Analyzing Compliance of Service-Based Business Processes for Root-Cause Analysis and Prediction

Carlos Rodríguez, Patrícia Silveira, Florian Daniel, Fabio Casati
University of Trento, Via Sommarive 14
38123 Povo, Trento, Italy
{crodriguez, silveira, daniel, casati}@disi.unitn.it

Abstract. Automatically monitoring and enforcing compliance of service-based business processes with laws, regulations, standards, contracts, or policies is a hot issue in both industry and research. Little attention has however been paid to the problem of *understanding* non-compliance and *improving* business practices to prevent non-compliance in the future, a task that typically still requires human interpretation and intervention. Building upon work on automated detection of non-compliant situations, in this paper we propose a technique for the root-cause analysis of encountered problems and for the prediction of likely compliance states of running processes that leverages (i) on event-based service infrastructures, in order to collect execution evidence, and (ii) on the concept of key compliance indicator, in order to focus the analysis on the right data. We validate our ideas and algorithms on real data from an internal process of a hospital.

Keywords: Compliance, Decision Trees, SOA, Root-Cause Analysis.

1 Introduction

Compliance means conformance with laws, regulations, standards, contracts, policies, or similar sources of requirements on how to run business. Effective *compliance management*, i.e., the practice of assuring compliance, is an increasingly more important concern in today's companies, since the set of compliance requirements a company has to implement grows fast and their effect on the "traditional" business practices in a company may be considerable. Despite its increasing importance, compliance is however to a large extent still managed 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 questions like: 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? And so on.

While IT has been supporting (in more or less automated fashions) the execution of business processes for long time now, in the past the adoption of ad-hoc and monolithic software solutions did not provide the necessary insight into how processes were executed and into their runtime state, preventing the adoption of IT also for compliance assessment. The advent of workflow management systems and, especially today, of web service-based business interactions and the service-oriented architecture

The final authenticated version is available online at:
https://doi.org/10.1007/978-3-642-16985-4_25

(SOA) have changed this shortcoming, turning business processes into well-structured, modular, and distributed software artifacts that provide insight into their internals, e.g., in terms of execution events for tasks, service calls, exchanged SOAP messages, control flow decisions, or data flows. All these pieces of information can be used for online monitoring or enforcement of compliant process behaviors or they can be logged for later assessment. Unfortunately, however, the resulting amount of data may be huge (in large companies, hundreds of events may be generated per minute!), and – especially in terms of reporting and analysis – it is not trivial to understand which data to focus on and how to get useful information out of them.

Doing so is challenging and requires answering questions like how to collect and store evidence for compliance assessment in service-based business processes, how to report on the compliance state, and how to support the analysis of non-compliant situations. But more than these, the challenges this paper aims to solve are how to collect evidence in a way that is *as less intrusive as possible*, how to devise solutions that are *as useful as possible*, yet – at the same time – *as generic as possible* and independent of the particular IT system to be analyzed, and, finally, how to provide compliance experts with information that is *as useful and expressive as possible*. In light of these challenges, this paper provides the following contributions:

- A method for the definition and a dashboard for the visualization of so-called *Key Compliance Indicators* (KCIs) for at-a-glance reporting on compliance;
- An algorithm and a tool for the *mining of decision trees* from process execution logs that particularly look at data from the perspective of compliance;
- An application of the algorithm mining approach to *real-world data* stemming from a typical business process running in a large Italian hospital.

In the next section we provide the necessary details about this process and highlight its compliance requirements, so as to derive the requirements for this paper in Section 3. In Section 4 and 5, we then discuss how to report on compliance and how to analyze non-compliance, respectively. In Section 6 we discuss some related works, and in Section 7 we conclude the paper.

2 Scenario: Drug Reimbursement in Hospitals

Let us consider the case of a drug reimbursement process in the healthcare domain. The process is the case study in one of our EU projects, where we cooperate with Hospital San Raffaele (Milan, Italy), which runs the process shown in Figure 1. The overall purpose of this process, from the hospital's point of view, is to obtain reimbursements from the Italian Health Authority for the drugs dispensed to outpatients (i.e., patients that are not hospitalized). In order to obtain the reimbursement, there are many compliance requirements imposed by the Health Authority, among which we mention privacy preservation in personal information processing, separation of duties, and the adherence of standard template of dispensation reports.

The core process that generates the information that needs to be sent to the Health Authority occurs inside the Ward. The process starts when a patient visits the hospital's ward to consult a doctor. After diagnosing the patient, the doctor prepares a drug prescription that is delivered to a nurse, who is in charge of dispensing the prescribed

drugs to the patient. If the amount of drugs is going below a certain threshold, the nurse issues a drug request to the central pharmacy of the hospital, which must replenish the ward's drug stock in no later than 48 hours. The execution of this process is fully supported by the ward's SOA-based information system, and all progress events generated during process executions are recorded in an event log for later inspection.

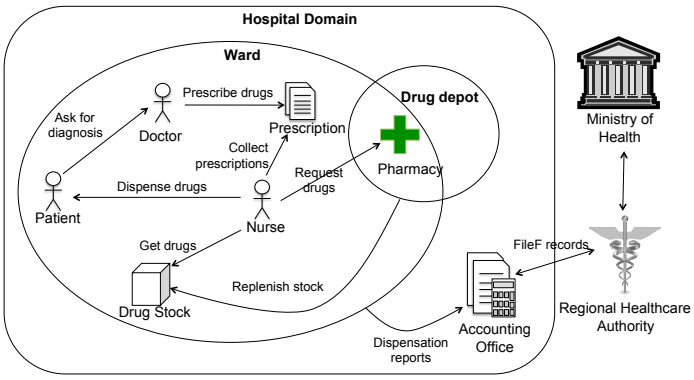

Fig. 1 Summary of the direct drug reimbursement process.

While the process above is executed daily, the preparation of dispensation reports for drug reimbursement is a monthly task. That is, at the end of each month, the records of drug dispensations are collected from the various wards of the hospital and the corresponding dispensation reports to be sent to the Health Authority are created. These reports consist in simple text files (known as *FileF*) in which data about the dispensations are included. Examples of data included in these files are *hospital identification*, *patient, doctor, dispensed drug and quantity,* and *amount in Euros*. Whenever the report is ready it is sent to the Health Authority, which checks the quality of the report against some compliance requirements imposed on dispensation reports. For instance, one compliance requirement that decides whether a dispensation can be reimbursed or not regards the completeness and correctness of records: no *null* or incorrect data are tolerated in any field. If there are such problems in the report, the Health Authority sends a feedback to the hospital indicating the number and type of errors found for each record of the file, and, in turn, the hospital must correct them so as to get the reimbursement.

The complete reimbursement process is complex, and not complying with the applicable requirements can be costly. Therefore, in order to better control the compliance of the reimbursement process, the hospital wants to implement an early warning system that allows the hospital's compliance expert to have updated information on daily compliance issues, e.g., in form of indicators, reports, or predictions on the compliance of its processes. In addition, in case of repeated problems, it is important to understand why they happen and how they can be solved for the future. However, manually analyzing the data in the event log is time consuming and also error-prone but, still, the hospital wants to improve its compliance in order not to lose money for not reimbursed drug dispensations.

3 Service-Oriented Compliance Management: Requirements

The above scenario describes a service-based business process that is distributed over the hospital's ward and the drug depot and that asks for proper compliance management, that is, compliance assessment, reporting, and analysis.

As this paper has its roots in two EU FP7 research projects, i.e., Compas and Master, that both assist **compliance assessment** in the SOA, here we do not propose a new assessment technique and rather rely on the techniques proposed there: *Compas* (www.compas-ict.eu) strongly focuses on model-driven development of compliant processes and proposes a compliance checking approach that is based on (i) compliance requirements expressed in logical rules or process fragments and (ii) complex event processing (CEP) and business protocol monitoring to detect non-compliance with requirements. *Master* (www.master-fp7.eu), instead, specifically focuses on the security domain and proposes a two-layered approach to compliance assessment: first, it supports the CEP-based monitoring of running processes and the enforcement of individual rules; then, offline, it checks compliance of executed processes by assessing their conformance to a so-called ideal process model. Both approaches have in common the use of an instrumented service orchestration engine for the execution of business processes and the generation/logging of suitable execution events, starting from a signaling policy that specifies which events are necessary for compliance assessment.

Building on this background, **reporting on the state of compliance** requires being able to *store process execution and compliance data* and to develop a *reporting dashboard* on top, a task that we partly approached in [1]. But we also need to devise a method for the *easy specification* and, then, automated computation of *key compliance indicators* (KCIs), in order to visualize them in the dashboard. Next, the **analysis of root-causes for non-compliance** requires selecting a suitable *analysis algorithm* and – more importantly – understanding *which data* to look at, out of the huge amount of data that is available for this task, and to validate the algorithm in the context of the described scenario.

4 Reporting on Compliance

In order report on the compliance of business processes, the common approach is to visualize the compliance status at a high-level of abstraction, for instance, by means of KCIs that are graphically rendered in a compliance governance dashboard (CGD) [1]. KCIs support compliance experts with an overview of the compliance performance of business processes and can be seen as particular type of KPIs (key performance indicators) that specifically measures how compliant a process is with given requirements. A typical KCI may, for example, measure how many process instances, out of all the executed ones, satisfy a separation of duties requirement; but also a traditional QoS indicator (e.g., the average process execution time) can be seen as KCI, if we are subject to a compliance requirement regarding QoS (e.g., deriving from a contract with the customer). As we will see, KCIs also provide a starting point for finding the root-causes of non-compliance. This section explains how we store

process execution data, specify and compute KCIs, and visualize them through effective visual metaphors.

4.1 Storing Process Execution and Compliance Data

The main sources of process execution and compliance data are the *event logs* generated by the execution of service-based business processes. Therefore, let us first conceptualize the key ingredients characterizing event logs, as we perceive them for our analysis. An *event* is a tuple $e = <t, s, ts, d, p_1,\ldots, p_n, B>$, where t is the type of the event (e.g., *ProcessStart*, *ActivityExecuted*, *Violation*), s is the source that generates the event, ts is a timestamp, $p_1,\ldots, p_n$ is a set of properties (e.g., event message header properties such as correlation data, process instance identifier or similar), and B is the body of the event message (e.g., containing business data needed for the computation of an indicator). Using this data, events can be grouped together by their process instance and ordered by timestamp, forming this way traces. A *trace* is a sequence of events $T_i=<e_{i1}, e_{i2}, \ldots, e_{in}>$, where i refers to a process instance identifier and n is the number of events that compose the process instance. This way, an *event log* can be expressed as a set of traces $L=\{T_1, T_2, \ldots, T_k\}$, where k is the total number of traces.

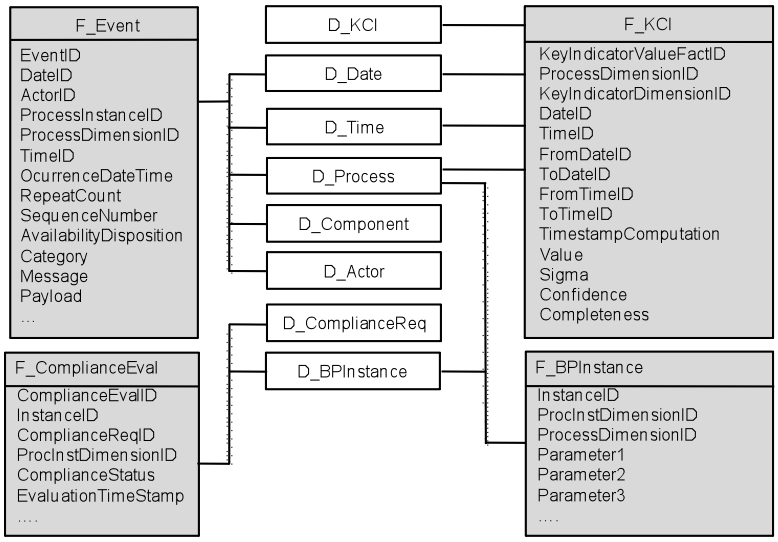

Fig. 2 Simplified schema of the data warehouse model.

The events in the log are processed by Extract-Transform-Load (ETL) flows, in order to store them into a data warehouse (DW), which is modeled using a compliance-oriented dimensional data model. The reason for doing this is that we aim at leveraging the capability of dimensional models for keeping a conciliated view on the process execution and compliance data, and for supporting further analysis, e.g., by means of root-cause analysis algorithms or Online Analytical Processing (OLAP) tools. Figure 2 shows an excerpt of the schema of the DW. The tables in white are the *dimensional tables* that allow us to slice and dice through the *fact tables* (shaded in

gray). The fact table F_Event stores the events as they come from the event log, F_KCI stores the computed values of indicators, F_BPInstance, the instances of processes, and F_ComplianceEval, the compliance status of process instances as computed, for instance, by the compliance checking algorithms adopted in the context of the Compas or Master projects.

The F_BPInstance table deserves a further explanation, as it constitutes an *abstraction* of the process execution data, and the basis for computing indicators and performing root-cause analysis. In our DW model, each business process *BP* has its own F_BPInstance table, or, as we call it, *process instance table* (e.g., in our scenario we have a F_DrugDispensationInstance table). In these tables, each row corresponds to an instance of the associated process, while columns (i.e., parameters of the process instance table) correspond to business data that are of interest for the analysis of each process. Table 1 shows a conceptual view on the process instance table for the drug dispensation process, where each row corresponds to a single drug dispensation. The *DrugType* column refers to the type of drug, *ErrPerData* indicates whether there was an error in the information about the patient, *ErrCompData* tells us if there was an error in any other complementary data, and *Compliant* tells us whether the dispensation was free of error. These parameters are obtained from the attributes of the events that are part of the event trace. Sometimes, the parameter values can be directly extracted from events without modifications (e.g., the *DrugType* parameter), while in other cases the values are obtained by performing aggregation/computations over a set of events and attributes of process instances (e.g., the *Compliant* parameter).

Table 1: Example of a process instance table for the drug dispensation process.

InstanceID	DrugType	ErrPerData	ErrCompData	…	Compliant
38769	1	False	False	…	True
32537	6	True	False	…	False
27657	1	False	False	…	True
32547	2	False	True	…	False
35340	1	False	False	…	True
….	….	….	….	…	….

Finally, it is worth to mention that in order to populate the DW, the ETL usually needs to access other sources of data such as user management systems and human task managers, which are the main data providers for dimension tables, as opposed to event logs, which provide mostly the evidences of process executions.

4.2 Specifying and Computing Key Compliance Indicators

Generally, indicators are computed out of a variety of data and by means of different functions, ranging from the lowest business data granularity to the highest business goals. In the context of compliance assessment, a KCI is a measure (i.e., a numeric value) that quantifies compliance performance against compliance targets in a predetermined time interval. For instance, one of the compliance requirements imposed

by the Healthcare Authority is that of sending drug dispensation reports without errors in the data about dispensed drugs and patients. Whenever there is an erroneous record of drug dispensation, the corresponding drug is not reimbursed to the hospital, and, thus, it is important for the hospital to keep an eye on the accomplishment of this compliance requirement. KCIs are therefore useful means to assist this task.

KCIs can be easily specified by using the available information in Table 1. For example, a KCI may be defined as the percentage of non-compliant process instances out of all instances in the DW (and the reporting time interval). More precisely, we can use the *Compliant* column of a process instance table to compute KCIs, and we can express their respective formulas using standard SQL queries. 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. Yet, the ease with which we are able to express KCIs stems from the abstraction we made on the process execution data by using the so called process instance tables.

4.3 Compliance Governance Dashboard

Finally, KCIs are rendered to the compliance experts by means of a CGD, such as the one depicted in Figure 3 [1]. The CGD features are a graphical representation of KCIs and serves as start point for further root-cause analysis. More specifically, the CGD creates an awareness of possible violations and concentrates the most important information to be evaluated at-a-glance. The interactive table (at the front in Figure 3) provides a drill-down and roll-up mechanism for the compliance status, for example, for the different drug dispensation locations controlled by the hospital (i.e., clinics, laboratories, dispensaries), according to two main analysis perspectives (compliance performance vs. process performance), down to the individual event level (e.g., the list of incomplete records associated to a drug (background of Figure 3).

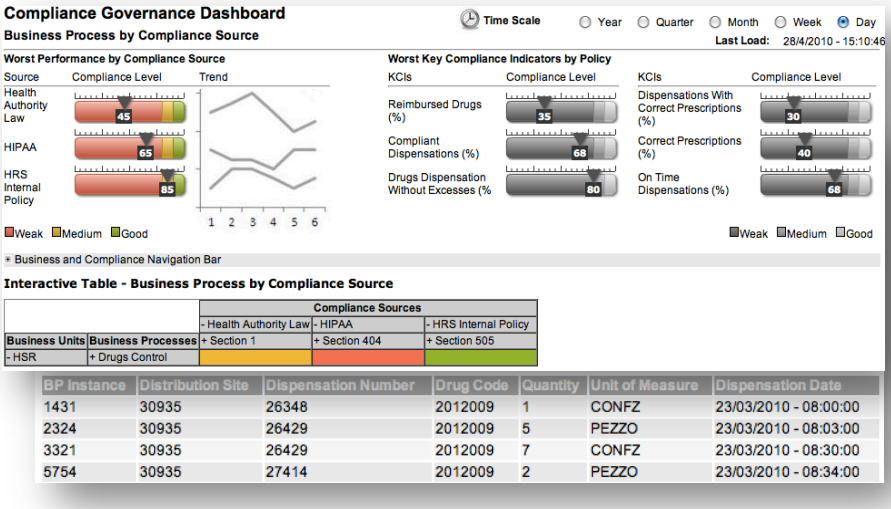

Fig. 3 CGD with KCIs and the interactive table for drill-down and roll-up [1].

5 Analyzing Non-Compliance

While *checking* the compliance of business process instances means determining whether the process instances are compliant or not at the individual event trace level, *analyzing* non-compliance of business process executions, i.e., understanding and explaining the underlying reasons of non-compliance, needs to be performed over a set of traces in order to be able to derive meaningful knowledge that can be used to improve processes for future executions.

Incidentally, labeling event traces as compliant or non-compliant, which is the main goal of compliance checking, is very similar to *classifying* data tuples, a data mining practice that is well-studied in literature [20]. There are several algorithms that can help in performing this analysis, among which we choose decision trees, as they are good for knowledge discovery where neither complex settings nor assumptions are required [20], and they are easy to interpret and analyze. In this section, we discuss how we address the issue of compliance analysis through decision trees, going from data preparation to the actual building and interpretation of the decision tree.

5.1 Preparing the Analysis

In Section 4.1, we introduced our DW model, which constitutes the basis for our CGD and the root-cause analysis. Preparing the analysis therefore means selecting which data, out of the huge amount of events stored in the DW, are suitable for identifying root-causes for non-compliance. In the same section, we also introduced the idea of having process instance tables, one per process, in which we store those process parameters that are used for computing indicators. Recall that each tuple in a process instance table represents a particular instantiation of the process under consideration and that each instance comes with its compliance label. Now, considering that we are interested in analyzing non-compliance problems for process instances, it is interesting to note that the process instance tables initially conceived for the computation of indicators also contain the data we are searching for. In fact, by defining a set of indicators for each process (and the events and data attributes that are necessary to compute them), the compliance expert implicitly performs a pre-selection of the data that are most likely to be related with compliance issues. The availability of the compliance label for each instance indicates that the best choice for the root-cause analysis is to use the process instance tables to feed the decision tree mining algorithm, as their data naturally fits the typical input format of these kinds of algorithms.

For instance, considering again the process instance table shown in Table 1, one way of building the training tuples for the decision tree is to use the *Compliant* column as the *class attribute* (leaf nodes) for the decision tree, while *ErrPerData* and *ErrCompData* can be used as the attributes on which the algorithm defines the split points (for internal nodes). This way, the training tuples can be represented as

$$<ErrPerData, ErrCompData, \mathbf{Compliant}>$$

The set of training tuples can be easily obtained through trivial SQL queries, and the retrieved result set can be used directly to feed the decision tree algorithm. Note that, as in the case of the specification and computation of the KCIs, the task of build-

ing the training tuples is greatly facilitated by the abstraction provided by the process instance tables.

5.2 Understanding Key Factors

The algorithm we use in our prototype implementation for building decision trees extends the C4.5 algorithm to handle uncertain data [21]. In this paper we do not discuss the uncertainty aspect in mining data. However, our prototypes are equipped to handle uncertainty in the event logs we use for analyzing business process executions (for details on how uncertainty in event logs can be handled, see [6]). Instead, here we focus more on the aspect of discovering and understanding the key factors that affect the compliance of business executions.

As in any decision tree, the internal nodes contain the criteria used for classifying tuples. The leaf nodes, instead, contain the classes to which tuples are classified. For instance, if we choose the *Compliant* column of Table 1 as the class attribute, we will obtain a decision tree where the leaf nodes contain the *compliance outcomes* for the paths drawn from the root of the tree. However, nothing prevents us from choosing any other parameter of the process instance table as the class attribute when searching for the root-causes of non-compliant process executions.

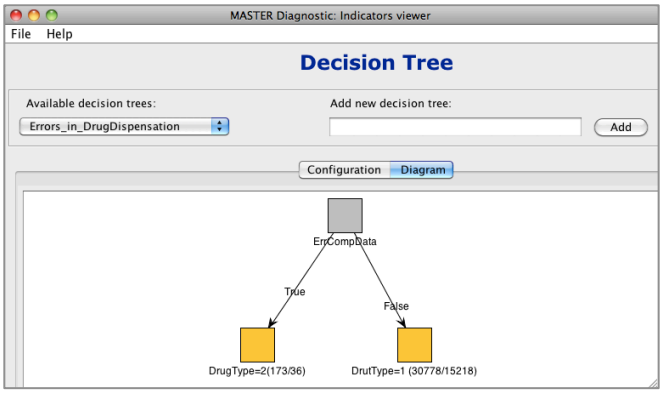

Fig. 4 Decision tree computed over non-compliant instances of the drug dispensation process.

For instance, as part of the validation of this approach, we performed experiments on a dataset of more than 30000 drug dispensations performed between January and April of 2009 in the hospital described in the scenario (Section 2). To this end, a process instance table with around 25 relevant parameters was build for the drug dispensation process, among which the parameters shown in Table 1 were included. Since the dependence of the *Compliance* column on the *ErrPerData* and *ErrCompData* columns was fairly obvious (but still, proven with our tools), we narrowed our analysis by considering only those process instances that were not compliant. After exploring some combinations of parameters, we found out that there was a relation between the *ErrCompData* and *DrugType* parameters. More precisely, we found that 393 drugs dispensations out of around 30000 had some error, among which 173 had errors of the type *ErrCompData* and 220 errors of the type *ErrPerData*. While the decision tree was not able to tell us anything that was really significant about errors of

the type *ErrPerData*, it was able to find something useful for the errors of the type *ErrCompData*, as shown in Figure 4. More precisely, the decision tree discovered that 137 out of 173 (79%) erroneous process instances corresponded to drugs of the type 2 (DrugType=2), which are drugs for ambulatory usage, while the rest (21%) corresponded to drugs of the type 6, 9 and 11.

Since the *ErrCompData* refers to error in the dispensation data (such as the drug code, quantity and unitary price), this may be an indication that, for example, this type of drugs is dispensed at ease, and thus, a better monitoring or compliance enforcement need to be carried out on the controls related to this compliance requirement.

5.3 Predicting Compliance States

While decision trees are generally perceived as simple classifiers, we however use them rather for discovering and understanding better the root-causes of undesirable behaviors. Furthermore, we advocate the use of decision trees also for predicting the potential outcomes of process instances that are still running. In fact, each decision point in a tree corresponds to an event (or better to an attribute of an event). So, if during process execution an event that corresponds to a decision point is generated, this allows performing predictions on the likely outcome (in terms of compliance) of the process instance: it suffices to inspect the path in the tree determined by the registered event to identify the instances' likely compliance label.

Thus, in the case of predictions of non-compliant behaviors, enforcement actions can be enacted in order to align process executions, whenever possible, to the corresponding compliance requirements. This is particularly useful in cases when the process has several tasks and long running times that span, e.g., over several hours. Also, the prediction is particularly useful in the case compliance is enforced manually, because it allows the compliance expert to better focus his effort on those process instances that are likely to be non-compliant, leaving out compliance ones.

6 Related Work

The major part of compliance management approaches focuses on the *business process modeling* aspect at design time [7-9]. Typically, they are based on formal languages to express compliance requirements (e.g., Business Property Specification Language, Linear Temporal Logic) and simulations to prevent errors at runtime (e.g., finite state machine, Petri nets). In this context, just few approaches address *compliance monitoring at runtime*. For instance, Trinh et. al. [10] monitor time constraints during the execution of process activities, using UML Timing Diagrams to specify constraints and Aspect Oriented Programming to control executions. Chung et. al. [11] check if the user-defined process is compliant to pre-defined ontology and a specific model, in which compliance requirements are described. An IBM research group [12] advocates the use of the REALM (Regulations Expressed As Logical Models) metamodel to define temporal compliance rules and the Active Correlation Technology to check them. That way, it can detect duplicate events or compute a user-definable function, which checks whether a function exceeds some threshold.

Concurrently, commercial *Business Activity Monitoring* (BAM) solutions have been developed to support compliance management (e.g., IBM Tivoli, HP Business Availability Center, Nimbus, Oracle Business Activity Monitoring). Although, such tools still do not have the capability to process and interpret *generic events* (e.g., user-defined business or compliance-related events). They only support the definition of thresholds for parameters or SLAs to be monitored. Also, the ability to *compare* monitored business process executions or, more in general, business patterns with expected execution behaviors is not supported.

Regarding reporting on compliance and KCIs, few works address this aspect and they do it partially. For example, [18] studies the representation of data through visual languages for risk and compliance management. In [19], the authors purpose a model-driven fashion approach to report on business performance and design dashboards.

To the best of our knowledge, no *mining approaches* have been specifically proposed to understand the root-cause of the compliance violations. However, few related approaches for the mining of business processes are in place [3-5][14-16]. Similar to our solution, they adopted log files and a consolidated warehouse containing business and process historical data, from where data subsets are extracted and used as input to mining algorithms in order to predict or understand the origin of undesired business process execution behaviors.

Finally, we can conclude that Compas and Master have been done significant contributions in all the fields mentioned in this section, since they provide solutions to manage, monitor and report on compliance based on generic events. For instance, [2][13] provide approaches to the management of the compliance monitoring at runtime, [17] states how to compute uncertain key indicators from uncertain data, [1] presents CGD to report on compliance and this paper presents root-cause analysis based on data mining techniques to understand non-compliant business processes.

7 Conclusions and Future Work

In this paper, we leverage on automated compliance checking techniques and complement them with a tactical perspective that targets compliance experts, which are accountable for assuring and improving compliance. We assist them by automating the analysis of the huge amount of data that is produced during process execution and specifically provide (i) a reporting dashboard with KCIs and KPIs to assess the state of compliance, (ii) a root-cause analysis technique to understand non-compliance. Our experiments with real data from a major Italian hospital show that the developed dashboard is effective in highlighting encountered problems and that the proposed abstractions and selection of data indeed allow us to identify also unexpected causes for non-compliant situations out of a large amounts of data.

It is important to note that, although in this paper we focused on the case of compliance, the ideas and solutions we propose are of general nature and can, for instance, easily be applied to the computation and analysis of KPIs. Similarly, we are not limited to process engine event as only source of information; events may also stem from web services, human task managers, or similar – if suitably instrumented.

Acknowledgements: This work was supported by funds from the European Commission (contract N° 216917 / MASTER and contract N° 215175 / COMPAS).

References

1. Silveira, P., Rodríguez, C., Casati, F., Daniel, F., D'Andrea, V., Worledge, C., Taheri, C.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: NFPSLAM-SOC'09, Springer, Stockholm, Sweden (2009)
2. Daniel, F., D'Andrea, V., Strauch, S., Schumm, D., Leymann, F., Mulo, E., Zdun, U., Dustdar, S., Sebahi, S., de Marchi, F., Hacid, M.: Business Compliance Governance in Service-Oriented Architectures. In: AINA'09, IEEE Press (2009)
3. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in Business Processes. BETA Working Paper Series, WP 164, Eindhoven University of Technology, Eindhoven (2006)
4. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M. Business Process Intelligence. Computers in Industry Journal, vol. 53/3, pp. 321-343 (2004)
5. Seol, H., Choi, J., Park, G., Park, Y.: A framework for benchmarking service process using data envelopment analysis and decision tree. ESA, vol. 32, pp. 432-440 (2007)
6. Rodríguez, C., Daniel, F., Casati, F., Cappiello, C.: Toward Uncertain Business Intelligence: the Case of Key Indicators. IEEE Internet Computing, vol. 14, no. 4 (2010)
7. Liu, Y., Mueller, S., Xu, K.: A Static Compliance Checking Framework for Business Process Models. IBM Systems Journal, vol. 46, no. 2, pp. 335-361 (2007)
8. Lu, R., Sadiq, S., Governatori, G.: Compliance Aware Business Process Design. In: BPM'05, pp. 120-131 (2007)
9. Awad, A., Weske, M.: Visualization of Compliance Violation in Business Process Models. In: BPI'09, LNCS, vol. 43, pp. 182-193 (2009)
10. Trinh, T., Do, T., Truong, N., Nguyen, V.: Checking the Compliance of Timing Constraints in Software Applications. In: KSE'09, pp. 220-225 (2009)
11. Chung, P., Cheung, L., Machin, C.: Compliance Flow - Managing the compliance of dynamic and complex processes. Knowledge-Based Systems, vol. 21 no. 4, pp. 332-354 (2008)
12. Giblin, C., Müller, S., Pfitzmann, B.: From Regulatory Policies to Event Monitoring Rules: Towards Model-Driven Compliance Automation. IBM Research Report RZ 3662 (2006)
13. Mulo, E., Zdun, U., Dustdar, S.: Monitoring Web Service Event Trails for Business Compliance. In: SOCA'09, IEEE Computer Society Press (2009)
14. Grigori, D., Casati, F., Dayal, U., Shan, M.: Improving business process quality through exception understanding, prediction, and prevention. In: VLDB'01, pp. 159-168 (2001)
15. Apte, C., Bibelnieks, E., Natarajan, R., Pednault, E., Tipu, F., Campbell, D., Nelson, B.: Segmentation-Based Modeling for Advanced Targeted Marketing. In: Knowledge Discovery in Databases and Data Mining. ACM, pp. 408-413 (2001)
16. Bibelnieks, E., Campbell, D.: Mail Stream Streamlining, Catalog Age, vol. 17, no. 12, pp. 118--120 (2000)
17. Rodríquez, C., Daniel, F., Casati, F., Cappiello, C.: Computing Uncertain Key Indicators From Uncertain Data. In: ICIQ'09 (2009)
18. Bellamy, R., Erickson, T., Fuller, B., Kellogg, W., Rosenbaum, R., Thomas, J., Wolf. T.: Seeing is believing: Designing visualizations for managing risk and compliance. IBM Systems Journal, vol. 46, no. 2, pp. 205-218 (2007)
19. Chowdhary, P., Palpanas, T., Pinel, F., Chen, S., Wu, F.: Model-driven Dashboards for Business Performance Reporting. In: EDOC'06, pp. 374-386 (2006)
20. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2006)
21. Tsang, S., Kao, B., Yip, K., Ho, W., Lee, S.: Decision Trees for Uncertain Data. In: ICDE'09. IEEE pp. 441-444 (2009)