Making Business Processes Compliant to Standards & Regulations

Michael P. Papazoglou
European Research Institute in Service Science (ERISS),
Tilburg University,
5000 LE, Tilburg, the Netherlands
e-mail: mikep@uvt.nl

Abstract Compliance regulations require enterprises to review their SOA applications to ensure that they satisfy the set of relevant compliance requirements. Despite an increasing number of methods and tools, organizations have a pressing need for a comprehensive compliance framework to help them ensure that their business processes comply with requirements set forth by regulations, laws, and standards.

In this paper we explain how to cope with business process compliance requirements and present a framework to capture and manage compliance requirements. We introduce a declarative Compliance Request Language for specifying compliance requirements. We also examine a set of compliance patterns to support the definition of frequently recurring compliance requirements in association with business processes. This approach enables the application of automated tools for compliance analysis and verification.

Keywords Regulatory compliance; Compliance constraint detection and prevention; Compliance Request Language; Compliance patterns; Root-cause analysis.

I. INTRODUCTION

Business processes form the foundation for all organizations, and as such, they are impacted by industry regulations. Without explicit business process definitions, flexible rule frameworks, and audit trails, organizations face litigation risks and even criminal penalties. Compliance regulations, such as HIPAA, Basel II, Sarbanes-Oxley, standards and codes of practice, e.g., SCOR or the ISO9000 standard for certification, require all organizations to review their business processes and ensure that they meet the relevant derivatives.

Business process compliance is about ensuring that business processes, operations and practices are in accordance with a prescribed and/or agreed on set of norms [1]. A business process compliance constraint refers to any explicitly stated rule or regulation that prescribes any aspect of an internal or cross-organizational business process. Compliance constraints may emerge from different sources, such as legislation and regulatory bodies, standards and code of practices and contracts between interacting parties, e.g., Service-Level Agreements (SLAs). Business process compliance rules may include terms relating to data acquisition and archival, document management, data security, financial accounting practices, and financial reporting. In all cases, these new control and disclosure requirements create auditing demands for SOAs.

Internal control is associated intrinsically with business process compliance. Internal control can be defined a set of directives designed to provide reasonable assurance on achievement of company’s objectives in areas of effectiveness and efficiency of processes and economic use of resources, reliability of financial reporting information and compliance with external rules and regulations as well as internal policies and procedures [2].

Internal controls are preventive or detective in nature. Examples of preventive controls include authorization lists, computer edits, segregation of duties, and prior supervisory approval. Preventive controls may be time consuming and expensive. Detective controls do not prevent fraud or errors. They will identify that a problem has occurred. Examples of detective controls include reconciliation, exception reports, and supervisory review.

Compliance regulations influence the process of internal control, demanding that internal control procedures are continuously monitored, tested and improved. At the business process level, internal controls are applied to specific business activities and associated business process segments. Most business processes are automated and integrated with enterprise application systems, resulting in many of the controls at this level being automated as well. For instance, typical financial reporting control might mitigate the risk of misstating revenue due to inadequate physical or electronic security over documents and electronic files. However, some controls within the business process remain as manual procedures, such as authorization for transactions, separation of duties and manual reconciliations. Accordingly, controls at the business process level are a combination of manual controls operated by the business and automated controls.

Contemporary approaches to managing internal controls are rather fragmented, costly, and lead to reactive – rather than proactive – risk prevention, inefficient compliance procedures, and a lack of visibility into access risk. Existing solutions and tools are rather outdated, mainly offering solutions for monolithic applications (such as ERP systems) that are not feasible in SOA environments [3].

Most existing Computer Assisted Audit Techniques, provide support merely for document management, financial data-analysis and flowcharting [4], [5]. On the other hand, Business Process Management (BPM) and
Business Process Activity (BPA) solutions predominantly focus on business performance monitoring and continuous evaluation of process execution against service level objectives. They usually, depict information that allows business analysts to manage business process performance, identify problems, measure process improvements, and communicate these to the organization. However, they do not check processes against regulations and compliance acts, and lack support for compliance reporting, alerting and initiating corrective actions.

Almost any business process can be assessed through comparing it against a process compliance index – a set of essential conformance criteria (denoted by Key Performance indicators or KPIs) that indicate success or failure of the process. Such an index is a multi-level description of process compliance metrics – constraints on the scope, timing, schedule, etc, for the entire process and for its particular elements (steps, outputs, etc). The business process is continually assessed against the process compliance measurements to produce a report for the process’s analysts and auditors to qualify its whole design (or its every specific running) as successful or error-prone.

To address the problems that relate to compliance management, including reuse and adaptability, requires a sustainable approach throughout the business process life cycle [1]. Compliance management techniques have a preventive nature and aim at guaranteeing that process instances will be regulatory compliant. A preventive focus is fundamentally required from the early stages of business process design ranging to the monitoring and adaptation of the running process instances [6].

Fig. 1 illustrates how the main phases in the business process lifecycle are interconnected to internal control to achieve compliance to acts and regulations. The use of BPM solutions enables an organization not only to implement more effective business processes, but also to significantly improve its internal control. The design (and introduction) of internal controls impacts the behavior of business processes and may lead to process adjustments and (re)design. Similarly, the (re)design of a business process in many cases may cause a corresponding update of risk assessment, which may lead to a new/updated set of controls. In most cases these adjustments are implemented by involving a business analyst. Authorization, segregation of duties, application control and auditability are the standard categories of IT internal control [7, 8].

Due to inherent communication, accounting and process complexity, most companies cannot address these requirements without a strategy and associated framework for automating compliance audits.

In this paper we explain how to cope with business process compliance requirements and present a conceptual model and framework to capture and manage compliance requirements. The compliance framework uses a declarative Compliance Request Language (CRL) for specifying compliance requirements, which are eventually checked against existing business processes. The compliance language also employs a set of compliance patterns to support the definition of frequently recurring compliance requirements in association with business processes. This approach enables the application of automated tools for compliance analysis and verification.

The paper first explains the importance of risk management techniques, and then introduces a framework for compliance management in section-III. This framework incorporates a compliance model (in section-IV) and a compliance pattern-based formalism (in section-V). Section-VI explains how the framework deals with root-cause analysis of compliance violations, while section-VII briefly introduces the main elements of
compliance request language based on the compliance patterns in section-V and presents a number of examples. Section-VIII summarizes the paper. Emphasis in this paper is on preventive techniques relating to design-time compliance.

II. COMPLIANCE RISK MANAGEMENT

Fig. 1 shows that risk management is an essential component of internal control, while root-cause analysis is the bond between internal control functions and improved performance business processes.

Compliance risk management refers to the recognition, assessment, measurement, limiting and monitoring of compliance risks that are involved in core enterprise business processes and operational objectives. In general, typical risk factors may include the control environment, size of the organization, complexity, health and safety, environment, product/service failure, change integration, and results of previous reviews/audits. It is important to realize that not all risks are equal. Some risks are more likely to occur while others will have a greater impact. For example, risks to safety or security of individuals, data or personal information could have significant consequences. Once identified, the assessment regarding the probability and significance of each risk is critical. The risk assessment design should be understandable, consider relevant risk factors and, to the extent possible, be objective. Risk and compliance management must be incorporated right into planning, design and operational execution of business processes.

Compliance risk management is part of Enterprise Risk Management (ERM), which takes an integrated and holistic perspective on risks facing an organization and views risk management as a core competency. The Institute of Management Accountants defines ERM as a structured and disciplined approach, which aligns strategy, processes, technology, and knowledge with the purpose of managing all key business risks and opportunities [9].

The risk and compliance planning process, which is related to risk assessment in ERM, identifies the top risk categories and compliance initiatives that affect business strategy and performance. It pinpoints the risks and compliance objectives to be monitored. It lets business analysts and auditors specify the risk thresholds. For example, business analysts can define the context within which business risks are to be managed – including what business activities are to be assessed.

To understand the approach to compliance risk management, consider the Enterprise Risk Management framework of COSO [8], which is the de facto standard for establishing internal control systems. COSO breaks out internal environment into two parts (internal environment and objective setting) and risk assessment into three parts (event identification, risk assessment and risk response). The key components of COSO’s ERM solution, which relate to internal control and compliance risk management are:

1. Event identification, which identifies internal and external events that impact an organization achieving its objectives.
2. Risk assessment, which allows an organization to understand the extent to which potential events may impact objectives.
3. Risk response, which evaluates options to an identified risk and determines the course of action.
4. Control activities, which include policies and procedures, and directives.
5. Information and communication, which is the identification and dissemination of pertinent information in a form and timeframe that enables people to carry out their responsibilities.
6. Monitoring the effectiveness of components includes ongoing activities and/or separate evaluations and making modifications as necessary.

The above components constitute a sound basis for automated risk and compliance assessment and management the compliance framework introduced in this paper. They enable organizations identify and assess the impact of risk and compliance events on business objectives. It helps document deficiencies, identify issues for review and remediation, and prioritize effective response strategies.

Key risk indicators (usually associated with compliance KPIs) are aligned directly to business processes and overall corporate strategic objectives and continuously monitored to alert business analysts when risks exceed acceptable thresholds. Business analysts receive reports, and support for proven best practice risk processes and key risk indicators that help them identify and respond to risks. They can use tools that indicate early risk or identify frequently occurring risks. Leveraging these and other functions, business analysts can manage and mitigate their risks proactively.

III. BUSINESS PROCESS COMPLIANCE MANAGEMENT FRAMEWORK

Fig. 2 provides insight into an architectural approach to business process compliance management grounded on SOA principles. This figure was inspired by [10] and gives an operational view of the main components in Fig. 1.

The business process lifecycle starts with the analysis and design of the processes [11]. This involves the analysis of existing processes and the design of ‘to-be’ processes taking into account various factors, such as business objectives, risks, industry best practices, architectural frameworks, and compliance requirements. The processes can be modeled using a process definition notation, such as BPMN, followed by the construction of detailed-level executable business process and service specifications using, for instance, languages such as BPEL or XPDL. Once these business process specifications are tested and have reached a steady state, they are deployed and executed. Business process executions are subsequently monitored by tracking the progress of
individual process instances. In this way, information regarding their state and performance is acquired. Monitoring collects metrics relating to compliance constraints to ensure that business processes are performing in accordance with service-level objectives. Metrics ensure conformance, attempt to prevent or warn of conformance deviations, and measure non-conformance. Monitoring may therefore result in corrective actions or adjustment of business process definitions if SLA terms and parameter thresholds are violated. Adjustments and corrective actions could effect changes in business process components in an SOA-based application.

The compliance framework in Fig. 2 encompasses three main compliance assurance activities: design-time, run-time and offline analysis and monitoring. Design-time compliance verification involves the static verification of business process models against formal compliance rules. Design-time compliance verification and runtime monitoring are two important and complementary business process compliance checking phases. Their interaction provides a lifetime guaranteed compliance assurance. In design-time compliance verification, a process model is considered as compliant with the set of relevant compliance requirements if it allows only for the execution of process instances not violating these constraints [12]. Consequently, it is (theoretically) ensured that corresponding process instances are compliant. Design-time compliance verification supports the idea of detecting and resolving any compliance violations as early as possible before the actual business process execution, thus providing a preventive compliance support. Design-time compliance checking has lower verification costs compared to the subsequent dynamic verification phase. Hence, it is desirable to verify as much compliance requirements as possible during design-time. However, it is not always feasible to enforce compliance on a process models at design time. Some compliance requirements may require variable instantiations or runtime information that cannot be captured during design time. Runtime requirements are compliance requirements that cannot be checked during design-time and should be verified against running business process instances during runtime. Runtime compliance monitoring and offline compliance analysis and monitoring ensure an integrated, holistic view of compliance throughout the remaining phases of the business process lifecycle. During runtime compliance monitoring, the execution of business process instances is observed against compliance constraints and associated metrics. During runtime the execution of business processes is often suspended preventing violations to arise. Static and runtime compliance monitoring is preventive in nature aiming to detect violations before they occur. Offline monitoring, on the other hand, involves after-the-fact reporting. Possible violations and trends resulting from the analysis of the execution data with respect to (offline) compliance rules are presented on monitoring dashboards in the form of compliance KPIs.

A formal specification of constraints allows automated compliance verification and monitoring techniques to be applied. However, not all compliance requirements can be formally specified. For these cases, the compliance is
complemented by manual controls and inspections that involve full or partial human intervention. For example, checking if “a certain type of document is secured stored in locked cabinets” involves a manual control. The approach depicted in Fig. 2 presents only these assurance activities that employ automated verification and monitoring practices.

IV. COMPLIANCE MODEL

An integral element of the conceptual architecture for compliance management is the compliance model. This model connects two domains, business processes and compliance requirements, to facilitate traceability. Managing traceability involves mainly tracing compliance requirements back to their sources, or forward to the processes that enforce them [13]. Bi-directional traceability is important as it helps to recognize the implications of changing requirements and processes. It allows analyzing why a particular decision in a process was made and what the implications of changing these specifics are in relation to the compliance requirements.

A key element linking the compliance and the business process domains in Fig. 3 is the compliance target construct. A compliance target is an abstract concept representing a generic ‘object’ comprising compliance requirements. These are in the form of business processes or process elements. A control applies to compliance targets and their properties.

A business process compliance assessment is performed to verify and ascertain that an organization is designing and executing processes that satisfy the compliance requirements applicable to them. It involves checking (during design-time, runtime, and offline) whether compliance targets conform to applicable rules with the purpose of identifying if and how a target can be changed to make it (more) compliant.

Only those controls that can be implemented and checked effectively through automated compliance assurance should be formally specified into compliance rules. A rule usually takes the form of ‘if-then’ statement consisting of a set of conditions as its antecedents, and one or more conclusions. Business process elements and their attributes are the building blocks of the conditions and conclusions in rules. For example, a control mandating...
“orders above 100,000 $ must be checked by supervisors” can be formally specified using Linear Temporal Logic (LTL) as follows:
\[ G(\text{Order.Amount} > 100,000 \) \] 
\[ \rightarrow F(\text{CheckOrder.Role(Supervisor))} \]

Despite their undeniable value for automated compliance assurance, formal specifications are difficult for end-users to understand, specify and use. To help surmount the complexity of formalisms, we advocate the use of patterns that help compliance/business experts to specify controls in an intuitive semi-formal manner.

V. COMPLIANCE PATTERNS

Compliance patterns are high-level domain-specific templates used to represent desired proven compliance properties and constraints that apply to process segments to facilitate work with recurring compliance matters (expressions). These techniques exhibit a generic nature, are reusable and have a preemptive nature. They aim at guaranteeing that process instances are compliant to standards or regulations.

![Table 1 Compliance Patterns](image)

Compliance patterns have been used quite widely in literature. For example [15] propose an approach based on compliance patterns to compute the deviation of a given business process from a specific compliance pattern. In [16] the authors view business processes spanning organizational boundaries, with the constraints stated in business contracts in a form resembling compliance patterns. For this purpose a logic-based formalism that allows for the description of obligations, prohibitions, permissions and violations conditions in contracts is explored. Finally, in [17] the authors present a formalism for defining business process compliance checking that relies on control patterns. Control patterns constitute a generic and reusable solution to a specific problem and, therefore, can be used to ensure that BP models containing them are regulatory compliant.

In this paper we use the compliance patterns described in [18] and summarized in Table-1. As an example, if we consider a loan approval application, the expression “Create.Order LeadsTo Approve.Order” is built using the LeadsTo compliance pattern so that it respects the directive that requires ‘Approve.Order’ to follow ‘Create.Order’ activity. Compliance/business experts use these patterns to design pattern-based expressions as an intermediate specification between controls and formal compliance rules. These directives are formal language agnostic and can be easily transformed into a variety of formal statements.

The compliance patterns in Table-1 are divided into three categories:

- **Atomic Patterns**: these are adapted from Dwyer’s property specification patterns originally introduced in [19]. Atomic patterns deal with occurrence (exists, absent, universal in Table-1) and ordering constraints (precedes, leadsTo, XleadsTo in Table-1).

- **Composite Patterns**: these include patterns that are built up from combining two or more atomic patterns connected via logical operators, such as not, and, xor, imply, and iff.

- **Timed Patterns**: these include patterns that are used in conjunction with other compliance patterns to handle time-depended constraints.

- **Miscellaneous patterns** such as role segregation, substitution, pre-requisite and co-requisite patterns.

To exemplify the use of constraint patterns we use a relatively simple scenario representing an online product selling application. This scenario starts with a customer checking product information on a Website. Next, if the customer chooses a specific product, she submits an order along with her customer data. Following this, the sales application validates the order by contacting the credit

![Table 1 Compliance Patterns](image)
bureau to check the credit worthiness of the customer. Afterwards, the financial department creates the invoice and checks for payments. Finally, a delivery request is sent to the supplier.

Table 2 Typical Compliance Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Compliance Requirement</th>
<th>Pattern Representation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Sales order confirmations or cancellations are sent to customers after validating the order.</td>
<td>( \text{ValidateOrder}(x) ) \text{ LeadsTo } \text{SendConfirm}(x) \text{ MutexChoice } \text{SendCancel}(x) )</td>
<td>ValidateOrder for sales order ( y ) and customer ( x ) is followed by either sending a confirmation or cancellation to ( x ).</td>
</tr>
<tr>
<td>R2</td>
<td>Sales orders over a set threshold require approval by management before acceptance.</td>
<td>( \text{(SalesOrder}(y, \text{threshold}) \text{ exists}) \text{ Imply } \text{Approve}(y, \text{manager}) \text{ Precedes } \text{Accept}(y) )</td>
<td>If there is a salesOrder ( y ) that exceeds a threshold then Approve action performed by manager should precede Accept of ( y ).</td>
</tr>
<tr>
<td>R3</td>
<td>Segregation of duties is maintained between credit checking and cashing functions.</td>
<td>( \text{CreditCheck}(x) \text{ SegregatedFrom } \text{Cashing}(x) )</td>
<td>CreditCheck function for customer ( x ) should be segregated from the Cashing function for the same customer.</td>
</tr>
</tbody>
</table>

Table-2 shows typical compliance requirements related to the above scenario. Each compliance requirement is described in terms of: (i) an identifier, (ii) internalized compliance requirement, (iii) its representation in the form of compliance patterns, and, (iv) an explanation of its pattern representation.

VI. ROOT CAUSE ANALYSIS OF COMPLIANCE VIOLATIONS

A compliance violation in a business process definition may occur due to a variety of reasons and it is of upmost importance to provide the compliance expert intelligent feedback that reveals the root-causes of these violations and aids their resolution. This feedback should contain a set of rational explanations regarding the underlying reasons why the violation occurred and what strategies can be used as remedies.

In this section we analyze and present root-causes for each pattern in the compliance pattern taxonomy in section -V. More specifically, we investigate and report all possible causes of a violation of a compliance constraint represented by a specific pattern.

For this purpose, we have adapted the Current Reality Tree (CRT) technique from Goldratt’s Theory of Constraints (TOC) [20]. A Current Reality Tree is a statement of a core problem and the symptoms that arise from it. It maps a sequence of causes and effects from the core problem to the symptoms arising from one core problem or a core conflict. If the core problem is removed, each of the symptoms may be removed. Operationally the process works backwards from the apparent undesirable effects or symptoms to uncover or discover the underlying core causes [20]. The CRT has been chosen due to its simplicity and the ease of representation of the causes and effects.

A CRT usually starts with a list of problems called Undesirable Effects (UDEs), which represent negative or unacceptable conditions. They are also ‘effects’ because for most part they are caused by something else. The key question begins with 'why a violation occurs?' (at the root of the tree). The answer to this question will generate potential children of the UDE under consideration. For each child, which might be a UDE, the same “why” question is applied, and the answer is depicted as a deeper level in the tree. This process continues iteratively until the UDE under consideration is the root-cause of the problem (at the leaf level of the tree). Incoming connections to an UDE from its children are connected using a logical ‘or’ operator, unless otherwise specified.

The root of each CRT represents an undesirable effect with a UDE being a violation of a specific pattern. Hence, the root of each tree represents a violation of this pattern. Effects/Causes are drawn in a CRT using rectangles and are connected to each other using arrows forming a hierarchical tree of effects/causes. The root-causes are depicted as leaf rectangles of the tree. Fig. 4 shows CRTs representing atomic patterns. A state sequence in this figure is denoted by the sequence \( S_1, S_2, \ldots, S_n \) where: \( S_1 \) is the start state, \( S_n \) is the end state, and there is a transition relation from each \( S_i \) to \( S_{i+1} \), where \( 1 \leq i < n \). \( S_k, S_l \) and \( S_m \) are intermediate states, where \( 1 \leq k, l, m \leq n \).

Let’s consider as an example the violation of the ‘\( (P \text{ precedes } Q) \)’ pattern which coincides with the UDE of the preceeds current reality tree in Fig. 4. The answer to this question is: because \( (Q \text{ Exists is satisfied}) \) while \( (P \text{ exists is violated}) \) prior to it. This is depicted as the second level of the tree. Further analysis continues until reaching the root-causes of the problem, i.e. the leaves of the tree are reached.

For each leaf, the user is provided with guidelines regarding remedies to compliance violations. As shown by the Precedes CRT, the root-cause of the violation is mainly because \( P \) does not exist before \( Q \) (assuming that \( Q \) exists). In addition to this cause, it might be the case that \( P \) exists after \( Q \). Although this situation does not violate the constraint ‘precedes’, it gives an indication of misplacement in the specification that might have acted as a hidden reason that led to the violation. These cases are presented as caveats (or warnings) for the users to consider. In this particular case, the user might decide to interchange the occurrence of \( P \) with \( Q \) to resolve the violation. This is depicted in the CRTs as a squared bracket linked to the leaves, e.g. ‘Interchange each occurrence of \( P \) with \( Q \)’, where \( P \) and \( Q \) are business process activities that will be parameterized with actual activity names. In case the leaf is a composite pattern it will be replaced by its corresponding CRT.
The original CRT notation has been extended to distinguish hidden causes from root-causes. Rounded rectangles in the CRT represent the checks for identifying hidden causes, i.e., the cases which do not constitute the actual causes of violations but might act as underlying reasons for it. In case the operand in the leaf rectangle (root-cause) is a pattern expression, it is then replaced by its corresponding CRT. This process iterates continuously until all operands in the leaf rectangles of the tree are atomic BP elements.

Fig. 5 presents the application of the root-cause analysis and associated CRTs of the violations to compliance constraints R2 of the Loan Origination and Approval application, which are presented in Table-1 (rules R1 and R3 are omitted for conciseness). As illustrated in Fig. 5 the CRT of the violation to R2 is an ‘implies’ composition between the two atomic patterns exists and precedes.

VII. COMPLIANCE REQUEST LANGUAGE

To enable design-time compliance verification, we introduce a Compliance Request Language (CRL) for the formal specification of compliance requirements that is grounded on Linear Temporal Logic and is able to express the compliance patterns presented in section-V. Compliance patterns that are formulated on the basis of CRL are high-level abstractions of frequently used logical formulas.

The CRL in this paper is based on [21] and follows a thorough analysis of wide range of compliance legislations and frameworks including Basel II [22], Sarbanes-Oxley [23], IFRS [24], COSO [7], and COBIT [8], and a variety of relevant works on the specification of compliance requirements. CRL is focused on solving the complexity and non-monotonicity concerns associated with regulatory compliance. These can be defined as follows:

- Complexity: Refers to the usability, understandability and comprehensibility of CRL to be utilized by end-users.
- Non-monotonicity: A violation of some compliance requirements is not necessarily an error. Non-monotonic rules are open to violation to a certain extend and under specific conditions. Depending on the rigidity of the rule, the user may decide on the type of the rule and the exceptions under which a specific rule can be overridden if it is non-monotonic.

The Compliance Request Language represented using UML class diagram in Fig. 6. This meta-model illustrates the main syntactic constructs of CRL. As shown in Fig. 6, an expression comprises compliance patterns (patterns in short), operands and scope. Expressions can combine multiple (sub) expressions by using Boolean operators. The figure that the compliance pattern class is the core element of the language with each pattern that we described in section-V being its sub-type. The compliance pattern class is further sub-divided into three main subclasses of patterns: atomic, composite and timed.

Figure 6.

Figure 5. CRTS for the violation of rule R2 in Table-2.
form of LTL/MTL statements to enable their automatic verification against a business process model and associated compliance rules and regulations (stored in a compliance repository).

Figure 6 Compliance request language meta-model and expressions.

In the following we shall first introduce a more complex case study, which we shall use to exemplify the use of CRL expressions.

A. Loan Approval Case Study

The process flow in a simplified loan approval case study may be described as follows. Once a customer loan request has been received, a credit broker checks the customer’s banking privileges status. If privileges are not suspended, the credit broker accesses the customer information and checks if all loan conditions are satisfied. Next, a loan threshold is calculated, and if the threshold amount is less than 1M$, a supervisor checks the credit worthiness of the customer by outsourcing it to a credit bureau service. Next, the supervisor initializes the loan form and approves the loan. If the threshold amount is greater than 1M$, a manager is responsible for performing the same activities instead of the supervisor. Finally, the manager evaluates the loan risk, after which she normally signs the loan form and sends the form to the customer to sign.

B. Loan Approval Compliance Rules in CRL

In this section we shall provide a series of examples regarding compliance requirements expressed in CRL, which are apply for the loan approval case study introduced in the earlier section. We also identify the source of these requirements.

R1 “The customer should receive an automated email notification when his personal data is collected by the Credit Bureau service”. Source is 95/46/EC data protection directive. This compliance requirement is formulated in CRL as follows:

\[ ((\text{RequestBankInformation} \quad \text{Or} \quad \text{CheckCreditWorthiness}) \quad \text{XLeadsTo} \quad \text{Notify}(\text{Customer})) \]

The above request signifies that customer information is collected by conducting a credit bureau service to check the credit worthiness of the customer if this information is not already available. The pattern expression R1 ensures that the customer will be notified immediately after her data has been accessed.

R2 “If loan conditions are satisfied, the customer can check the status of her loan request at anytime (infinitely often)”. Source is internal bank policy. This compliance requirement is formulated in CRL as follows:

\[ \text{LoanConditions} = \text{‘True’ \quad WFairness \quad CheckLoanStatus} \]

R2 represents a typical weak fairness property, which can be expressed in CRL using the WFairness pattern. This pattern states that if an activity is continuously enabled (no temporary disabling) then it has to be executed infinitely often. This means that the customer is allowed to check her loan status ‘infinitely often’ in case loan conditions are satisfied.

R3 “The activity ‘Customer bank privilege check’ (to be performed by Credit Broker) should be segregated from ‘credit worthiness check’ (to be performed by a supervisor)”. Sources are Sarbanes-Oxley Sec. 404 and ISO 27002-10.1.3. This compliance requirement is formulated in CRL as follows:

\[ (\text{CheckCustomerBankPrivilage.Role('CreditBroker')} \quad \text{SegregatedFrom} \quad \text{CheckCreditWorthiness.Role('Supervisor')}) \]
R3 represents the typical segregation of duties compliance requirement and the SegregatedFrom composite pattern is sufficient to capture this requirement.

R4 “The branch office Manager checks whether risks are acceptable and makes either the final approval or rejection of the request”. Sources are Sarbanes-Oxley Sec. 404 and ISO 27002-10.1.3.

This compliance requirement is formulated in CRL as follows:

R4.1 (JudgeHighRiskLoan.Role(‘Manager’)) Exists R4.2 (JudgeHighRiskLoan LeadsTo (SignOfficiallyLoanContract MutexChoice DeclineDueToHighRisk))

R4.3 (SignOfficiallyLoanContract(Role1) Or DeclineDueToHighRisk(Role1) implies Role1 = ‘Manager’) isUniversal

Compliance requirement R4 can be represented as three expressions: Expression R4.1 checks if JudgeHighRiskLoan activity occurs. Then R4.2 checks whether it is followed by either SignOfficiallyLoanContract or DeclineDueToHighRisk (but not both or neither of them) by using the MutexChoice composite pattern. Finally, R4.3 ascertains that no other role than the manager can perform either SignOfficiallyLoanContract or DeclineDueToHighRisk activities.

R5 “The offer in the signed loan contract is valid for 7 working days and afterwards it is closed”. Source internal bank policy.

This compliance requirement is formulated in CRL as follows:

R5.1 ((SendSignedLoanContract LeadsTo ReceiveCustomerSignedContract) Within 7 days)
R5.2 ((CloseLoanContract Substitutes ReceiveCustomerSignedContract) ExactlyAfter 7 days)

R5 is a complex compliance requirement as it uses the timed composite pattern LeadsTo combined with the Within timed pattern (R5.1). It also combines both the composite pattern Substitutes and the timed pattern ExactlyAfter (R5.2). R5.1 states that SendSignedLoanContract should always be followed by ReceiveCustomerSignedContract within a time interval less than or equal to 7 days from the start of the first activity (SendSignedLoanContract). The second expression (R5.2) states that CloseLoanContract substitutes the absence of ReceiveCustomerSignedContract in the next state after 7 days have elapsed.

VIII. SUMMARY

Business processes—many of which are implemented as a SOA these days—form the foundation for all organizations, and as such, are impacted by laws, policies and industry regulations. Compliance management should be considered from the very early stages of the business process design, such that compliance constraints are seeded into service-enabled processes. To enable automated techniques for verifying and ensuring compliance, these compliance constraints should be grounded on a formal language.

In this paper, we introduced a declarative language based on high-level compliance patterns to allow business analysts express easily their compliance concerns. These patterns are used to generate formal expressions using temporal logic for meeting the associated compliance requirement and managing risks associated with business processes. Using patterns can significantly facilitate the work of the compliance and business process expert, as they are not burdened with the complexities of a formal language.

An interactive graphical prototype system has been developed to enable the specification of compliance requirements graphically using patterns in a drag-and-drop fashion and to automate the process of transforming these definitions into logical formulae. The prototype includes design-time compliance verification, and root-cause analysis of compliance, which were the main concern of this paper.

ACKNOWLEDGMENT

The author is grateful to Amal Elgamal and Oktay Turetken for their help, and suggestions during the preparation of this paper.

REFERENCES


