BP-MaaS: A Runtime Compliance-Monitoring System for Business Processes

Ahmed Barnawi[1], Ahmed Awad[2], Amal Elgammal[2], Radwa Elshawi[3], Abduallah Almalaise[1], and Sherif Sakr[4]

[1] King Abdulaziz University, Saudi Arabia {ambarnawi,aalmalaise}@kau.edu.sa
[2] Cairo University, Egypt {a.gaafar,a.elgammal}@fci-cu.edu.eg
[3] Princess Nourah Bint Abdulrahman University, Saudi Arabia
rmelshawi@pnu.edu.sa
[4] King Saud bin Abdulaziz University for Health Sciences, Saudi Arabia
University of New South Wales, Australia
ssakr@cse.unsw.edu.au

Abstract

Today's enterprises demand a high degree of compliance in their business processes to meet diverse regulations and legislations. Several industrial studies have shown that compliance management is a daunting task, and organizations are still struggling and spending billions of dollars annually to ensure and prove their compliance. In this demonstration, we present, *BP-MaaS* (**B**usiness **P**rocess **M**onitoring-**as**-**a**-**S**ervice), a *runtime* business process compliance-monitoring framework which incorporates a wide range of expressive high-level compliance patterns for the abstract specification of runtime constraints. The framework provides the end-users with a friendly interface for modeling their compliance monitoring rules. Compliance monitoring is achieved by means of anti-patterns, a novel evaluation approach that is independent of any underlying technology. The applicability, feasibility and utility of *BP-MaaS* is validated by applying the approach on two real-life large-scale case studies in the banking domain.

1 Overview

Compliance monitoring at process execution time is of crucial importance and it complements the design-time checking with techniques to detect violations that are hard or even infeasible to address at the earlier stages of the process lifecycle. For example, time span constraints between tasks can only be checked at runtime, as time-related information is usually not available during prior phases. In this demonstration, we present, *BP-MaaS*, a runtime business process compliance-monitoring framework which adopts a rich and wide set of compliance patterns for the abstract specification of monitoring requirements, spanning the four structural facets of BPs; i.e. control flow, data, employed resources and timing constraints [1], [2]. The monitoring evaluation approach is based on the notion of *anti-patterns* [1], a novel evaluation technique that operates by continuously monitoring process execution events and looking for sequences of events or lack of events that may indicate that a violation has occurred or possible to occur in the future. These violation scenarios are denoted as *anti-patterns*. The main features/functionalities provided by BP-MaaS are:

- a *graphical compliance requirements builder* that implements the compliance patterns in an intuitive and user-freindly manner, and enables process designers to build pattern-based expressions in a drag-and-drop fashion
- a *mapping scheme* that automatically maps graphical pattern-based expressions, stored as XML, into the underlying formalisms of the complex event processing backend engine
- a novel *monitoring evaluation* approach based on the notion of anti-patterns
- a *monitoring dashboard*, which provides updated information about violations in process instances, the rule/pattern that has been violated and contextual information of the sequence of events that yields to the violation to facilitate its prevention/resolution

As a proof-of-concept of one possible realization of the anti-patterns monitoring approach, we have implemented *BP-MaaS* by using Complex Event Processing (CEP) technology [3], and applied the approach on two large-scale case studies in the banking domain. The first case study is borrowed from the EU-funded project COMPAS, which has been provided by COMPAS industrial partners, and addresses the loan approval business scenario. While the second case study is concerned with anti-money laundering, which has been developed in the Governance, Risk and Compliance Technology Centre (GRCTC) as a part of a large-scale project which is funded by the Irish government. The evaluation study [1, 2] has revealed that our approach is sufficiently expressive to capture a wide range of real-life compliance requirements with full support of 70% of the requirements being considered [1].

2 Architecture and Implementation

Fig. 1 illustrates the architecture of the *BP-MaaS* framework which consists of the following main components:

Compliance Repository. This is a central repository that stores and maintains business process and compliance-related specifics, where business and compliance concepts are semantically aligned.

Compliance Rule Editor. This editor provides a graphical representation of the compliance patterns, where *Compliance patterns* are defined as high-level abstractions of frequently used compliance requirements, which help non-technical users to abstractly represent desired properties and constraints [2].

COMPAS: http://www.compas-ict.eu

PriceWaterHouseCoopers the Netherlands (http://www.pwc.nl/) and Thales Service France (https://www.thalesgroup.com/)

GRCTC: http://www.grctc.com/

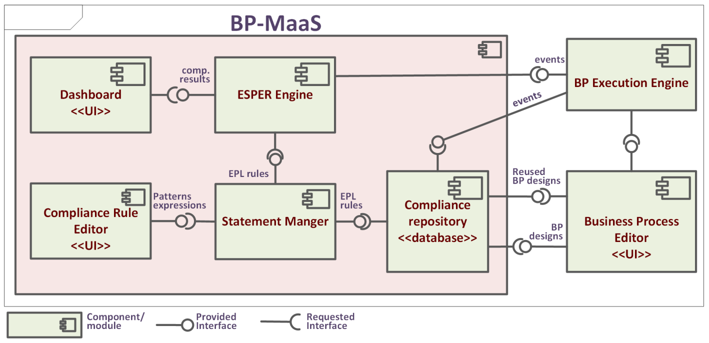


Fig. 1. BP-MaaS architecture represented as a UML component diagram

Fig. 2 illustrates the set of compliance patterns which are supported by the *BP-MaaS* framework. For a detailed description of the compliance patterns, we refer the reader to [1]. The visual editor component has been implemented as a plugin on top of the Oryx editor. The lower-most layer of Fig. 3 represents a screenshot of the visual rule editor representing the typical segregation of duties compliance constraint [1] which mandates that two activities cannot be performed by the same roles or actors in order to minimize the possibility of fraud.

Statement Manager. This module is responsible for automatically compiling the visually modelled compliance rule into a set of statements/queries based on the defined mapping scheme. For BP-MaaS, we are considering Event Processing Language (*EPL*) queries of the ESPER framework [5]. In this context, streams replace tables as the source of data with events replacing rows as the basic unit of data. Listing 1 shows an example of automatically generated EPL statement for the *absence* anti-pattern. The absence pattern requires that a specific activity *not* to be executed within a specific scope of the process execution [1]. Generated EPL queries are sent to the compliance monitoring component.

```
INSERT into RuleViolationEvent (processID, Message, RuleID, RuleType)
SELECT s.ProcessID, 'Event_{Antecedent}({TaskName})_occurred_less_than
{MinOccurs}_within_{ScopeStart}({st})_and_{ScopeEnd}({se})_in_the
process_instance', '{RuleID}','{RulePattern}'
FROM PATTERN [
 every( s = {ScopeStart}(cast(s.Task, string)='{st}')
 ->(e = {ScopeEnd}(cast(e.Task, string)='{se}',ProcessID=s.ProcessID)
 ))] as scope
WHERE {MinOccurs}>(select count(*)from{Antecedent}.win:keepall() as T
WHERE cast(T.Task, string)='{TaskName}' and
 (T.TimeStamp between scope.s.TimeStamp and scope.e.TimeStamp ) )
```

Listing 1 . EPL statement to detect below-min-occurrences *absence* anti-pattern

---

`https://code.google.com/p/oryx-editor/`

`http://esper.codehaus.org/esper-4.2.0/doc/reference/en/html/epl_clauses.html`

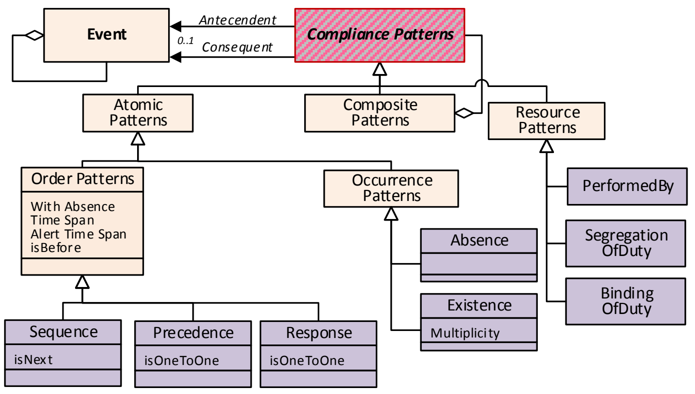


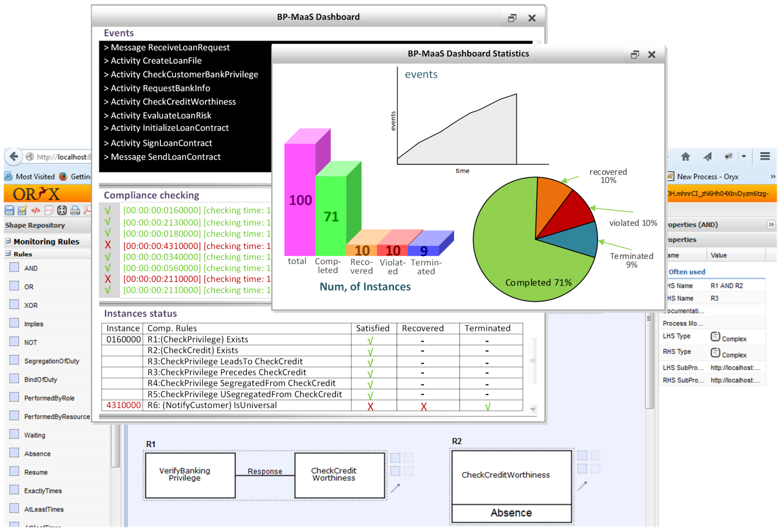
Fig. 2. The set of compliance patterns supported by the BP-MaaS framework [1]

Business Process Editor and Execution Engine. Provides the end users with a user-friendly modelling environment where the users can model their business process using the standard BPMN 2.0 language. We employ the open source BPM platform Activiti as a realization of this component where the user can model and enact business processes. We also did an extension of the Activiti engine to allow emitting process execution events to our compliance monitoring engine.

Compliance Monitoring Engine. The open source complex event processing platform ESPER is responsible for *continuously* evaluating the generated statements from the 'Statement Manager' over the stream of events, which is received from the BP execution engine. The engine triggers the execution of the *compliance actions* for any detected violations of the compliance rules. Compliance recovery actions are defined as meta-data for each defined rule. Our choice of Esper is mainly because it provides an environment for developing applications that can process large volumes of incoming messages or events, regardless of whether the incoming messages are historical or real-time in nature. It also supports filtering and analyzing of events in various ways, and responds to conditions of interest. In addition, ESPER shows acceptable performance as it is able to handle about 120,000 events per second [4,5] making it scalable to handle process execution environments with numerous process instances.

Monitoring Dashboard. The dashboard is a user-friendly interface that enables the end-user to monitor the stream of events and manipulate (e.g., adding, removing, activating, deactivating) the set of registered compliance rules in addition to being able to receive the notifications about the detected non-compliance instances. Fig. 3 shows screenshots of the developed dashboard, which has been implemented using Microsoft C# .Net technology.

`http://activiti.org/`



Events

```
> Message ReceiveLoanRequest
> Activity CreateLoanFile
> Activity CheckCustomerBankPrivilege
> Activity RequestBankInfo
> Activity CheckCreditWorthiness
> Activity EvaluateLoanRisk
> Activity InitializeLoanContract
> Activity SignLoanContract
> Message SendLoanContract
```

Compliance checking

```
√ [00:00:00:0160000] [checking time: 1
√ [00:00:00:2130000] [checking time: 1
√ [00:00:00:0180000] [checking time: 1
X [00:00:00:4310000] [checking time: 1
√ [00:00:00:0340000] [checking time: 1
√ [00:00:00:0560000] [checking time: 1
X [00:00:00:2110000] [checking time: 1
√ [00:00:00:2110000] [checking time: 1
```

Instances status

Instance	Comp. Rules	Satisfied	Recovered	Terminated
0160000	R1:(CheckPrivilege) Exists	√	-	-
	R2:(CheckCredit) Exists	√	-	-
	R3:CheckPrivilege LeadsTo CheckCredit	√	-	-
	R3:CheckPrivilege Precedes CheckCredit	√	-	-
	R4:CheckPrivilege SegregatedFrom CheckCredit	√	-	-
	R5:CheckPrivilege USegregatedFrom CheckCredit	√	-	-
4310000	R6: (NotifyCustomer) IsUniversal	X	X	√

Fig. 3. BP-MaaS System Screenshots.

3 Demonstration Scenario

In our demo, we are presenting the implementation of the *BP-Maas* System. In particular, we are showing the scenario where we model a compliance rule using the graphical rule editor (Fig. 3). Then, the modeled rule is registered to the compliance monitoring component. This is followed by showing how the dashboard is updated with information about the newly registered rule. The loan approval BP from the EU COMPAS project is used for monitoring its execution steps. When the execution events start to arrive at the monitoring component and a violation scenario is detected, we show how the dashboard is updated with information about the instance(s) violating a specific rule.

Acknowledgment

This work was supported by King Abdulaziz City for Science and Technology (KACST) project 11-INF1991-03.

References

1. A. Awad, A. Barnawi, A. Algammal, A. Almalaise, R. Elshawi, and S. Sakr. Runtime Detection of Business Process Compliance Violations: An Approach based on Anti Patterns. In *SAC*, 2015.
2. A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou. Formalizing and applying compliance patterns for business process compliance. *Software and Systems Modeling*, 2014.
3. O. Etzion and P. Niblett. *Event processing in action*. Manning, 2010.
4. A. Mathew. Benchmarking of complex event processing engine- esper, 2014.
5. V. Mijovic and S. Vranes. A survey and Evaluation of CEP Tools. In *YUINFO*, 2011.

Video demonstration of BP-MaaS is available on `https://www.youtube.com/watch?v=wRdZKsOi5x4`