the *Deming cycle* [38], known from business process improvement. The methodology consists of four phases, which we illustrate in Fig. 3.

In the ***Plan*** phase, first we model a compliance template, which can then be instantiated into a process model. Given a process model, it is possible to specify which KCIs to compute for the process. Given the compliance template and the KCI definitions, the necessary signaling policy can be generated automatically. In the ***Do*** phase, processes and the signaling policy are executed, that is, processes are instantiated and run by the process engine, and specified events are generated and logged for later inspection. In the ***Check*** phase, the system periodically loads logged events into the data warehouse and labels event traces, that is, process instances, as compliant or not. The so-prepared data are used to compute indicators and to prepare the reports, which can then be inspected in order to understand the compliance state of the company. Depending on the encountered compliance violations, the management may enforce compliance (this step is not assisted by our system). Finally, in the ***Act*** phase, the compliance expert and process modeler try to understand the root causes of violations, so as to improve processes and policies by refining the respective models and specifications and, hence, restarting from the Plan phase.

In this paper, we do ***not*** propose the use of automatic techniques for the *runtime enforcement* of compliant behaviors in business processes. While such techniques undoubtedly allow a company to better control compliance requirements at a technical and operative level (e.g., at the level of individual events), compliance management is, however, still an organizational and tactical activity that most of the times requires human intervention and interpretation. The main goal of this work is therefore enabling *humans* to be aware of *how compliant* business processes are and to understand *why* problems happen, in order to incrementally improve compliance.

3 Plan: designing compliant processes and defining evidence

For the purpose of designing compliant business processes, we complement traditional process modeling with three ingredients: (i) *compliance templates*, which define the compliance requirements of a process; (ii) a *signaling policy*, which specifies which events need to be generated, and (iii) a set of *KCIs*, which summarize events for reporting purposes.

3.1 Specifying compliant behaviors

To describe the compliant behavior a process should follow, we propose to use compliance templates. By using compliance templates, we can, for example, define the acceptable order in which activities should be performed, which activities are allowed, and which constraints exist among them [30]. This approach has multiple benefits. First, compliance requirements that apply to a class of process models can be defined by individuals that are knowledgeable in their respective compliance domain, that is, the compliance experts (typically members of the management or lawyers). Compliance experts are responsible for the compliance templates; they are the only ones that are allowed to authorize changes on them. Compliance experts are supported by business process experts when a compliance template must be changed, for example. Second, because templates, as the word denotes, are used as a starting point for defining the process itself by expanding and detailing them, following regulations are made easier during design time. In other words, templates are not only a compliance constraint, but also an aid to (compliant) process modeling. Finally, compliance templates can be stored (e.g., in a central repository) and reused in a variety of similar process models.

A ***compliance template*** comprises three parts, namely an abstract business process, a compliance descriptor, and a variability descriptor.

The ***abstract business process*** defines the *compliant behavior* of a process in terms of its control flow and of allowed activities. It is called "abstract" because it lacks the necessary implementation details to be instantiated and run in a process engine. Only activities labeled *constrained region* can be customized by the process modeler in order to get an executable business process. Customizing a constrained region means inserting activities into it. Process modelers cannot change activities or control flows originally included in a compliance template, as this might lead to non-compliant processes.

As an example, refer to Fig. 4, which shows an abstract process model (in the center of the figure) of the drug dispensation process sketched in Fig. 1. We use a pseudo language in Fig. 4 to specify the abstract process for reasons of simplicity. Any other process specification language may have been used to define this abstract process, because of the flexibility of the compliance template approach. The abstract process expresses a number of compliance constraints: Activity *Prescribe Drug* must always be executed before activity *Collect Prescriptions*; or, after the activity *Collect Prescriptions* has been executed, the activities *Get Drugs* or *Request Drugs* can be executed. The separation of duties requirement for the *Prescribe Drug* and *Dispense Drugs* activities can also be expressed as compliance rules and associated with the respective web services, which must provide for the generation of the necessary evidence (the events) to assess the rules.

The ***compliance descriptor***, at the left of the abstract process model, allows the definition of *constraints* for the constrained regions. Compliance descriptors can be defined independently of the abstract process, and a single compliance descriptor can be reused in more than one abstract process. The dashed arrow pointing from one compliance assurance rule (*Link to Constrained Region*) in Fig. 4 to a constrained region shows which compliance assurance rule is applied to the first constrained region. Rules are expressed in first-order logic. We chose first-order logic because the class of compliance rules used with compliance templates deals with presence and absence of activities within a constrained region. These kinds of compliance rules can be expressed at best using the negation operator in front of a predicate to describe the absence of activities. Predicates without preceding operand are used to describe the presence of activities. The name of the used predicate maps to the name of the activity. One example for such a compliance rule is *activity A and activity B must always be inserted together*. With first-order logic, the example compliance rule above can easily be expressed as $A \wedge B$.

Compliance rules are evaluated at design time (in our graphical process development tool) every time the process modeler inserts an activity into a constrained region. The graphical tool notifies the process designer about which modifications are allowed and which modifications violate the implicit compliance of the abstract process. For example, an invocation of the *Pharmacy* web service in the first constrained region in Fig. 4 would violate the compliance of the

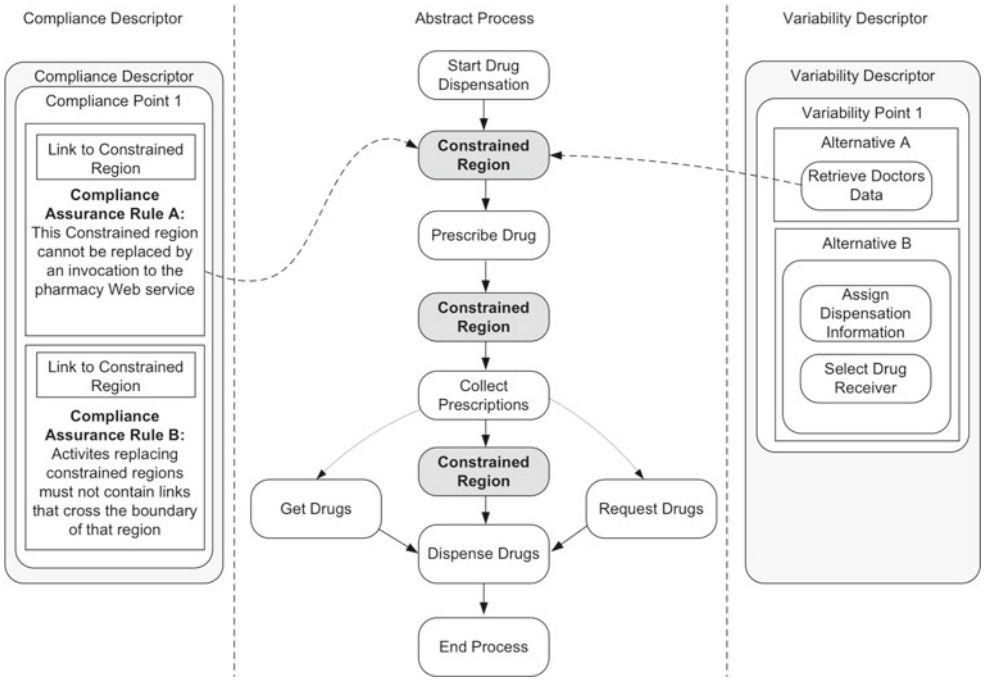

Fig. 4 Compliance template for drug dispensations

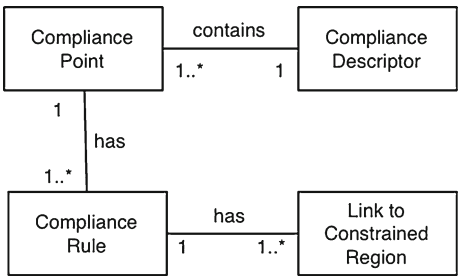

Fig. 5 UML meta model of a compliance descriptor

abstract process, because the activities *Prescribe Drug* and *Collect Prescriptions* would not yet have been executed.

The meta model of a compliance descriptor is shown in Fig. 5. A compliance descriptor contains one or more compliance points comprising compliance rules. These compliance rules can be linked to the constrained regions in the abstract process of a compliance template.

The ***variability descriptor***, at the right of the abstract process model in Fig. 4, contains variabilities that can be used to fill the constrained regions of the abstract process. The dashed arrow shows which variability descriptor is associated with which constrained region. A variability descriptor assists the process modeler by providing him/her with the *set of allowed activities* that can be used inside each constrained region; activities can again be compliance templates containing constrained regions. For instance, we have used *Alternative A* in Fig. 4 in the design of the compliant BPEL process. The activities in *Alternative B* are used in other constrained regions. Here, it is, for example, important that the compliance expert only allows services (activities) in the variability descriptor that natively encrypt the data they exchange with other services, in order to provide for the anonymization of patients' data.

Compliance templates can be designed for robustness or for reusability. *Robustness* is achieved by adding detailed, domain-specific constraints that guide the process modeler through an only narrow scope of action during the instantiation of a compliance template. *Reusability* is achieved by keeping the template more general. It is up to the compliance expert to decide what is more important to him/her. In fact, while our approach facilitates the expression of compliance rules, it is still important for the human expert to have the right regulatory and domain knowledge in order to correctly interpret the company's compliance requirements and express them in terms of compliance rules.

3.1.1 Modeling compliance templates and processes

In order to assist the compliance expert in defining compliance templates, we have extended the Oryx[2] BPMN editor. Oryx is a web-based BPMN editor that fully runs inside a web browser and does not require the installation of any additional software. Figure 6 shows a screen shot of Oryx at work. It mainly consists of three parts: the shape repository (labeled 1), the modeling canvas (labeled 3), and a pane on

[2] http://code.google.com/p/oryx-editor/.

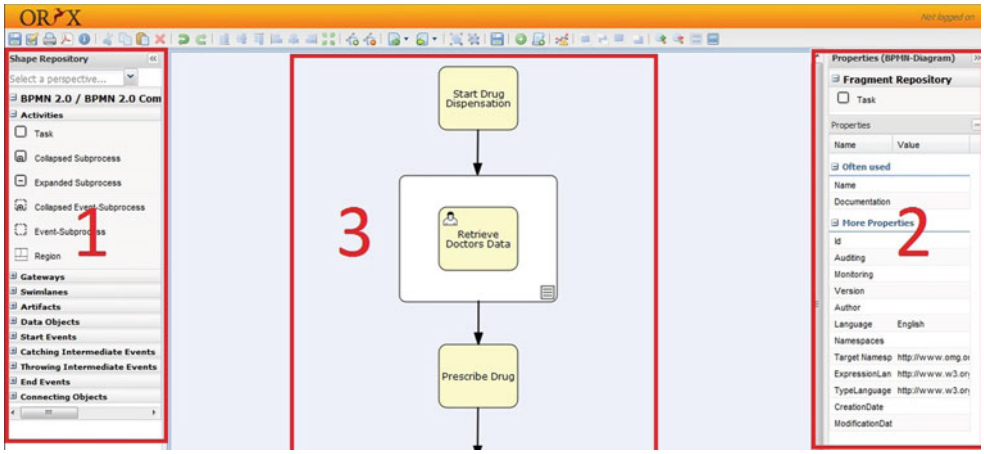

Fig. 6 Oryx BPMN editor for compliance templates

the right-hand side (labeled 2 and called Fragment Repository) containing the activities that compose the variability descriptor and a properties section.

To be able to create compliance templates, we added a new activity type named *Region*. In Fig. 6, the region activity type is shown in the shape repository and on the modeling canvas containing the task named Retrieve Doctors Data. To implement the compliance descriptor described above, we added a property named Compliance Descriptor to the region activity type. The Fragment Repository on the right implements the concepts of a variability descriptor as described before. Another addition we made is a compliance checker plug-in. This plug-in is used to check whether an activity inserted into a constrained region violates a constraint or not. The resulting BPMN 2.0 process model is transformed into a BPEL model that includes the mandatory activities imposed by the compliance template as highlighted in Fig. 7.

3.1.2 Creating the signaling policy

To measure the compliance of a process, evidence on process execution must be generated, to be able to certify which activities have been executed by a given process instance and which have been skipped or which have generated errors. This evidence is represented by *execution events*, which provide insight into the status of the process instance at the time of their generation. As we want to check compliance with the abstract process of a compliance template, it implicitly defines the minimum set of events (or the event traces) that characterize a compliant process instance. In order to check compliance, it therefore suffices to generate suitable *Start* and *End* events for each mandatory activity in the abstract process. Other execution events may be needed for computing indicators, for example, if an indicator is to be computed over non-mandatory activities.

The exact set of events is specified in the so-called *signaling policy*, that is, the policy that tells the business process engine which events must be generated during process execution. The necessary events that need to be produced in order to check the compliance of the designed business process can be chosen by compliance experts. A signaling policy can then be created with this information. In addition, the compliance expert can add properties to activities that hold any form of custom compliance policy beyond what can be expressed via the template. These are also checked in the assessment phase, discussed next.

3.2 Specifying key compliance indicators

Business performance is commonly measured by means of key indicators, typically key performance indicators [20], which are metrics that summarize in a single number how well predefined business goals are being achieved. Similarly, we advocate the use of KCIs to measure *how compliant* a company is with its compliance requirements and to *better target* the company's efforts to check and improve compliance, lowering the overall complexity of compliance management.

KCIs can be computed out of the evidence collected from process executions. Given the huge quantity of available events and runtime data that are typically available for each single process instance, this can, however, be a very complex task both from the perspective of metaphors and languages for defining such indicators and from the perspective of performance.

We approach both issues by providing the compliance expert with a so-called ***process instance table*** for defining and computing indicators. This is an abstract table that is specific to a given process model and contains one row per executed process instance. The attributes of the table are those process data items that the compliance expert needs for the

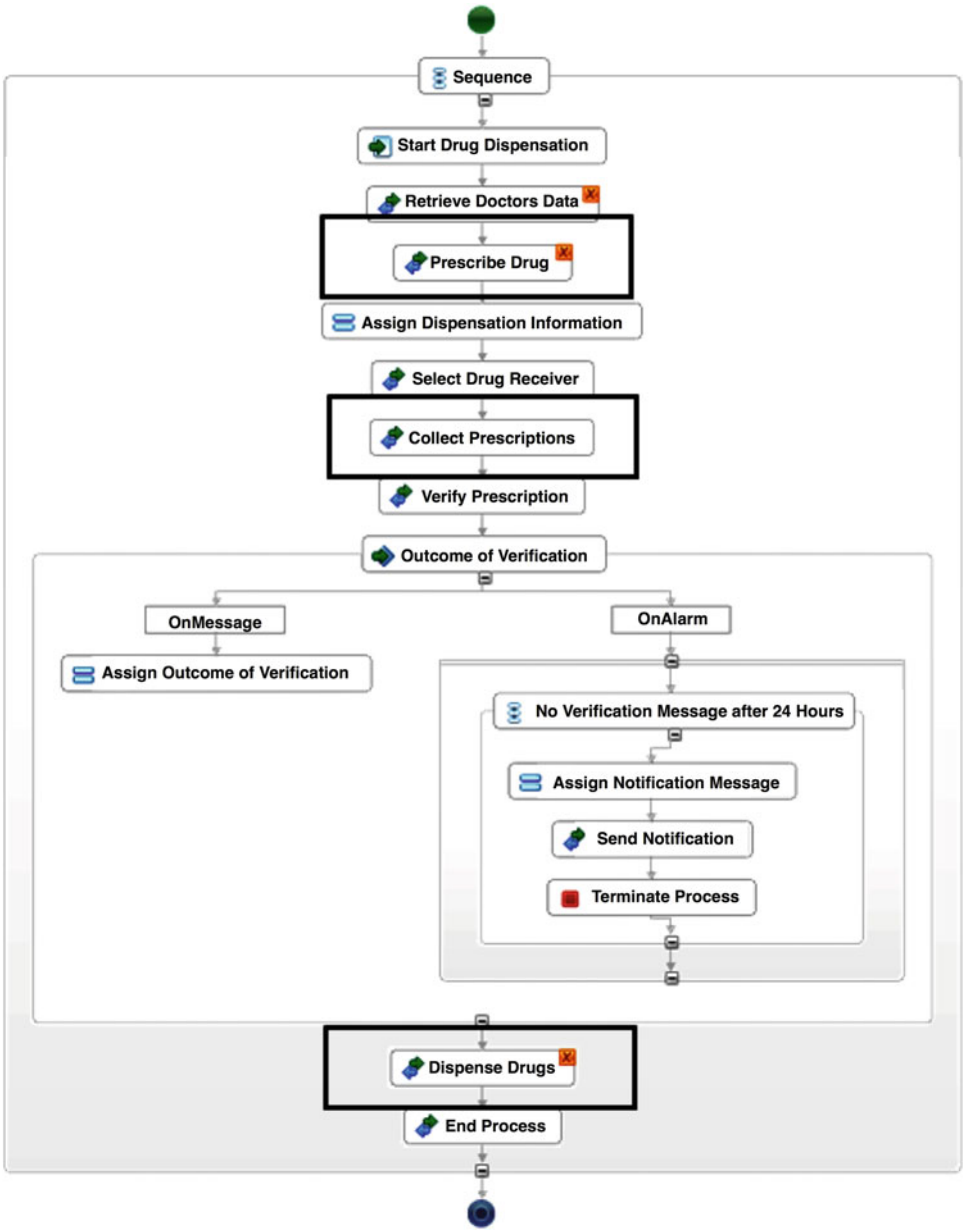

Fig. 7 Executable drug dispensation process (for presentation purpose, we omit data assignments)

definition of indicators, plus one or more Boolean attributes for each template to which the model must be compliant (if more than one template apply), reflecting the compliance requirements the process should satisfy. The values of the data items are carried by the events generated at runtime, while all necessary events are specified in the signaling policy and are either derived automatically from compliance templates or manually defined (if they are not yet part of the template). We will see in the assessment section how process instance tables are implemented and populated.

Given a process instance table, an indicator can now be defined as regular mathematical expression over the attributes and rows of the table (on paper by the compliance expert), and it can be implemented via standard SQL queries (by the process modeler). Although indicators typically come in the form of percentage values, averages, sums, or similar, the process instance table abstraction allows us to support the full expressive power of SQL in the computation of indicators. SQL has been designed also as a language for computing aggregates and is well known, understood, and supported, so there was no reason to come up with another language.

Table 1 shows, for instance, an excerpt from the process instance table of the drug dispensation process. The columns *TimeRequest* and *TimeReplenish* represent the time at which a

Table 1 Excerpt of the process instance table for the drug dispensation process

PID	*TimeRequest*	*TimeReplenish*	*ReqWard*	*PendPresc*	*WrongDisp*	*Compliant*
72665	13-05-10 22:32	14-05-10 16:45	W5	1	2	True
72666	13-05-10 22:39	15-05-10 12:55	W3	3	1	False
72667	13-05-10 22:55	14-05-10 08:59	W3	3	1	True
72668	13-05-10 23:01	14-05-10 23:33	W7	25	4	False
72669	13-05-10 23:49	14-05-10 02:57	W6	2	0	True
…	…	…	…	…	…	…

drug request was issued and the time at which the request was fulfilled, respectively, while *PendPresc* and *WrongDisp* tell us the number of pending patient prescriptions and the number of wrong dispensations of drugs by the pharmacy (e.g., with a wrong drug type of quantity). Notice that the number of wrong dispensations is computed when loading the data warehouse, and it can be done, for example, by checking the records on the complaints from patients about wrong dispensations. Finally, the column *ReqWard* represents the identification of the ward that issued the request (we omit the other attributes). In the table, we assume that the process should only follow one template, so there is only one compliance column.

Having in mind the structure of Table 1, the compliance expert can now, for instance, specify an indicator $KCI_{CompInst}$ to monitor compliance with the process compliance template:

$$KCI_{CompInst} = \frac{|CompliantInstances|}{|AllInstances|}$$

The process modeler expresses the formula then as follows (we only show a simplified query, e.g., without time intervals; for more details, please refer to [21]):

```
count_compliant_inst =
 select count(Compliant)
 from drug_dispensation_instance_table
 where Compliant = true;

count_all_inst =
 select count(Compliant)
 from drug_dispensation_instance_table;

KCI_CompInst =
 count_compliant_inst / count_all_inst;
```

The formula presented above is stored in the data warehouse together with the definition of the indicator, from where the ETL procedure can retrieve periodically for the computation of indicators.

The specification and computation of the indicator presented in this example is rather trivial. The real challenges reside (i) in identifying which are the most effective indicators (and events) and (ii) in the transformation and correlation of raw events in order to create the process instance tables. In fact, the ease with which we specified and computed the above indicator is a consequence of this data preparation and one of the most important benefits of making this data arrangement.

Although the above examples associate KCIs with individual business processes, it is important to note that we can also have KCIs that measure properties of *multiple related processes* (e.g., a process and its sub-processes). Such kind of advanced KCIs can easily be specified by defining the indicator function over the join of the respective process instances tables, practically enabling the definition of arbitrarily complex indicators.

4 Do: running processes and generating evidence

Once business processes have been implemented according to their compliance templates and the signaling policy has been completed, processes can be executed and evidence can be collected. In case the process is implemented in BPEL, we also provide support to execute and most importantly to collect evidence (we support BPEL as it is a common situation; in case of ad hoc languages and infrastructures, we expect probes to be developed to generate the necessary events). We have chosen to extend the Apache ODE (http://ode.apache.org/) engine, although any other engine could be extended similarly.

Apache ODE is equipped with a mechanism to issue events at certain state changes of a BPEL process during execution. These events are saved in an internal database, the audit-trail. The audit-trail can be queried via a web service interface to check the execution traces (the sequence of generated internal events) of processes that have been executed and of processes that are still in the executing phase.

A drawback of this mechanism is that the audit-trail saves all events generated during process execution. In most cases, a third party is only interested in a subset of events, for example, events indicating that the process took a certain branch.

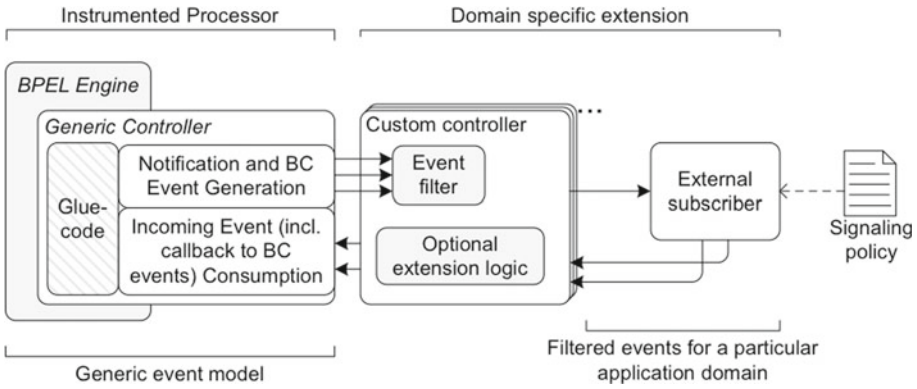

Fig. 8 Architectural overview of the core components of the signaling extension of Apache ODE [15]

Thus, these particular events must be separated from the rest of the events in the audit-trail, which is not always an easy task. Also, if we think of distributed business processes with multiple cooperating parties (such as our reference scenario), for security reasons it is typically not possible to query a partner's audit-trail. This is a major limitation for the assessment of the compliance of processes whose execution is distributed over multiple parties. To address these problems, we extended the Apache ODE BPEL engine to emit events to external subscribers, where the set of events and the allowed subscribers can easily be configured (e.g., by means of the signaling policy) [37].

Figure 8 gives a schematic overview of the ***extensions*** made to ODE. On the left side, the instrumented BPEL engine is shown. We extended the BPEL engine with a so-called *generic controller*. It comprises the glue code connecting the process navigation parts of the BPEL engine to the event handling part in the generic controller. At certain points in this execution logic, we throw events that are sent to one or more pluggable custom controllers, which correspond to the domain-specific part of the signaling architecture (this part corresponds to the *Signaling plug-in* introduced in Fig. 2). External stakeholders can write custom controllers to meet the requirements of their particular domain. All events occurring during the execution of a BPEL process are sent to every registered custom controller. In each custom controller, incoming events can be filtered and transformed. These filtered events can then be provided to external subscribers. The external subscribers can configure the filtering logic of the custom controllers. In our case, we use an external controller to parse the signaling policy and to instruct the custom controller to generate only those events that are required to assess compliance.

The ***signaling policy*** contains XPath expressions that point to the activity elements in a BPEL file, which is written in XML. We extended these XPath expressions with event indicators, since each BPEL activity may issue a number of different events. The XPath expressions in a signaling policy thus indicate which events of which activity need to be issued.

The underlying concept of the event subscription is resource-centric. We map process models, process instances, and activities deployed on a BPEL engine to resources and provide a suitable management API that allows one to access the resources. The API is exposed via web service interfaces.

Notice that the approach we present here is for offline assessment of compliance. This means that we log events that will only later be used for compliance assessment (e.g., during night hours). The performance issues for the generation of evidence regard more to the logging of event rather than the actual compliance assessment. Since logging performance is not the focus of this paper, and state-of-the-art logging systems are capable of handling this issue very well, we do not discuss this concern further.

5 Check: assessing compliance

From an IT perspective, assessing compliance means developing an assessment engine that "executes" the specification discussed in Sect. 3 over the event log, which constitutes our "evidence". Specifically, the engine should *verify* that process execution conforms to the different process templates and *compute compliance indicators*. The challenge lies in how to do this in a way that minimizes the development work needed for each new process, new template, or new indicator; otherwise, the system will not be easily maintainable. Given that changes are frequent (especially in regulations), this is an important aspect.

To compute ***conformance with templates***, we leverage on raw events. Although events in different processes may have different formats—as the process-specific data differ from process to process—what matters for verifying conformance is the process-independent part of events, that is, their type (Start or End), the activity that generates them, the process and instance in the context of which they are generated, and the occurrence time.

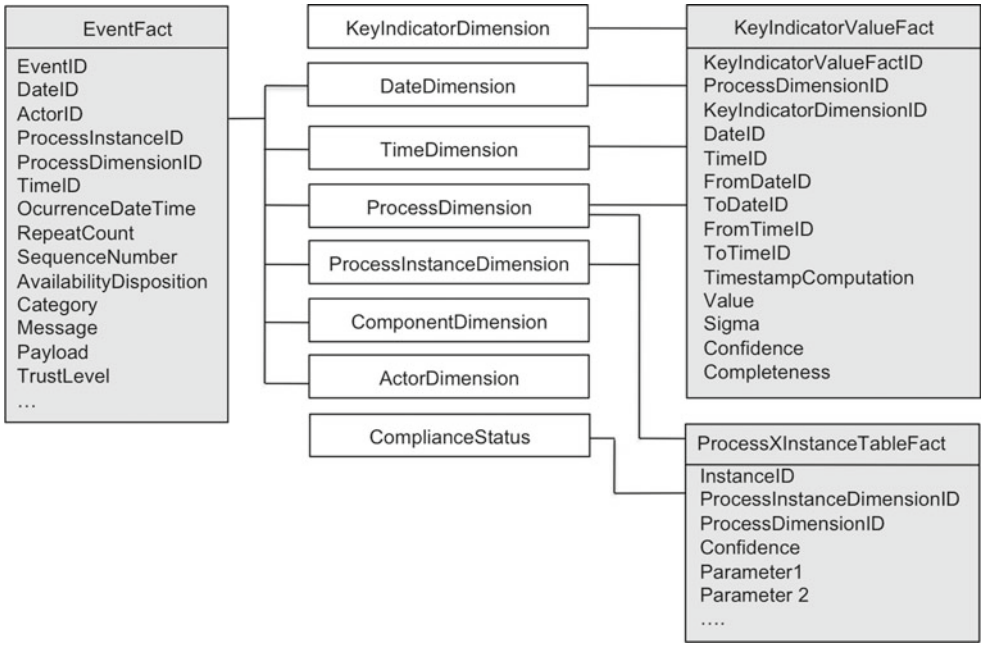

Fig. 9 Excerpt of the DW model; fact tables are shaded in *gray*, dimension tables are *white*

Reasoning in terms of language theory, process models are analogous to grammars and event traces are analogous to language strings. Therefore, computing whether a trace of events as described above conforms to a process model becomes analogous to checking if a string can be generated by a grammar. This is a well-known problem, and therefore, we do not discuss it further. The output of this procedure consists thus in giving a set of yes/no "labels" to each instance, one for each process template that was associated to the process model of that instance. Developing the conformance algorithm does not require any process-specific logic, which means that no new code is needed each time a new process or template is defined.

The case is different for computing ***indicators***. The metaphor we adopt for the indicator language implies that indicators can be arbitrarily complex queries over a dataset of process execution data with compliance information. This aspect combined with the needs of providing efficient navigation and drill-down/roll-up (i.e., navigation through the dimension and fact tables of the data warehouse) over a complex dataset as well as the need for a more sophisticated root cause analysis suggests that a sensible approach to leverage is that of building a data warehouse of process data, oriented at computing and assessing compliance and key compliance indicators.

Figure 9 shows an excerpt of our ***dimensional data model***. In the model, described in detail in [21], the facts are essentially the events and the process instances, while dimensions are process models, activities, and actors. Because different processes have different data items, instances of different processes are stored in different fact tables, where the attributes correspond to those process variables that are considered useful for compliance analysis and for computing indicators. These constitute the physical representation of the process instance tables discussed in Sect. 3.

An alternative approach would have been that of storing all process instance data *vertically*, where each tuple contains instance ID, variable name, and value. The benefit of this approach is that the warehouse schema does not change when new processes are defined. However, writing queries over vertical tables is more difficult and performance is lower, especially due to the high number of self-joins necessary.

The main data source of the warehouse is the event log. From there, the ETL procedure determines how to fill the process instance tables, based on *mapping specifications* done by the *compliance template editor* or the process modeler. In essence, these mappings specify from which event and parameter the attributes in the table take their values. This is done via simple XPath statements. For instance, every time a new drug request is inserted into the system, an event of the type $NewDrugRequest$ is emitted that carries the number of pending prescriptions for that drug among its attributes. In order to obtain this information so that we can fill the $PendPresc$ attribute for each row in Table 1, we take all events of type $NewDrugRequest$ and access their $PendPresc$ parameters.

This means that for each new process model or for each change in an existing process model, the process modeler needs to (i) define the process instance table (this is also done in conjunction with the compliance analyst who defines

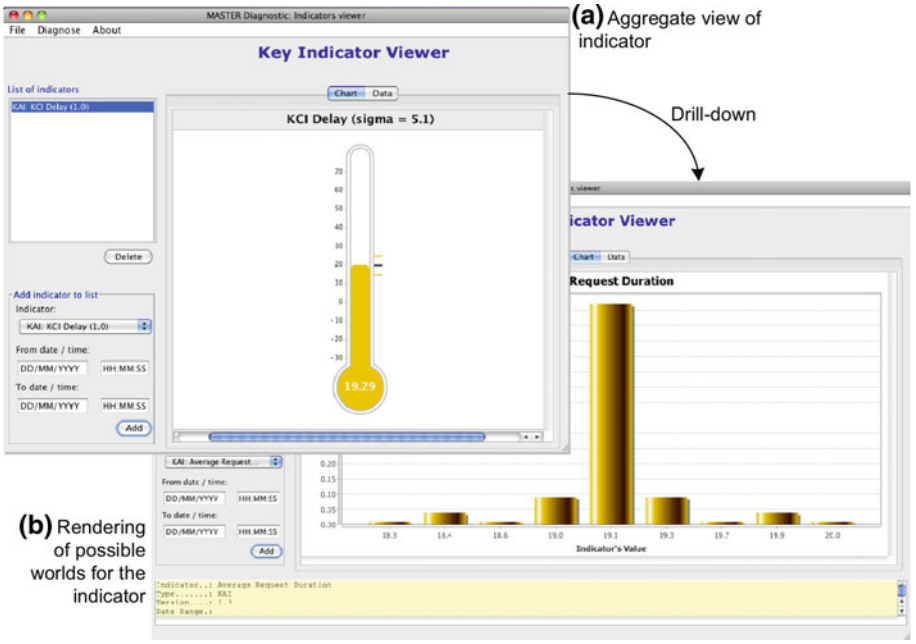

Fig. 10 Visualizing key compliance indicators

which attributes are relevant for compliance analysis) and (ii) define the XPath expression that is used to populate the attributes for each given instance. Overall, this is something that can be done rather easily. The key problem is figuring out good indicators. Once that is done, the time taken to configure the process instance tables for their computation is small.

The ETL procedure that loads the warehouse incrementally and computes indicators runs *periodically*, for example, each night or once a week. Only new and completed event traces (process instances) are loaded; running process instances are not considered. This assures that all events needed to assign the compliance labels are available (partial traces could be analyzed, e.g., to identify early violations; however, compliance can still only be ascertained after process termination). Once computed, also the values of indicators are stored in the warehouse (see the *KeyIndicatorValueFact* table in Fig. 9) and made available for reporting and further analysis, such as correlation and risk analysis.

KCIs can then be rendered in the *reporting dashboard* as illustrated in Fig. 10, which also takes into account data uncertainty when rendering indicator values to users. A description of the dashboard with details on how uncertainty is managed can be found in [23].

6 Act: improving processes and compliance

This is the last phase of the Deming cycle that aims at *understanding* problems identified in the Check phase. While the cycle is closed by the compliance expert and process modeler by applying their findings in a new Plan phase, IT can significantly assist this phase and reduce the complexity of the analysis task. In the context of compliance management, IT can assist in (i) *identifying correlations among KCIs*, (ii) *identifying correlations among compliance states and business data*, and (iii) *reconstructing the actual behavior* of implemented processes.

6.1 Analyzing correlations among indicators

As explained in Sect. 3.2, indicators measure how well business processes conform to compliance requirements. In doing so, each indicator looks at a different aspect of a process, typically a different compliance requirement. Identifying correlations among indicators therefore allows us to identify relationships among compliance requirements. If we visualize all identified relationships in a graph, this allows one to trace back from one indicator to another to understand its root causes. It is important to notice that correlation only indicates likely causal relationship, not certain causalities. The idea of using correlation is to help the human expert to spot places where to look at for root causes.

A particularly interesting analysis is that of *cross-correlating* multiple indicators over time: There may be situations in which changes in the values of an indicator KCI_1 are associated with changes in the values of another indicator KCI_2, but only after a time interval that also needs to be determined. The typical reason for this result is that KCI_1 is computed over events raised at an early stage of the process while KCI_2 is computed over events raised at a later stage. The dynamics of KCI_2 has therefore its likely roots in the part of the process measured by KCI_1.

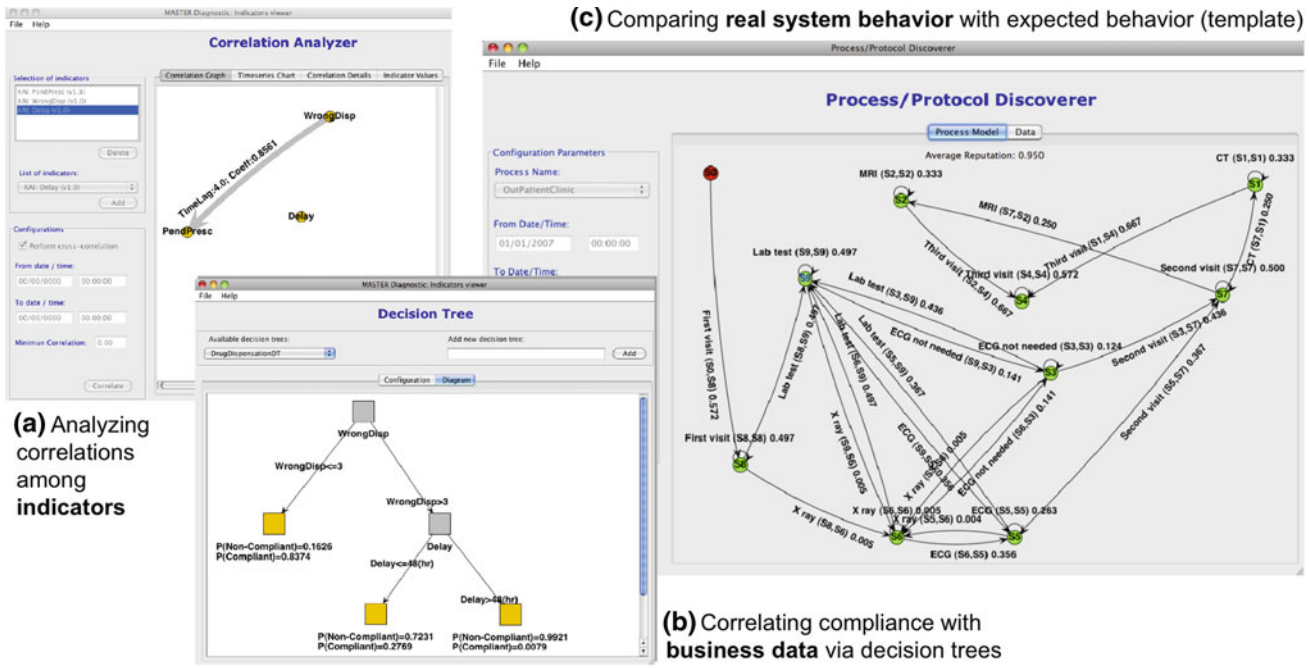

Fig. 11 Instruments of the analysis workbench: correlation analyzer, decision tree miner, protocol discoverer

Figure 11a shows the output of our *correlation analyzer* if applied to the three indicators KCI_{Delay} (introduced in Sect. 3.2), $KCI_{PendPresc}$ (number of pending prescriptions), and $KCI_{WrongDisp}$ (number of erroneous dispensations, i.e., with wrong drug type or quantity). The correlations are based on the cross-correlation technique proposed in [6]. The graph shows a dependency ($coeff = 0.856$) among $KCI_{PendPresc}$ and $KCI_{WrongDisp}$ with a time lag of 4 days (the arrow head of the correlation goes back in time), while there is no correlation with KCI_{Delay} ($coeff < 0.70$). That is, too many pending prescriptions in the system systematically lead to errors (e.g., wrong quantities or drugs dispensed) at dispensation time. More than a simple implementation issue, this correlation hints at an organizational problem in the drug dispensation (e.g., understaffed personnel).

6.2 Classifying compliance evaluations

We have shown earlier that we use process instance tables to store each process instance's event trace along with the data the events provide access to and that we associate compliance labels (i.e., *classes*) to each instance for each compliance template the process has to comply with. This conceptualization of the compliance problem allows us to apply standard classification algorithms to identify correlations between the compliance classes (compliant yes/no) and the process data, hopefully unveiling unknown dependencies. Indeed, the process instance table with the compliance "labeling" is the typical data format that can be fed to classification tools. We use *decision trees*, as they are simple and fast in classifying tuples and—more importantly—they are suitable for knowledge discovery without complex settings or assumptions and are easy to interpret and analyze.

Figure 11b shows, for instance, the decision tree built out of the data stored in Table 1. For this purpose, we have adopted the algorithm presented in [36]. As can be seen in the figure, the main decision point that affects compliance is $WrongDisp$: if $WrongDisp > 3$, non-compliance is very likely. Along this branch, the second decision point depends on the $Delay$ parameter: If it exceeds 48 h, non-compliance is almost sure (99 % of probability), yet also for lower values of $Delay$ non-compliance is the most likely (72 % of probability) outcome.

Decision trees can also be used as a prediction (or risk detection) mechanism. For example, from Fig. 11b, we can derive the following rule:

$$\textbf{if } WrongDisp > 3 \textbf{ and } Delay > 48hr \textbf{ then}$$
$$p(non-compliant) = 0.9921$$

This rule can be used to predict the compliance of process instances while they are still in execution, which allows a company to focus its attention to those process instances that are at risk.

6.3 Discovering business protocol models

The use of the compliance templates introduced in Sect. 3.1 helps the process modeler to specify process models that are

compliant by design with the stated requirements and the logic rules contained in the compliance descriptor. Yet, typically, auditors do not assess compliance by looking at models only; rather, they look at how processes have been *executed concretely*. In fact, it is important to recognize that compliant models *do not* assure compliant execution. In practice, problems simply happen, for instance, due to *human factors* (e.g., untrained personnel or the process administrator explicitly changing a running instance or a deployed model without notifying the compliance expert), *misconfiguration* (e.g., wrong service endpoints), or *system failures* (e.g., a hard drive error). It is impossible to predict these kinds of problems, and therefore, it is even more important to identify them after they occurred.

We approach this need by means of protocol discovery, a problem for which there already exist valuable contributions. [17] presents a good overview on the protocol discovery problem and existing approaches to deal with it. Specifically, we have adapted the algorithm introduced in [16] as this algorithm supports the identification of models from service conversations that may be noisy (erroneously containing data from different conversations) and incomplete (missing part of the data produced in one conversation). The reason for this choice, instead of mainstream process and workflow mining techniques, is that we are interested in mining *events* as generated by the infrastructure, which might consist not only of events from the core business process but also of events generated by control processes put on top of it. Instead of focusing on message exchanges, that is, SOAP messages, we feed the algorithm with events and we identify "conversations" by grouping events according to the process instance they stem from. Figure 11c, for example, shows the output of the *protocol discoverer* if applied to data from the drug dispensation process. The tool uses finite state machines (FSMs) to graphically represent the reconstructed protocol model: Nodes represent intermediate states of a process execution; edges represent events raised during the execution. Nodes are labeled with incremental numbers that serve simply as state IDs, edges with the name of the event they represent, and the probability that the corresponding event took place [21].

6.4 User study and evaluation

Together with Hospital San Raffaele, we carried out an in-depth evaluation of the usability and understandability of methodology described in this paper. The evaluation involved our *target users*, specifically the business process owner (the pharmacy), the process analyst/modeler, an internal auditor, a quality and accreditation expert, IT staff, and the CIO of the hospital, and took the form of a two-day evaluation workshop, which allowed us to collect feedback via questionnaires, interviews, and focus groups.

As *object of the evaluation*, we used the prototypes and demos developed with the audit experts from Deloitte and described in this paper. The evaluation was performed using real data from the scenario described in this paper. The dataset consisted in 30,000 drug dispensations done between January and April 2009. This dataset allowed us to make a realistic demo of our tool to showcase the indicator correlation and decision tree analysis as well as the business protocol discovery. The required size of the dataset for building good correlation and decision tree models depends strongly on the properties of the dataset (e.g., on whether indicators are computed for each process instance only weekly or monthly, or on the number of decision points inside a given process). The protocol discovery algorithm can instead infer a model already from a single process instance, capturing, however, only the behavior of this single process execution. The complexity of cross-correlation is linear in the number of KCIs by the number of considered data items by the number of time shifts [6]; the performance of decision tree computation and protocol discovery is discussed in [36] and [16], respectively.

According to the study, both the compliance templates and the Reporting Dashboard tool (for the Check phase) used to display indicators and navigate through the collected compliance data were perceived as very useful by all participants, while the process analyst, quality and accreditation expert, internal auditor, and business process owner particularly emphasized the usefulness of the correlation analyzer, decision tree miner, and business protocol miner. The overall judgment of the set of tools for the Check and Act phase reached an average score of 8 in an interval that ranges from 1 (very negative) to 10 (very positive).

The complete evaluation report D1.3.2 is available via the project web site (http://www.project-master.eu). Details on the implemented tools and a set of demonstration videos are available via http://mashart.org/SOCA-Compliance.

7 Related work

We discuss the related work in five areas as related to our work, namely IT governance, SOA governance, business process compliance, reporting on business performance, and mining process execution logs.

7.1 IT governance

IT governance aims at ensuring that companies' IT systems sustain and extend the companies' strategies and objectives. Many frameworks have been proposed to approach IT governance, including COBIT, ITIL, ISO 2000, and ISO 17799. The focus of each of these varies from one another, from the alignment of business objectives to IT objectives (e.g., COBIT), to IT service management (e.g., ITIL), and to IT

security management (e.g., ISO 17799). While these frameworks typically provide general guidelines and best practices on how to govern IT, they provide no guidelines that are specific to compliance management. IT governance may act either as the source of compliance requirements or as a guide on how to instrument IT for compliance management. In the first case, for example, it may happen that a company must comply with one of these frameworks in order to provide services to a third party; in the second case, the framework itself can help enable compliance management. As such, IT governance and compliance management therefore complement each other.

7.2 SOA governance

SOA governance can be considered as a branch of IT governance where the focus is put on SOA-based systems. As in IT governance, many frameworks have been proposed to approach SOA governance. For example, Brauer and Kline [2] approach SOA governance in the area of business service life cycle through two key infrastructures: the business service registry and business service management. Software AG [33] proposes a maturity model with six levels: technology enablement, SOA enablement, SOA business services, SOA lifecycle management, SOA consistency, and SOA optimization. It further describes the lifecycle of a service and SOA roles and provides a list of best practices and common mistakes to avoid. SAP AG [28] proposes a list of common guidelines and patterns for the modeling and implementation of enterprise services at different levels, including map of process components and business objects, service interfaces and services operations per business objects, structure of message types, common set of reusable data types, transactional behavior, and service implementation. Oracle's approach to SOA governance [18] proposes a framework and a list of best practices that expands throughout the service lifecycle. It furthermore proposes a list of six steps to a successful SOA governance model, which aims at maturing the overall SOA and thereby its business goals. IBM [3] proposes an approach that relies on a lifecycle for SOA governance, which is distinguishable different from a service lifecycle that is governed. The SOA governance lifecycle consists of 4 phases: (i) in the plan phase, the governance focus is determined, (ii) in the define phase, the SOA governance model is defined, (iii) the enable phase, is where the SOA governance is implemented, and (iv) the measure phase, is where the governance model is measured and refined. All these frameworks deal with the governance of SOA-based systems to different degrees. Just like IT governance focuses on managing the company's IT, SOA governance focuses on managing the overall lifecycle of SOA-based systems, and the guidelines provided there are only at the high level and therefore they are not useful for compliance management as addressed in this paper.

7.3 Business process compliance

There is a considerable amount of work in the area of business process compliance. In [7], the authors describe, for instance, an algorithm to generate a BPMN model from a set of constraints written in deontic logic. In [9], deontic logic is also used to annotate business process models with compliance rules. Such annotations are then used to check compliance of the business process. Hoffmann et al. [14] instead use first-order logic to annotate business process models with compliance constraints. The authors also show how to check compliance of a business process with these constraints. In [5], the authors propose the use of domain-specific languages to annotate processes with compliance constraints, and they equip their modeling tool with compliance-specific views on the process. Shadiq et al. [27] describe how control objectives can be modeled in formal contract language to annotate process models in the form of control tags that can be used by analysis tools to perform compliance checks on the business process model. Governatori et al. [8] advance this line of static compliance check of normative control objectives and provide status reports that highlight problematic cases together with the control objectives that are violated.

Our approach is based on compliance templates that are the starting point for the development of a compliant business process. With this approach, we cover the first phase of the compliance management life cycle. A compliance template implicitly defines the compliant behavior of the resulting business process. The variable parts of the compliance template are annotated with constraints written in first-order logic. As opposed to lines of work like [27] and [8], we prefer to work with first-order logic because it is a standard and well-understood formalism that suffices for our purposes and because compliance experts are more likely to be familiar with it. The so-represented constraints prevent the compliance template from modifications that violate the compliance rules associated to the template. Yet, conceptually every formalism that allows us to express compliance rules over process events could be adopted in our system. From the modeling perspective, we advocate the use of these compliance templates because they are closer to compliance experts and process modelers. We further use compliance templates to provide process modelers with real-time conformance feedback during the instantiation of compliance templates (static compliance checks).

7.4 Reporting on business process performance

Several works focus on the reporting on business process performance. For instance, works like [4,11,22,29,32] and

[23] focus on warehousing process execution data, so as to make these data available in a suitable schema for reporting and OLAP purposes. We face similar reporting issues in our dashboard, yet our aim is to analyze compliance of business processes not performance. This also leads us to the concept of KCIs as a special type of key performance indicator (KPI). In [20], the authors model KPIs and the relationships that exist among them. Our internal, XML-based representation of KCIs is very similar to the model proposed for KPIs, while, instead of modeling relationships, we discover them via cross-correlation for root causes analysis. Finally, there are many business process management commercial suites that include reporting on business process performance as part of the toolset, for example, HP Business Process Monitor, IBM Business Process Manager, Oracle Business Process Management Suite, SAP Business Process Management, and TIBCO Spotfire.

7.5 Mining process execution logs

Data mining techniques have been also used for analyzing business process execution data. As for the *root cause analysis*, Grigori et al. [11,12] focus on understanding, predicting, and preventing exceptions in business executions by using decision trees built upon workflow log files. In the same line of thought, Rozinat and van der Aalst [25] mine event logs for decision point analysis, Apte et al. [1] focus on classification and prediction of customer behaviors, and Seol et al. [31] use the inputs and outputs of each process to build decision trees for the analysis of the efficiency of processes. There are, however, no works that specifically address the problem of understanding compliance violations. In the context of *process and workflow mining*, several works have been proposed that aim at discovering process models and checking the conformance of process executions using process execution data. For instance, works like [10,16], and [19] aim at discovering workflow/process models from execution logs with special focus on the behavioral/structural aspects of the process models. Rozinat and van der Aalst [26,24] focus instead on the automatic verification of how well process executions conform with a predefined process model. We adopt algorithms of the first class for discovering protocol models; however, algorithms of the second class could be adopted for compliance assessment.

8 Conclusion

With this paper, we approach a relevant and critical issue in today's business reality, that is, compliance management, and we do so by specifically taking into account the peculiarities of the service-oriented architecture and of distributed business contexts, two paradigms that heavily influence current and future business practices. Differently from many works in literature, we do not focus on monitoring and enforcement at the individual message level. We rather take the auditor's perspective and focus on the design of compliant processes and the assessment and improvement of their compliance. We assist these activities by means of (i) a *model and tool* to design compliant processes, (ii) an *extended service orchestration engine* to generate process execution evidence, and (iii) a *reporting and analysis suite* to report on compliance and support root cause analysis, in order to provide for better informed decision making. As such, the models and instruments we propose in this paper complement existing monitoring and enforcement approaches and provide for a comprehensive approach to service-based compliance management.

Our aim was to devise a solution having in mind the real needs of auditors (internal and external ones) and—more importantly—with the help of people who are involved every day in the auditing of companies (the dashboard [32] and solutions proposed in this paper have extensively been discussed with partners from Deloitte). While this paper specifically targets a company's internal compliance expert and process modeler, also the external auditor can benefit from the proposed system, for example, by using the compliance reporting dashboard as a starting point for his analysis. This will not change the auditor's own auditing practice, yet the sole use of a systematic and assisted approach to compliance management will surely impact positively on the auditor's perception of the company's commitment to compliance.

Acknowledgments This work was supported by funds from the European Commission (Contract Nbr. 216917 for the FP7-ICT-2007-1 project MASTER).

References

1. Apte C, Bibelnieks E, Natarajan R, Pednault E, Tipu F, Campbell D, Nelson B (2001) Segmentation-based modeling for advanced targeted marketing. In: KDD'01, pp. 408–413
2. Brauer B, Kline S (2005) SOA governance: a key ingredient of the adaptive enterprise. Tech rep, Hewlett-Packard. http://goo.gl/WxTSe
3. Brown W, Moore G, Tegan W (2006) SOA governance: IBM's approach. Tech rep, IBM. http://goo.gl/q9Ini
4. Casati F, Castellanos M, Dayal U, Salazar N (2007) A generic solution for warehousing business process data. In: VLDB'07. VLDB Endowment, pp 1128–1137
5. Daniel F, Casati F, D'Andrea V, Strauch S, Schumm D, Leymann F, Mulo E, Zdun U, Dustdar S, Sebahi S, de Marchi F, Hacid MS (2009) Business compliance governance in service-oriented architectures. In: AINA'09. IEEE Press
6. Dunn P (2004) Measurement and data analysis for engineering and science. McGraw-Hill Science, New York
7. Goedertier S, Vanthienen J (2006) Designing compliant business processes from obligations and permission. In: BPM workshops, vol. 4103. Springer, pp 5–14

8. Governatori G, Hoffmann J, Sadiq SW, Weber I (2008) Detecting regulatory compliance for business process models through semantic annotations. In: BPM workshops, pp 5–17
9. Governatori G, Sadiq S (2009) The journey to business process compliance. In: Handbook of research on business process management, pp 426–454
10. Greco G, Guzzo A, Pontieri L, Sacca D (2006) Discovering expressive process models by clustering log traces. IEEE TKDE 18(8):1010–1027
11. Grigori D, Casati F, Castellanos M, Dayal U, Sayal M, Shan M (2004) Business process intelligence. Comput Ind 53(3):321–343
12. Grigori D, Casati F, Dayal U, Shan MC (2001) Improving business process quality through exception understanding, prediction, and prevention. In: VLDB'01. San Francisco, CA, USA, pp 159–168
13. Hagerty J, Hackbush J, Gaughan D, Jacobson S (2008) The governance, risk management, and compliance spending report, 2008–2009: Inside the $32B GRC Market. Tech rep, AMR Research
14. Hoffmann J, Weber I, Governatori G (2012) On compliance checking for clausal constraints in annotated process models. Inf Syst Frontiers 14(2):155–177
15. Khalaf R, Karastoyanova D, Leymann F (2007) Pluggable framework for enabling the execution of extended BPEL behavior. In: WESOA'07. Springer
16. Motahari-Nezhad HR, Saint-Paul R, Benatallah B, Asati F (2008) Deriving protocol models from imperfect service conversation logs. IEEE Trans Knowl Data Eng 20(12):1683–1698
17. Musaraj K, Yoshida T, Daniel F, Hacid MS, Casati F, Benatallah B (2010) Message correlation and web service protocol mining from inaccurate logs. In: Proceedings of ICWS'10
18. Oracle (2007) SOA governance: framework and best practices. Tech rep, Oracle. URL http://goo.gl/dtZjz
19. Pinter SS, Golani M (2004) Discovering workflow models from activities' lifespans. Comput Ind 53(3):283–296
20. Popova V, Sharpanskykh A (2010) Modeling organizational performance indicators. Inf Syst 35(4):505–527
21. Rodriguez C, Daniel F, Casati F, Anstett T, Schleicher D, Burri S (2009) Warehouse model and diagnostic algorithms. Deliverable d6.2.2, MASTER project. URL http://www.master-fp7.eu/
22. Rodríguez C, Daniel F, Casati F, Cappiello C (2009) Computing uncertain key indicators from uncertain data. In: ICIQ'09, pp 106–120
23. Rodriguez C, Daniel F, Casati F, Cappiello C (2010) Toward uncertain business intelligence: the case of key indicators. IEEE Internet comput 14(4):32–40
24. Rozinat A, van der Aalst WMP (2008) Conformance checking of processes based on monitoring real behavior. Inf Syst 33(1):64–95
25. Rozinat A, van der Aalst WMP (2006) Decision mining in business processes (BETA publicatie: working papers, No. 164) Eindhoven: Technische Universiteit Eindhoven, 16 pp. http://www.tue.nl/en/university/departments/industrial-design/research/experts-expertise/detail/ep/p/d/ep-uid/202689/
26. Rozinat A, Van der Aalst W (2006) Conformance testing: measuring the fit and appropriateness of event logs and process models. In: Business process management workshops. Springer, pp 163–176
27. Sadiq SW, Governatori G, Namiri K (2007) Modeling control objectives for business process compliance. In: BPM'07, pp 149–164
28. Sap AG (2007) Governance for modeling and implementing enterprise services at SAP. URL http://goo.gl/kFAvS
29. Sayal M, Casati F, Dayal U, Shan MC (2002) Business process Cockpit. In: VLDB '02. VLDB Endowment, pp 880–883
30. Schleicher D, Anstett T, Leymann F, Mietzner R (2009) Maintaining compliance in customizable process models. In: CoopIS'09. LNCS, vol 5870, pp 60–75
31. Seol H, Choi J, Park G, Park Y (2007) A framework for benchmarking service process using data envelopment analysis and decision tree. Expert Syst Appl 32(2):432–440
32. Silveira P, Rodríguez C, Casati F, Daniel F, D'Andrea V, Worledge C, Taheri Z (2009) On the design of compliance governance dashboards for effective compliance and audit management. In: NFPSLAM-SOC'09. Springer
33. Software AG (2007) SOA governance: "Rule your SOA". Tech rep, Software AG. URL http://goo.gl/EtgEi
34. Tarantino A (2008) Governance, risk, and compliance handbook. Wiley, New York
35. Trent H (2008) Products for managing governance, risk, and compliance: market fluff or relevant stuff?. In-depth research report, Burton Group
36. Tsang S, Kao B, Yip KY, Ho WS, Lee SD (2009) Decision trees for uncertain data. In: ICDE'09. IEEE, pp 441–444
37. van Lessen T, Leymann F, Mietzner R, Nitzsche J, Schleicher D (2008) A management framework for WS-BPEL. In: ECOWS'08. IEEE, pp 187–196
38. Walton M (1988) The deming management method. Perigee Books, New York