A Framework for the Systematic Comparison and Evaluation of Compliance Monitoring Approaches

Linh Thao Ly
Ulm University
thao.ly@uni-ulm.de

Fabrizio Maria Maggi
University of Tartu
f.m.maggi@ut.ee

Marco Montali
Free University of Bozen-Bolzano
montali@inf.unibz.it

Stefanie Rinderle-Ma
University of Vienna
stefanie.rinderle-ma@univie.ac.at

Wil M.P. van der Aalst
Eindhoven University of Technology
w.m.p.v.d.aalst@tue.nl

***Abstract*—To support the whole business process compliance lifecycle, one also needs to monitor the actual processes and not just check their design. Recently, many approaches have been proposed that utilize a broad range of constraint languages and techniques to realize compliance monitoring solutions. Due to this diversity, the comparison of existing approaches is difficult and consequently hampers the evaluation of which approaches are suitable for which application scenarios. This paper provides a framework to compare and evaluate existing compliance monitoring approaches. The framework is based on ten typical Compliance Monitoring Functionalities (CMFs). These have been derived using a systematic literature review and five case studies from different domains. Existing approaches are evaluated based on the CMF framework, resulting in a list of open questions and a discussion of new challenges in this field.**

***Keywords*-Process Mining; Business Process Compliance; Compliance Monitoring; Operational Support**

I. Introduction

For today's organizations, proving that their business processes comply with certain regulations has become a major issue. Even though the necessary compliance checks are predominantly viewed as a burden [1], business processes that violate regulations cannot only cause damage to an organization's reputation and thus harm the business success but can also lead to severe penalties and even legal actions [2], [3]. In general, compliance checking deals with the problem of understanding whether actual, logged traces represent behaviors that are aligned with the expected behaviors foreseen by compliance rules (such as business rules, guidelines and best practices, external laws and internal regulations).

Lifecycle support for process compliance comprises design time compliance checks, (online) compliance monitoring during runtime and post-mortem compliance analysis. Compliance monitoring is considered an important building block in this lifecycle [3] for reasons such as timely detection of non-compliance as well as provision of reactive and proactive countermeasures. In particular, compliance monitoring is related to *operational decision support*, which aims at extending the application of process mining techniques to on-line, running process instances, so as to *detect* deviations, *recommend* what to do next and *predict* what will happen in the future instance execution [4]. In this paper, we explicitly focus on compliance monitoring approaches.

Fig. 1 depicts the general compliance monitoring architecture. The business IT tier at the bottom is where the business processes, scattered over different systems (e.g., ERP or CRM tools), are executed. To gather events from the different systems and make them available for compliance monitoring, an event bus (or a similar infrastructure) is required. Such event architecture is often already in place in many companies as part of the middleware stack.

In general, compliance monitoring approaches are driven by two factors: (1) the *compliance rule language* that is used to specify the compliance requirements and (2) the *event format* the compliance checks are based on. Due to the possible heterogeneity of the data sources employed, an integrated target event format is desirable. In 2010, the IEEE Task Force on Process Mining has adopted XES (eXtensible Event Stream) [5] as the standard for storing, exchanging and analyzing event logs. Due to its extension mechanism, XES enables the consideration of many aspects relevant in the context of business process, e.g., time and organizational issues.

Lately, a multitude of compliance monitoring approaches has arisen that utilize a broad range of languages to express compliance requirements and different techniques to realize monitoring solutions. Due to this diversity, the comparison of existing approaches is difficult and consequently hampers the evaluation of which approaches are suitable for which application scenarios. Hence, the overall goal of this paper is to provide a framework for evaluating existing approaches based on typically used *Compliance Monitoring Functionalities* (CMFs) using a style similar to existing frameworks for process patterns [11], [12], [13].

From a methodological point of view, CMF engineering can be conducted following the explicitly specified and well-proven research methodology for engineering process change and time patterns as applied in [12], [13], i.e., by 1) defining selection criteria, 2) describing the data sources

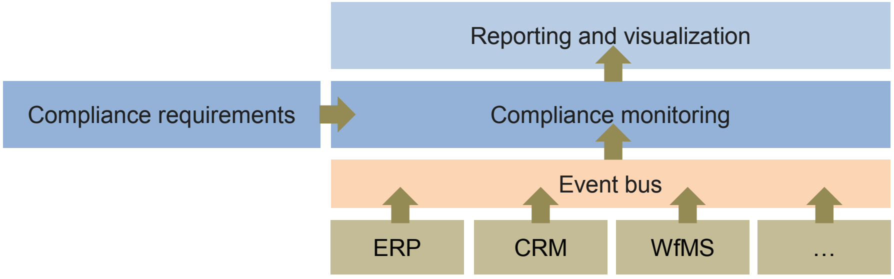

Figure 1. The general compliance monitoring architecture

Table I
CASE STUDIES

Domain	Project	URL	Reference
Health care	EBMC^2	ebmc2.univie.ac.at	[6]
Manufacturing	Adventure	www.fp7-adventure.eu	[7]
Higher education	HEP	www.wst.univie.ac.at/communities/hep/	[8]
Maritime safety	Poseidon	www.esi.nl/poseidon/	[9]
IT project management	SeaFlows	www.seaflows.de	[10]

and the process of data collection and 3) describing the CMF identification procedure. In the following, we briefly discuss how each of these steps was applied in this paper.

1) *Selection Criteria*: We focus on the elicitation of functionalities that are relevant for process compliance monitoring, i.e., the observation and enforcement of compliance constraints that are imposed over business processes during *run time*. This means that functionalities for compliance verification at process design time or compliance by construction are not considered. Moreover, we exclude monitoring of constraints that do not refer to any process activity, e.g., integrity constraints on process data elements [14].
2) *Data Sources and Collection*: The acquisition of the CMFs used two information sources: a systematic literature review on process compliance monitoring approaches [15], [16], [17], [10], [18], [19], [20], [21], [22], [23], [24] and case studies from five different domains as summarized in Table I. For the literature review, we harvested relevant keywords in the context of compliance monitoring and used them for literature selection by matching with title and abstract. The selection of the case studies was mainly driven by covering different application domains as well as the availability of the related processes and compliance requirements.
3) *CMF Identification Procedure*: We systematically collected CMF candidates from the different sources as described above. In principle, we considered every CMF discovered from these sources, but only after "cleaning" and/or aggregating/segmenting the set of CMF candidates. This resulted in a framework consisting of 10 CMFs. The cleaning and aggregating/segmenting task was conducted based on three rounds of expert discussions. The expert discussions were conducted along the 10 CMFs in a qualitative way. As the research area is further developing, more CMFs may be added to this framework over time.

The main contribution of this paper is the framework of ten typical functionalities for process compliance monitoring presented in Section II. Existing approaches including implementations are discussed along the proposed CMF framework in Section III resulting in a collection of open challenges and research questions. Section IV concludes the paper.

II. COMPLIANCE MONITORING FUNCTIONALITIES

We now introduce the framework of *Compliance Monitoring Functionalities* (CMFs). They have been engineered following the methodology as set out in [12], [13] and described in Section I. In brief, we first harvested CMF candidates from a literature study and the five case studies summarized in Table I. Based on expert discussions, these candidates were then cleaned and aggregated to the following ten CMFs. Each CMF is described by listing its *name*, a brief *problem* statement, a *description*, guidelines about the *evaluation criteria*, *implementation* hints, *related patterns* and *examples* of compliance rules illustrating the functionality.

CMF 1: Constraints may refer to time

Problem: The bulk of real-world compliance rules involve the combination of multiple activities or events in time. Hence, time is obviously one of the most important dimensions that a compliance rule language must tackle.

Description: Time-related conditions within compliance monitoring constraints may be *qualitative* or *quantitative* (i.e., metric time). This determines how temporal entities can be related to each other. A qualitative notion of time supports comparison between temporal entities without referring to their actual distance. Typical qualitative temporal patterns are "before" and "after". Such temporal relations are utilized, for example, to capture the fundamental ordering between events constrained by a compliance rule. In contrast to qualitative time constraints, metric (or quantitative) time constraints specify the distance between time entities. Metric constraints typically refer to deadlines, delays and latency constraints in compliance rules [25], [13].

Evaluation criterion: To fully support this functionality, the approach under evaluation must be able to monitor qualitative *and* quantitative time-related conditions.

Implementation: We briefly discuss the case of atomic timestamps, which are associated to a point-based algebra (see CMF 4 for a discussion on durative time entities). Temporal logics such as LTL, CTL* and μ-calculus [26] all adopt an inherent qualitative notion of time. Thus, they easily capture qualitative temporal relations such as "before" or "after". If not already inherent, such temporal relations can be introduced to the compliance rule language as the semantics of these relations can be defined over execution traces (and their linear/branching future). When metric times come into play, two approaches are typically followed for their representation: an *implicit* approach, embedding them inside temporal operators (like in real-time logics such as MTL and TLTL [27]) and an *explicit* approach, where explicit time variables are introduced and subject to arithmetic constraints (like in extensions of logic programming such as the Event Calculus [28]).

Related patterns: CMFs 2, 9.

Examples:
- (Qualitative time) For payment runs with amount beyond €10,000, the payment list has to be signed before being transferred to the bank and has to be filed afterwards for later audits [10].
- (Qualitative time) When an investor receives an amount of money, she becomes in charge of eventually investing it in bonds or in stocks and she cannot receive money anymore before the investment [15] (this is called "alternate response" in the Declare language [18], [29]).
- (Quantitative time) For Stage 1A patients, an appointment for sonography has to be made within 12 months (European Skin Cancer Treatment guideline [30]).
- (Quantitative time) A passenger ship leaving Amsterdam has to moor in Newcastle within 16 hours [9].

CMF 2: Constraints may refer to data

Problem: Compliance rules often not only define constraints on activities or events but also contain conditions on data processed in a business process.

Description: Data refer to the ability of the compliance rule language to not only target the control-flow component, but also the *data component*. This leads to data-aware compliance rules that can include constraints, requirements and expectations about data and their values. For what concerns the constraints' shape, a major distinction can be drawn between *unary data conditions* that just involve a single data object and *extended conditions* that possibly relate multiple data objects at the same time. Unary data conditions take the form $d \odot v$, where d is some data object, $\odot$ is a comparison operator and v is some value of d's domain. Extended data conditions compare instead expressions involving data objects and values. According to the classical data-related workflow patterns [31], we can further distinguish between different sources of data, namely *activity data*, i.e., data taken in input or produced by the activities of a business process and *case data*, namely data that are associated to a whole process instance and can be accessed/manipulated by all activity instances executed inside the case.[1]

Evaluation criterion: To fully support this functionality, the approach must be able to monitor unary data conditions *and* extended data conditions over activity *and* case data.

Implementation: Data-aware compliance rule languages typically employ variables to denote data objects and conditions to pose constraints over them. The main difference lies then in the domains of the data objects, as well as in the "shape" of such constraints. In order to support data-aware compliance rules, the corresponding compliance monitoring approach must be able to evaluate the truth of data conditions. This necessitates access to respective data sources within the process runtime environment.

Related patterns: CMFs 1, 3. In some approaches, time and resources can be dealt with at the same way as other types of data. CMF 9.

Examples:
- (Activity data) The value of the tumor marker CA-125 written by activity blood test before the first cycle of chemotherapy must be smaller than the value of the tumor marker CA-125 written by activity blood test executed after the third cycle of chemotherapy (ESMO clinical guideline [32]).
- (Case data/extended data condition) If a vessel (case) is of type fishing boat, the size of the boat is above 25 meters (100 tons) and it is located at 54 degrees of latitude and 8.5 degrees of longitude, it cannot be engaged in fishing [9].
- (Unary data condition) The value of data element hardCosts has to be smaller than 250 [33].
- (Comparison of multiple data objects) If the first test terminates with a certain result code, then all the consequent executions of the test should return the same result code [10].

CMF 3: Constraints may refer to resources

Problem: Compliance constraints often relate to organizational resources involved in the business process.

Description: Compliance rules often involve not only the control-flow and the data perspective but also the organizational perspective of a business process. This is particularly true for compliance rules stemming from legal sources. Resource-related conditions in compliance rules can be considered a special case of data-related constraints where the data refers to the resources involved. This is because resource-related information is often represented as case or activity data. Resource-aware compliance rules include constraints, requirements and expectations on resources (e.g., agents or roles) associated with activities or events. Similarly to data-related constraints, we can distinguish between *unary resource conditions* expressing expectations on specific resource properties in isolation

[1] Note that other data patterns like scope and block data are too fine-grained in the compliance monitoring setting.

and *extended resource conditions* relating multiple resources.

Evaluation criterion: To support this functionality, the approach under evaluation must be able to monitor unary resource conditions *and* extended resource conditions.

Implementation: Depending on the particular event model and the process runtime environment, language-wise constraints on resources may be dealt with in a similar manner as data-related constraints. Clearly, the evaluation of resource-related constraints requires access to resource information (such as originators, roles, groups) during process execution. This is supported by the XES organizational extension (cf. Section I).

Related patterns: CMFs 2, 9.

Examples:
- (Unary resource condition) Projects exceeding 5 days must be approved by the management.
- (Extended resource condition) Every closed project must be validated by a person who did not participate in the project (*4-eyes principle*).

CMF 4: Supporting not just atomic but also non-atomic activities

Problem: Activities in a process may be non-atomic, i.e., may have a duration. Hence, compliance rule languages must also support non-atomic activities.

Description: *Non-atomic activities* are durative activities whose execution spans across a time interval. While the execution of an atomic activity is associated to just a single event attesting that an instance of the activity has been "done", non-atomic activities are associated to multiple events and to a lifecycle that disciplines the allowed orderings among such events. The lifecycle contains at least the two event types *start* and *complete*. Moreover, often additional event types such as suspend, resume, abort are possible [5]. Compliance rules dealing with non-atomic activities follow either an *explicit* approach, talking about their multiple, atomic constitutive events, or an *implicit* approach, where the activities are mentioned as such without referring to their events.

Evaluation criterion: To fully support this functionality, the approach under evaluation must be able to monitor explicit *or* implicit conditions on non-atomic activities.

Implementation: Implementations differ depending on whether the explicit or implicit approach is adopted. With the explicit approach, the monitoring framework must be able to handle at least two types of information about each event: the activity it refers to and its type, which must be one of the event types constituting the activity lifecycle. Since the language directly tackles these constitutive atomic events, it typically relies on a *point-based algebra* to relate their relative position in time. The implicit approach directly targets activities and assumes that the time windows corresponding to their (non-atomic) executions can be reconstructed from the monitored event stream. Since the monitoring language predicates in this case over durative temporal entities, it relies on an *interval algebra* (such as the one by Allen [34]) to relate the execution of different activities over time. See [35] for a survey on temporal reasoning.

Related patterns: CMF 5, which extends CMF 4 with the ability of modeling and monitoring the activity lifecycle itself. CMF 9.

Examples:
- (Explicit) A flight payment cannot be canceled more than twice.
- (Explicit) An order creation cannot be completed until the customer registration is completed.
- (Implicit) Activity check project can be executed only while the project is under preparation.
- (Implicit) Activities check project and modify project must not overlap.

CMF 5: Supporting activity lifecycle

Problem: Non-atomic activities are associated to a lifecycle defining the allowed orderings of the constitutive events. Suitable monitoring mechanisms should be provided to check whether this lifecycle is indeed followed.

Description: The activity lifecycle describes the allowed executions of events constituting non-atomic activities. In particular, the lifecycle represents each constitutive event marking a step of such executions, in which states it can occur and to which state of the activity it leads. This latter aspect implicitly defines the allowed orderings of the constitutive events. The lifecycle is therefore mostly captured by a state chart (cf. ADEPT [36] or iUPC [37]). In general, multiple, independent executions of the same activity (i.e., activity instances) can occur inside a case. Each such instance corresponds to an instance of the activity lifecycle. A proper *correlation* mechanism is required to correctly manage the progressions of each lifecycle instance and, in particular, to associate a given event to the right corresponding lifecycle instance. For example, if two starts of some activity and two completions of the same activity occur during a case, it is necessary to identify to which start event each completion event refers to. From the monitoring point of view, (meta-)rules capturing the activity lifecycle and its instances can be used to check whether the activity executions contained in a given trace indeed comply with the expected lifecycle constraints.

Evaluation criterion: To fully support this functionality, the approach under study must capture the activity lifecycle *and* implement a correlation mechanism between events.

Implementation: Implementations of this CMF are possible if the compliance rule language supports: *(i)* the notion of "state", *(ii)* a correlation mechanism between events. Out-of-order events can either be ignored, or managed by putting the corresponding activity instance into a special "error" state, pointing out that a deviation from the expected lifecycle has been detected. Correlation can be realized by providing a special parameter used to identify the corresponding activity instance. This way, two events carrying the same identifier are recognized to be part of the same lifecycle. Events carrying different identifiers but referring to the same activity correspond to different parallel lifecycle instances.

Related patterns: CMFs 4, 9.

Examples:
- (Activation) A start event creates an activity instance and puts it into the "active" state.
- (Completion) Each completion event moves its associated activity instance to the "completed" state, provided that the instance is currently "active".
- (Balance start/complete events) For every activity instance, each start event has a single corresponding completion or cancelation event.

CMF 6: Multiple-instances constraints

Problem: There may be multiple instances of the same compliance rule in a trace due to multiple, possible parallel occurrences of the involved activities. Monitoring at the instance level allows for tracking fine-grained compliance rules.

Description: When compliance rules are able to constrain time in a quantitative way (CMF1) and/or data (CMF2) and/or resources (CMF3), the same compliance rule can be activated multiple times, as multiple events referring to the activities targeted by the rule occur, each with its own timestamp, data and resource information. In fact, each of such events provides a specific "context" for the compliance rule. This context is then used to instantiate the temporal/data/resource conditions possibly associated with the compliance rule. Consider, for example, the rule stating that *every time an order O is closed by the client, then order O must be eventually delivered by the warehouse*. Clearly, the constraint is instantiated for each specific closed order and each instance has its own evolution depending on events specific for this order. For example, it could happen that two orders are closed but only one is delivered. In this case, two instances of the compliance rule should be generated by the monitoring framework, then judging one of them as satisfied and the other one as violated. A discussion on multiple instances handling can be found in [10], [25].

Evaluation criterion: To support this functionality, the approach under evaluation must be able to monitor multiple instances based on metric time *and/or* data *and/or* resources.

Implementation: Supporting multiple instances of a compliance rule requires mechanisms to discriminate between different rule activations. This can be achieved by precisely characterizing which information (time, data, resources) contributes to define the "context" of the rule (see the examples below) and which are the events that create separate instances of the rule by filling this context with specific values. Each observed rule instance has to be associated with a separate compliance state in order to assess compliance at the rule-instance level.

Related patterns: CMFs 1, 2, 3, 9. Notice that CMF 5 and 6 are both concerned with introducing fine-grained analysis of compliance rules, but while CMF 5 focuses on multi-instance support for tasks, CMF 6 deals with multi-instance support for compliance rules. Hence, it may be the case that one of the two CMFs is supported, while the other is not.

Examples:
- (Multiple instances based on timestamps) Every final submission has to be corrected within 6 weeks. Here, every submission cycle termination creates an instance of the compliance rule determined by its timestamp t; the instance then checks that the correction occurs between t and $t+6$, assuming a granularity of weeks [8].
- (Multiple instances based on data and resources) The carbon footprint of a supplier must not exceed a value of x. Depending on the number of suppliers modeled as resources, the constraint is instantiated multiple times. If suppliers can be added during runtime, the number of constraint instantiations will increase accordingly.

CMF 7: Ability to reactively detect and manage compliance violations

Problem: If a violation is tailored to logical inconsistency, the computed answer of a monitor approach would simply be repeatedly "non compliant". However, a compliance monitoring approach may accommodate a variety of additional advanced features (besides detection) to continue the monitoring after a violation takes place, give feedback to the user and suggest compensation actions.

Description: In the context of reactive detection and management of compliance violations, these factors can be exploited to characterize the degree of support provided by a compliance monitoring approach:

- *Detection*, the ability to detect compliance violations.
- *Continuous monitoring*, the ability to continue monitoring after a violation.
- *Feedback*, the ability to provide detailed compliance reports (see CMFs 9 and 10).
- *Recovery and compensation* mechanisms, used to react to a violation with proper countermeasures.

Evaluation criterion: To support this functionality, the approach under evaluation must implement one or more recovery mechanisms to guarantee continuous monitoring. In addition, the approach must provide fine-grained feedback to the user.

Implementation: As for recovery and compensation, an added feature of the compliance rule language is the ability of dealing with violations. An event violating a rule can be used to contextualize it, making the rule active only when some violation takes place. This kind of rule represents a form of recovery or compensation, which introduces further constraints/requirements upon a violation. This can be realized, for example, by means of the so-called *contrary-to-duty* operator [38] to the compliance rule language. Continuous monitoring can be a challenge for logic-based approaches (e.g., [18]) as the approach must be able to tolerate inconsistencies to continue monitoring after a violation occurred. In [15] the authors introduce some recovery capabilities to realize different strategies for continuous monitoring showing that automata based approach are able to accommodate sophisticated recovery mechanisms.

Related patterns: CMFs 6, 8, 9, 10.

Examples:
- (Recovery and compensation) Generally, the patient has to formally confirm that she has been informed about risks prior to invasive treatments. If this is not the case (e.g., in emergency cases), this has to be documented and the patient has to be informed about the treatment risks afterwards (*contrary-to-duty* obligation).

CMF 8: Ability to pro-actively detect and manage violations

Problem: While recovery and compensation measures may be applied when detecting a violation, the violation itself cannot be undone. To prevent possibly costly compensation, a compliance monitoring approach should be able to provide support for pro-actively detect and manage possible compliance violations.

Description: Pro-active support includes detecting future violations and mechanisms for preventing violations. Future violations are violations whose source is not yet explicitly contained in the trace. They can be detected by implicit violations caused by currently conflicting rules. The presence of conflicting rules identifies violations that cannot be revealed by considering each compliance rule in isolation, but only by merging the contribution of two or more compliance rules. The early detection of such future compliance violations enables timely preparation of recovery and compensation actions. Support for preventing violations refers to the ability of a compliance monitoring framework to provide assistance for complying with imposed rules before compliance violations become manifest. This comprises, for example, predictions and recommendations of activities to be executed next in order to preserve compliance.

Evaluation criterion: To support this functionality, the approach under evaluation must implement mechanisms for the early detection of conflicting conditions *or* provide the user with recommendations about what to do next to avoid violations.

Implementation: Future violations as described can be detected when considering the interaction of all imposed compliance rules. A typical task is evaluating whether the compliance rules are not conflicting a-priori, i.e., that the whole set of rules admits at least one compliant execution trace. However, compliance rules that are not conflicting in general may still become conflicting at some point during the process execution. Thus, checking of compliance rules at design-time or per individual constraint, is not sufficient for detecting all types of future violations. It should be noted that supporting such implicit violations can become quite costly and the cost grows with the amount of rules involved. For expressive compliance rule languages, this becomes even undecidable. The identification of suitable decidable compliance rule patterns for data- and time-aware compliance rules is still an open challenge. To avoid violations in a running process instance, it is also possible to give recommendations about what to do next by exploiting complete cases stored in event logs (e.g., using process mining techniques).

Related patterns: CMFs 7, 9, 10.

Examples:
- (Early detection of a violation) Every time an order is delivered, the warehouse must be replenished. If the replenishment truck is broken, the warehouse cannot be replenished. Consider an execution where the truck is broken and the order delivered. Approaches able to detect conflicts among rules would in this case point out an (implicit) violation: the first constraint requires a replenishment and the second forbids it.
- (Predictions and recommendations) Requests for building permits need to be handled within 3 months. Based on historic information, i.e., comparing a request currently being handled with earlier requests, one can predict the remaining processing time. A counter measure is taken if the predicted remaining processing time is too long.

CMF 9: Ability to explain the root cause of a violation

Problem: Key to the practical application of a compliance monitoring approach is its ability to pinpoint the root cause of a compliance violation beyond providing the counterexample that resulted in the violation. This is particularly true when a compliance rule can be violated in multiple ways or multiple rules are involved in a violation.

Description: *Root cause analysis* enables to diagnose the root cause of a compliance violation, e.g., by isolating the responsible event occurrences or the involved compliance rules. Note that this kind of analysis is far from trivial and sometimes could lead to multiple possible explanations or to no explanation at all. Consider, for example, the case of a sequence of events that culminates in the expiration of a deadline: isolating the responsible events in this case is impossible in general. Similarly, as discussed in [16] there can be multiple sets of compliance rules that are involved in a violation at the same time and therefore fine-grained analysis is needed to identify the minimal set(s) of conflicting rules. Beside the root cause analysis itself, it is also of utmost importance to provide suitable ways for communicating the result of the analysis to the end users in a comprehensible and intuitive manner [10].

Evaluation criterion: To support this functionality, the approach must implement mechanisms for root cause analysis.

Implementation: For future research, effort should be taken to provide diagnostics and proactive recommendations based on the identification of the root cause of a violation as this is so far addressed by only few approaches [10], [17], [16].

Examples:
- (Root-cause of a violation within one rule) After the component test is finished, an integration test has to be conducted. Conceivably, it is required that the component is not changed between these tests (as then a new component test becomes necessary). If the component test is finished and yet the component is changed before the integration

test is finally carried out, clearly the constraint is violated. However, the root-cause is not that no integration test was conducted but that the component was modified before the integration test [10].

- (Root-cause of a violation involving multiple rules) If medicine A is administered, follow-up medication with B becomes necessary. If medicine C is administered, follow-up medication with D is needed. Due to an incompatibility, B and D must not be administered together. Even though there is no incompatibility between A and C, if tasks administer A and administer C both occur in a process, this will cause a violation.

CMF 10: Ability to quantify the degree of compliance

Problem: Compliance metrics and indicators should be employed by a monitoring framework in order to provide aggregated feedback to the users, summarizing the detailed information computed for each compliance rule.

Description: The practical feasibility of a compliance monitoring approach also relies on its ability to give practitioners a sense of the compliance situation. For that, crisp approaches associating two possible truth values to each compliance rule, representing whether it is satisfied or violated, is not sufficient. In contrast to crisp compliance characterization, fuzzy approaches allow for a range of values to capture the "degree of compliance" of the running trace with respect to a compliance rule. In this respect, we differentiate between approaches that *discretize* the possible truth values from approaches that adopt *continuous* distributions between 0 (violation) and 1 (satisfaction).

Evaluation criterion: To support this functionality, the approach under evaluation must be able to characterize the "healthiness" of a running trace through metrics.

Implementation: A typical approach to quantify the degree of compliance is to "count" the number of violations and devise meaningful metrics that give a measure about the *overall compliance degree* of a running process instance. This is particularly effective when multiple-instances are managed (cf. CMF 6). More fine-grained metrics can be devised by using detailed information about individual violations. Approaches using a continuous scale need to calculate a "degree" of compliance, rather than simply providing a yes/no answer. E.g., in the case of a deadline, a matching function could assign different noncompliance weights to traces missing the deadline, depending on the amount of time that exists between the deadline and the (late) event occurrence.

Related patterns: CMFs 1, 2, 3, 7.

Examples:

- (Metrics) A vessel cannot be not under command, a vessel with one occurrence of not under command is more "healthy" than a vessel with nine occurrences of not under command [9].
- (Fuzzy) A passenger ship leaving Amsterdam has to moor in Newcastle within 16 hours. It is desirable to judge with different degrees of violation a ship arriving in Newcastle after 16 hours and ten minutes and a ship arriving in Newcastle after 18 hours [9].

III. Evaluation

We evaluated compliance monitoring approaches using the 10 CMFs presented in this paper. We focused on compliance monitoring approaches, which mainly address compliance checks during the process execution. These approaches are different from other approaches that can be used in other phases of the process lifecycle such as process design (e.g., [39]) or the modeling of compliance constraints (e.g., [1], [40]) and trigger specific questions. For example, monitoring is carried out with actual data and by considering finite, evolving prefixes of event traces.

Furthermore, we selected the approaches to be evaluated based on the degree of detail on concepts provided in publications. In fact, a certain degree of detail (for example, in the used compliance rule languages) is necessary in order to properly evaluate the approaches through our framework. In addition, in order to compare them in terms of performance and effectiveness, toolset availability is crucial. The results of the evaluation are shown in Table II, where "-", "+" and "+/-" indicate functionalities that are not supported, supported and partially supported from the conceptual viewpoint. Bold emphasizes cases for which an implementation is *publicly* available.

Mobucon LTL [15], [16] deals with a qualitative notion of time (being based on LTL) but it does not support constraints concerning metric time. Mobucon LTL monitors finite-trace LTL constraints through deterministic finite state automata. Therefore, it does not tackle constraints referring to data and resources (ranging over finite state) because of the state space explosion problem. With this approach, it is possible to express rules on non-atomic activities but it does not fully support the monitoring of activity lifecycle. Indeed, with this approach it is possible to associate an event type to each occurrence of an activity. However, a correlation mechanism to link different events belonging to the lifecycle of the same activity cannot be defined. This is, again, related to the impossibility for an automata-based approach of monitoring constraints referring to data. Indeed, the most natural way of implementing such a correlation mechanism would be to connect events with the same value for a certain data (e.g., an activity ID). Mobucon LTL provides proactive support. Indeed, it is able to detect violations caused by the interplay of two or more constraints. The approach has been implemented and deployed as a ProM operational support plug-in.[2] The tool only supports simple metrics for quantifying the degree of compliance of a case [41].

In [41], the authors also provide ways for quantifying the degree of compliance in Mobucon EC [25]. However, also in this case, more sophisticated measurements for a thorough compliance evaluation are still missing. The core functionalities of Mobucon EC have been implemented in Prolog and are publicly available.[3] The integration with the operational support backbone of ProM is an ongoing effort. Being based on the Event Calculus, it

[2] http://www.win.tue.nl/declare/mobucon/

[3] https://www.inf.unibz.it/~montali/tools.html

Table II
CMFs and evaluation of some monitoring approaches

Approach	CMF 1 time	CMF 2 data	CMF 3 resources	CMF 4 non-atomic	CMF 5 lifecycle	CMF 6 multi-instance	CMF 7 reactive mgmt	CMF 8 proactive mgmt	CMF 9 root cause	CMF 10 compl. degree
Mobucon LTL	+/-	-	-	+	-	-	+	+	+	+/-
Mobucon EC	+	+/-	+	+	+	+	+	-	+/-	+/-
ECE Rules	+	+/-	+	+	-	-	+	-	+/-	+
Supervisory Control Theory	+/-	-	+	+	+	-	-	+	-	-
SeaFlows	+/-	+/-	+/-	+	+/-	+	+	+	+	+/-

can easily accommodate explicit (metric and qualitative) time constraints and activity data. Accordingly, it manages multiple constraint instances, with a limited form of root cause analysis (it is possible only to associate violations to compliance rule instances). Currently, case data is not supported. Due to the richness of the language, the framework cannot afford to support pro-active detection of violations, but only violations that are explicitly present in the monitored trace. In particular, it can only check the trace of events collected so far, but not analyze the possible future executions.

Similar limitations also apply to ECE rules [42]. Note that ECE rules are a domain-independent approach that was not specifically tailored for business process monitoring. Therefore, functionalities like support for case data and activity lifecycle were simply not investigated (this is matter of ongoing work). ECE rules can deal with both atomic and non-atomic temporal entities, capturing qualitative and metric time constraints, as well as point-based and interval-based ones. Two key features characterize ECE rules. First, they support an imperfect (i.e., fuzzy and probabilistic) matching between expected and occurred events and hence deal with several fine-grained degrees of compliance. Second, expected events can be decorated with countermeasures to be taken in case of a violation, hence providing first-class support for compensation mechanisms.

The framework described in [17] is based on Supervisory Control Theory. This approach allows for the definition of constraints on resources but, in general, it does not support data conditions. Through this approach, it is possible to supervise the process-aware information system by “blocking” those events that would lead to a violation. This can be considered as a very sophisticated form of proactive violation management, which is applicable only when the process-aware information system can be (at least partially) controlled by the monitor. Since violations are prevented, the framework does not directly consider the problem of reactive management nor violation explanation.

SeaFlows [10] aims at encoding compliance states in an easily interpretable manner to provide advanced compliance diagnosis. While qualitative time is supported in the graphical compliance rule language of SeaFlows, quantitative time is not supported. It supports unary data and resource conditions over activity and case data. However, extended data/resources conditions are not yet addressed. Even if the activity lifecycle is not addressed, SeaFlows provides a correlation mechanism for events that can be exploited for providing activity lifecycle support. Proactive support in terms of guiding the process execution to avoid compliance violations is supported. Even if the compliance states of multiple compliance rule instances can be aggregated to provide an overall compliance degree, the framework still lacks a deeper investigation about metrics.

As can be seen from Table II, several approaches deal with CMFs 1 - 4. Further work needs to be done in terms of supporting extended data and metric time conditions. Fundamental issues such as detection of violations are already well-understood as well. Better user support in detecting and managing compliance violations and explaining reasons for compliance violations should be provided (CMFs 7 - 9). Activity lifecycle and multiple-instances constraints are also not well supported (CMFs 5 - 6). Referring to CMF 10, sophisticated compliance metrics are still missing.

IV. Conclusion

In this paper, we provided a framework for the systematic evaluation of compliance monitoring approaches in the business process management area. Based on the evaluation framework, we analyzed five state-of-the-art approaches. Our evaluation pointed out that compliance monitoring poses challenges related to representation languages, semantics of compliance, monitoring technologies and reasoning facilities. None of the five approaches fully supports more than six CMFs and most approaches are not supported by publicly available software tools. As far as the language perspective is concerned, further research is needed in order to devise next-generation compliance rule languages that are able to suitably mediate between expressiveness and tractability. There are obvious relations between compliance monitoring, query languages and data stream analysis, so as with related fields such as policy management and enforcement (see, e.g., [43]). We believe that the cross-fertilization of the different research areas would be beneficial. Moreover, there is a need for benchmark data sets to compare the different approaches and tools with respect to performance and effectiveness.

Acknowledgment

The work presented in this paper has been partly funded by the EU Project ACSI (FP7-257593) and by the Austrian Science Fund (FWF):I743

REFERENCES

[1] S. Sadiq, G. Governatori, and K. Naimiri, "Modeling control objectives for business process compliance," in *Int'l Conf on Business Process Management*, 2007.

[2] N. S. Abdullah, S. W. Sadiq, and M. Indulska, "Emerging challenges in information systems research for regulatory compliance management," in *CAiSE 2010*, vol. 6051, pp. 251–265.

[3] M. E. Kharbili, S. Stein, I. Markovic, and E. Pulvermller, "Towards a framework for semantic business process compliance management," in *Proc. Int'l Workshop on Governance, Risk and Compliance - Applications in Information Systems (GRCIS'08)*, 2008, pp. 1–15.

[4] IEEE Task Force on Process Mining, "Process mining manifesto," in *Proc. BPM Workshops*, ser. Lecture Notes in Business Information Processing, vol. 99. Springer, 2011, pp. 169–194.

[5] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst, "XES, XESame, and ProM 6," in *Information Systems Evolution - CAiSE Forum 2010*, vol. 72, pp. 60–75.

[6] M. Binder, W. Dorda, G. Duftschmid, R. Dunkl, K. A. Fröschl, W. Gall, W. Grossmann, K. Harmankaya, M. Hronsky, S. Rinderle-Ma, C. Rinner, and S. Weber, "On analyzing process compliance in skin cancer treatment: An experience report from the evidence-based medical compliance cluster (ebmc2)," in *CAiSE 2012*, pp. 398–413.

[7] S. Schulte, D. Schuller, R. Steinmetz, and S. Abels, "Plug-and-play virtual factories," *IEEE Internet Computing*, vol. 16, no. 5, pp. 78–82, 2012.

[8] L. T. Ly, C. Indiono, J. Mangler, and S. Rinderle-Ma, "Data transformation and semantic log purging for process mining," in *Advanced Information Systems Engineering - 24th International Conference, CAiSE 2012*, 2012, pp. 238–253.

[9] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst, *Analyzing Vessel Behavior using Process Mining*, 2012, ch. Poseidon book, to appear.

[10] L. T. Ly, S. Rinderle-Ma, D. Knuplesch, and P. Dadam, "Monitoring business process compliance using compliance rule graphs," in *OTM Conferences (1)*, 2011, pp. 82–99.

[11] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[12] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features - enhancing flexibility in process-aware information systems," *DKE*, vol. 66, no. 3, pp. 438–466, 2008.

[13] A. Lanz, B. Weber, and M. Reichert, "Workflow time patterns for process-aware information systems," in *BPMDS 2010*, no. 50, 2010, pp. 94–107.

[14] S. Rinderle-Ma and J. Mangler, "Integration of process constraints from heterogeneous sources in Process-Aware information systems," in *EMISA 2011*, pp. 51–64.

[15] F. Maggi, M. Montali, M. Westergaard, and W. van der Aalst, "Monitoring business constraints with linear temporal logic: An approach based on colored automata," in *BPM 2011*.

[16] F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst, "Runtime verification of LTL-based declarative process models," in *RV 2011*, vol. 7186, pp. 131–146.

[17] E. A. P. Santos, R. Francisco, A. D. Vieira, E. F.R. Loures, and M. A. Busetti, "Modeling business rules for supervisory control of process-aware information systems," in *Business Process Management Workshops*, 2012, vol. 100, pp. 447–458.

[18] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari, "Declarative specification and verification of service choreographiess," *TWEB*, vol. 4, no. 1, 2010.

[19] S.-M.-R. Beheshti, B. Benatallah, H. Motahari-Nezhad, and S. Sakr, "A query language for analyzing business processes execution," in *BPM*, 2011, vol. 6896, pp. 281–297.

[20] A. Awad and M. Weske, "Visualization of compliance violation in business process models," in *Business Process Management Workshops*, 2010, vol. 43, pp. 182–193.

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

[22] T. Holmes, E. Mulo, U. Zdun, and S. Dustdar, "Model-aware monitoring of SOAs for compliance," in *Service Engineering*. Springer, 2011, pp. 117–136. [Online]. Available: http://eprints.cs.univie.ac.at/2842/

[23] A. Birukou, V. D'Andrea, F. Leymann, J. Serafinski, P. Silveira, S. Strauch, and M. Tluczek, "An integrated solution for runtime compliance governance in SOA," in *ICSOC*, 2010.

[24] M. Weidlich, H. Ziekow, J. Mendling, O. Günter, M. Weske, and N. Desai, "Event-based monitoring of process execution violations," in *CAiSE*, 2011.

[25] M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. van der Aalst, "Monitoring Business Constraints with the Event Calculus," *ACM Transactions on Intelligent Systems and Technology*, To appear.

[26] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model checking*. Cambridge, MA, USA: The MIT Press, 1999.

[27] R. Alur and T. A. Henzinger, "Real-time logics: complexity and expressiveness," *Information and Computation*, vol. 104, pp. 35–77, 1993.

[28] R. A. Kowalski and M. J. Sergot, "A logic-based calculus of events," *New Generation Computing*, vol. 4, no. 1, pp. 67–95, 1986.

[29] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, "Declare: Full support for loosely-structured processes," in *IEEE EDOC*, 2007, pp. 287–300.

[30] C. Garbe, K. Peris, A. Hauschild, P. Saiag, M. Middleton, A. Spatz, J. Grob, J. Malvehy, J. Newton-Bishop, A. Stratigos *et al.*, "Diagnosis and treatment of melanoma: European consensus-based interdisciplinary guideline," *European Journal of Cancer*, vol. 46, no. 2, pp. 270–283, 2010.

[31] N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst, "Workflow data patterns: Identification, representation and tool support," in *ER 2005*, vol. 3716, pp. 353–368.

[32] ESMO, "http://www.esmo.org/education-research/esmo-clinical-practice-guidelines.html." [Online]. Available: http://www.esmo.org/education-research/esmo-clinical-practice-guidelines.html

[33] M. T. G. López and R. M. Gasca, "Run-time auditing for business processes data using constraints," in *Business Process Management Workshops*, 2010, pp. 146–157.

[34] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of the ACM*, vol. 26, no. 11, 1983.

[35] F. A. Schreiber, "Is Time a Real Time? An Overview of Time Ontology in Informatics," *Real Time Computing*, vol. F 127, pp. 283–307, 1994.

[36] M. Reichert and P. Dadam, "ADEPT flex - supporting dynamic changes of workflows without losing control," *Journal of Intelligent Information Systems*, vol. 10, no. 2, p. 93129, 1998.

[37] J. Mangler and S. Rinderle-Ma, "Rule-Based synchronization of process activities," in *IEEE Int'l Conf on E-Commerce Technology (CEC'2011)*, 2011, pp. 121–128.

[38] H. Prakken and M. J. Sergot, "Contrary-to-duty obligations," *Studia Logica*, vol. 57, no. 1, pp. 91–115, 1996.

[39] G. Governatori, Z. Milosevic, and S. Sadiq, "Compliance checking between business processes and business contracts," *IEEE Int'l Enterprise Distributed Object Computing Conference (EDOC)*, pp. 221–232, 2006.

[40] M. Hashmi, G. Governatori, and M. T. Wynn, "Business process data compliance," in *Rules on the Web: Research and Applications*. Springer, 2012, pp. 32–46.

[41] F. M. Maggi, M. Montali, and W. M. P. van der Aalst, "An operational decision support framework for monitoring business constraints," 2012.

[42] S. Bragaglia, F. Chesani, P. Mello, M. Montali, and D. Sottara, "Fuzzy conformance checking of observed behaviour with expectations," in *AI*IA 2011*, vol. 6934, pp. 80–91.

[43] C. Ringelstein and S. Staab, "Papel: Provenance-aware policy definition and execution," *IEEE Internet Computing*, vol. 15, no. 1, pp. 49–58, 2011.