Monitoring of Business Process Execution Based on Performance Indicators

Antonello Calabro´
CNR-ISTI
Pisa, ITALY
Email: antonello.calabro@isti.cnr.it

Francesca Lonetti
CNR-ISTI
Pisa, ITALY
Email: francesca.lonetti@isti.cnr.it

Eda Marchetti
CNR-ISTI
Pisa, ITALY
Email: eda.marchetti@isti.cnr.it

***Abstract*—Nowadays, more and more industrial organizations are using Business Process Model and Notation (BPMN) for process modeling. Key performance Indicators (KPIs) are set on such process models so to get a quantitative assessment of critical success metrics. A timely and reliable monitoring of KPIs is instrumental to Business Process (BP) management, and several frameworks are being proposed for such purpose. Business process monitoring solutions can be embedded into the Business Process Modeling (BPM) execution framework or integrated as additional facilities. This paper presents an integrated framework that allows for modeling, execution and analysis of business process based on a flexible and adaptable monitoring infrastructure. The main advantage of the proposed approach is that it is independent from any specific business process modeling notation and execution engine and allows for the definition and evaluation of user-specific KPI measures.**

I. Introduction

In recent years, inside many industrial contexts and application domains the Business Process Model and Notation (BPMN) has increased its importance due mainly to the possibility to easily allow formal model specification, provide accepted and concise definitions and taxonomies, and develop an executable framework for overall managing of the process itself. The main benefits of the use of BPMN commonly rely on the possibility of creating a description of processes (in terms of participants and activities). This can be also used for processes' analysis and simulation and for executing the process models themselves by automating the process steps, also using cloud services. For this the Business Process Modeling (BPM) has become an effective mean for creating abstract representations of knowledge, providing formalized definitions of the different activities, evaluating the process executions and/or evolutions [1].

A key aspect of the BPM lifecycle is the continuous monitoring of business goals. Indeed new corporate strategies rely more and more on specific key performance indicators (KPIs), i.e. properly defined values useful for timely measure the business process performance. By using specific metrics the data collected during the business process execution are analyzed and elaborated so to calculate the KPIs values of interest and continuously tracking the behavior of the process. Indeed KPIs are considered a cost-effective means for measuring strategic objectives, process-specific goals and controlling the effective process execution [1]. Depending on the application context several kinds of KPIs can be defined, which differ in their nature, including financial, quantitative, qualitative, or time-based aspects, and different measures can be adopted for their calculation. Usually, data collection useful for KPIs evaluation relies on monitoring facilities, which can vary in their implementation depending also on the modeling notations adopted for the process itself. The basic idea is to extract relevant information from the events produced during the BPM execution and then process and analyze it so to perform the KPIs useful for controlling the process workflow.

As further detailed in this paper, several monitoring proposals are currently available, which can be mainly divided into two groups: those that are embedded in the BPM execution engine such as [2], [3] and those that can be integrated into the execution BPM framework as an additional component such for instance [4], [5]. Both the solutions have specific advantages. For sure, an embedded solution reduces the performance delay of the execution framework, mainly in terms of interactions and communication time. KPI indicators can be directly evaluated by the execution framework, which can also execute corrective actions in case of important deviations. The main disadvantage of these approaches is the lack of flexibility both in the business data collection, KPI measures definition and planning of corrective actions. Usually, in these proposals all the interested parameters, KPIs or activities have to be predefined and modeled directly into the business process model, by means of specific editors, and are dependent on the notation used for business process definition. Thus any change requires to redesign or improve the business model itself, preventing in such manner the possibility of dynamic modification.

Considering instead additional monitor facilities, they represent flexible, adaptable and dynamic infrastructures that are independent from any specific business process modeling notation and execution engine. The corrective actions triggered by the KPI violations can be designed without modifying the structure of the BPM and dynamically updated or modified when necessary. The type of data to be collected during the BPM execution is independently specified by the BPM and is not linked to the specific notation used for business process definition. Consequently KPIs and metrics adopted can be dynamically defined as monitoring properties and can be applied to different execution environments.

Among the two presented solutions in this paper we focus on the use of an additional monitor facility integrated into a BPM execution framework. In particular, we refer to the monitoring framework called Glimpse [6], [5], which is extremely flexible and adaptable to various scenarios and SOA architecture patterns. By means of Glimpse the useful monitoring data can be collected so that a set of defined KPIs can be evaluated.

The contribution of this paper can be summarized into:

1) the integration for the first time of Glimpse with a BPM execution engine.
2) the definition of the architecture of an integrated environment supporting BP KPI monitoring that is independent from specific notations and platforms, and allows for flexible KPI decisions.
3) a comparison of the proposed framework with closest related works and their critical analysis.

Moreover, a first validation of this framework considering a proof-of-concept learning system is presented in [7].

The remainder of this paper is structured as follows: Section II presents the proposed business process execution and monitoring framework. Section III addresses related works on monitoring of business process and their comparison with the proposed monitoring solution. Finally, Section IV concludes the paper also hinting at future work.

II. Business Process Execution and Monitoring

In this paper we propose a possible architecture of a framework able to execute the business process, monitor the proper data and evaluate the KPIs of interest.

For this we identify five main components:

- a *KPI manager* for the definition of the KPIs of interest and the final derivation of the KPIs values according to the data collected during the monitoring activity of the business process execution.
- a *BPM editor* for annotating the business process model with non-functional constraints to be monitored. For this a coherent set of domain-specific languages, expressed by meta-models can be used for annotating the business process. These annotations can be automatically translated into inputs for monitoring rules exploiting the support for automation offered by model-driven engineering techniques [5].
- an *executor engine* for executing step by step the business process instance. Different solutions for the business process execution engine are: Activiti, Camunda and jBPM. In this paper we rely on Activiti [8].
- a *monitoring infrastructure* for collecting data of interest during the run-time business process execution. There can be different solutions for collecting the important data. In this paper we rely on Glimpse infrastructure [6], [5] which has the peculiarity of decoupling the events specification from their collection and processing. Specifically, as further detailed in the next subsection, the monitoring component captures business process events by means of a set of probes, i.e. code in charge to collect and/or send raw data deriving from the execution of a business process.
- a *communication layer* which manages the communications among the framework components for the execution and monitoring of a BPM. This channel constitutes the communication backbone conveying all information (events, requests, notifications) flowing among all the components and implements the data transmission for instance by means of a publish-subscribe mechanism. Also in this case different solutions can be adopted. In this paper we rely on an Enterprise Service Bus (ESB) realized using ServiceMix [9] on which is running ActiveMQ [10] as message broker.

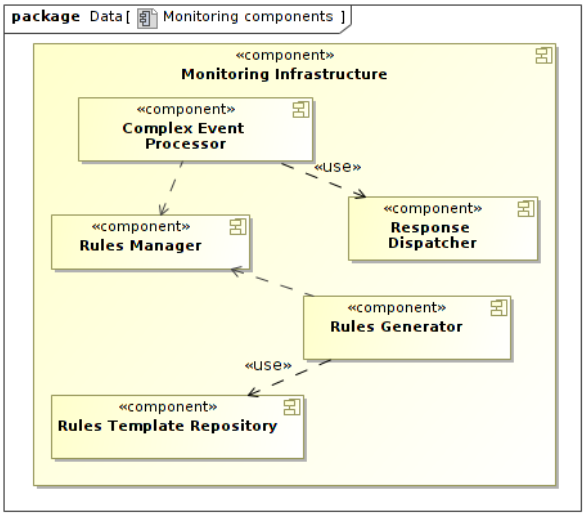

Fig. 1. Multi-source Monitoring Package Diagram

A. Glimpse

In this section we provide further details about the Glimpse architecture. As showed in Figure 1, the main components of this architecture are:

a) Complex Events Processor (CEP): i.e. a rule engine which analyzes the events generated by the probes and correlates them to infer more complex events. If the event triggers no rule, the event is just collected into the Event Stream of the CEP. Several rule engines can be used such as Drools Fusion, VisiRule, RuleML. In the current framework implementation, the Complex Event Processor is realized using Drools Fusion [11].

b) Response Dispatcher: i.e. a registry that keeps track of the requests for monitoring sent to the monitoring infrastructure. Once it receives the advice of a rule firing, pattern completion, or property violation from the CEP, the Response Dispatcher sends the evaluation to the requester.

c) Rules Generator: i.e. the component in charge to generate the rules using the templates stored into the *Rules Template Repository*. These rules are generated according to the specific properties and KPIs to be monitored. A generic

Listing 1. A meta-rule example

```
rule "BPMN_SESSIONID_COMPLETIONTIME"
  no-loop
   salience 10
   dialect "java"
     when
       $aEvent : GlimpseBPMNEvent(
          this.isConsumed == false,
          this.getEventName == "start",
          this.getSessionID == "_SESSIONID_",
          this.getEventActivityName == "_TASKID_",
          this.getTimeStamp == _TIMESTAMP_);

       $bEvent : GlimpseBPMNEvent(
          this.isConsumed == false,
          this.getEventName == "end",
          this.getSessionID == "_SESSIONID_",
          this.getEventActivityName == "_TASKID_",
          this after $aEvent);

     then
         $aEvent.setConsumed(true);
         update($aEvent);
         $bEvent.setConsumed(true);
         update($bEvent);

         ResponseDispatcher.LogExecution(
           "RULE",
           " auto_generated_rule ",
           "The completion time is :" + bEvent.getTimeStamp()
                                      - aEvent.getTimeStamp());

          retract ($aEvent);
          retract ($bEvent);
end
```

rule consists of two main parts: in the first part the events to be matched and the constraints to be verified are specified; the second part includes the events/actions to be notified after the rule evaluation. In Listing 1 an example of rule-template which calculates the completion time between two events (a and b) is provided.

d) Rules Template Repository: i.e. an archive of predetermined rules templates that will be instantiated by the *Rules Generator* when needed. A rule template is a rule skeleton, that is specified by the instantiation of a set of template-dependent placeholders. For the sake of completeness, the *Rules Template Repository* can also include sets of static rules that do not depend on the generative process discussed above.

e) Rules Manager: i.e. a component in charge to instruct the *Complex Event Processor*, to load and unload a set of rules. The *Rule Generator* will instantiate the template with appropriate values inferred from the specific properties to be monitored. Once a rule is istantiated this is loaded by the *Rule Manager* into the *Complex Event Processor*. The complex event detection process depends directly on the operations performed by the *Rules Manager* component. We refer to [6], [5] for a more detailed description of the monitoring architecture.

III. Related Work

Runtime monitoring represents the main element for the management of communication networks and distributed systems [12]. Nowadays, in many application domains, modeling of business process and business process goals evaluation is increasingly gaining importance and many proposals deal with different aspects of run-time monitoring of distributed business processes [13], [2], [4]. The authors of [2] provide an integrated framework for run-time monitoring and analysis of the performance of WS-BPEL processes. This work addresses a machine learning based analysis of QoS metrics that enables continuous observation of key performance indicators and allows to discover the main factors that influence the process performance. Other approaches deal with monitoring of WS-BPEL processes [13], [14]. Baresi et al. [13] propose external monitoring rules for dynamically controlling the execution of WS-BPEL processes. Traverso et al. [14] describe an architecture for monitoring web services described as BPEL processes. This approach allows to separate the service business logic from the monitoring functionality and provides the ability to check time related and statistic properties. A different monitoring approach is provided in [4] where a model-driven approach for generating a monitoring infrastructure based on events is presented. The goal of this work is to show how KPIs are modeled and transferred into event rules by a model-driven approach.

The works closest to our proposal are those described in [3], [15], [16] that combine modeling and monitoring facilities of business process. PROMO [3] allows to model, monitor and analyze business process. It provides an editor for the definition of interesting KPIs to be monitored as well as facilities for specifying aggregation and monitoring rules. Our proposal is different since it addresses a flexible, adaptable and dynamic monitoring infrastructure that is independent from any specific business process modeling notation and execution engine. The authors of [15] focus on monitoring business constraints at runtime by means of temporal logic and colored automata. The presented ProM tool allows for continuous compliance with respect to predefined business process constraint model and recovery after the first violation. Differently from this approach, our solution does not allow to take counter measures for recovering from violation of defined performance constraints. Moreover, in our solution these constraints are not specified in the business process but they are dynamically defined as monitoring proprieties that can be applied to different business process notations. Finally, the IBM Business Monitor [16] is designed to accept and process events from any source. Applications can be instrumented to emit events, then the tool extracts and analyzes the business data from the incoming events for later reporting purposes in the dashboards.

A comparison of our proposal with the above-mentioned closest related works and their critical analysis are presented in Table I. For each reference, we provide a short description in the second row of the table and we classify it according to six dimensions: if a facility for KPI definition is provided (row labeled *KPI Definition*); if the monitoring framework is independent from the business process modeling notation (row labeled *Language Independence*); if the proposed business process execution engine has an additional overhead due to the runtime monitoring (row labeled *Interaction Overhead*);

TABLE I
COMPARISON OF OUR PROPOSAL WITH MAIN RELATED WORK.

	Our Proposal	Promo[3]	IBM Business Monitor[16]	ProM[15]
Description	generic infrastructure for BP execution and monitoring	collaborative tool for BP modeling monitoring and analysis	monitoring of BPMN process application	monitoring based on temporal logic and colored automata
KPI Definition	user defined	user defined	user defined dashboard	expressed as LTL properties
Language Independence	✓	-	-	-
Interaction Overhead	cardinality of exchanged messages	embedded	embedded	embedded
Events Correlation	✓	✓	✓	✓
Offline Analysis	✓	✓	✓	✓
Recovery from Violations	-	-	-	✓

if the reference provides events correlation and inference of complex events (row labeled *Events Correlation*); if the reference allows to make offline analysis of the collected data (row labeled *Offline Analysis*) and finally if the monitoring solution provides facilities for recovering from constraints violation.

Table I evidences that all approaches allow the user to define the constraints to be monitored using specific notations and editing facilities. Our monitoring solution is independent from any business process notation. However, differently from the other monitoring approaches that are embedded in the business process execution engine, our monitoring component implies an additional overhead due to the interaction (messages exchange) with the business process execution engine. All the approaches provide events correlation and offline analysis whereas only the monitoring framework presented in [15] provides facilities for recovering from constraints violations.

IV. CONCLUSION

This paper presented a business process execution and monitoring framework based on Glimpse that is a flexible and configurable monitoring infrastructure. The peculiarities of the proposed framework mainly rely on its independence from any specific business process modeling notation and execution engine and on the definition and evaluation of user-specific KPI measures. An example of implementation of the proposed execution and monitoring framework as well as a preliminary validation on a real case study in the learning context have been presented in [7].

As future work, we would like to validate the proposed framework considering more complex business process models. Moreover, thanks to the possibility of the monitoring infrastructure to discover problems or anomalies during the BPM execution, we would like to integrate in the proposed framework also automated facilities able to rapidly advise and execute counter measures so to improve the support offered to the various stakeholders.

ACKNOWLEDGMENT

This work has been partially funded by the Model-Based Social Learning for Public Administrations project (EU FP7-ICT-2013-11/619583).

REFERENCES

[1] J. v. Brocke and M. Rosemann, "Handbook on business process management 1: Introduction, methods, and information systems," 2014.
[2] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, and F. Leymann, "Monitoring and analyzing influential factors of business process performance," in *Enterprise Distributed Object Computing Conference*, Sept 2009, pp. 141–150.
[3] P. Bertoli, M. Dragoni, C. Ghidini, E. Martufi, M. Nori, M. Pistore, and C. Di Francescomarino, "Modeling and monitoring business process execution," in *Service-Oriented Computing*, 2013, pp. 683–687.
[4] F. Koetter and M. Kochanowski, "A model-driven approach for event-based business process monitoring," in *Business Process Management Workshops*, 2013, pp. 378–389.
[5] A. Bertolino, A. Calabrò, F. Lonetti, A. Di Marco, and A. Sabetta, "Towards a model-driven infrastructure for runtime monitoring," in *SERENE*, 2011, pp. 130–144.
[6] A. Bertolino, A. Calabrò, F. Lonetti, and A. Sabetta, "Glimpse: A generic and flexible monitoring infrastructure," in *Proceedings of the 13th European Workshop on Dependable Computing*, 2011, pp. 73–78.
[7] A. Calabró, F. Lonetti, and E. Marchetti, "KPI Evaluation of the Business Process Execution through Event Monitoring Activity," in *Third International Conference on Enterprise Systems 2015*, submitted for publication.
[8] "Activiti BPM Platform," http://activiti.org/.
[9] Apache, "Apache ServiceMix," http://servicemix.apache.org/.
[10] ——, "Apache ActiveMQ," http://activemq.apache.org/.
[11] "Drools Fusion: Complex Event Processor," http://www.jboss.org/drools/drools-fusion.html.
[12] M. Mansouri-Samani and M. Sloman, "Network and distributed systems management," 1994, ch. Monitoring Distributed Systems, pp. 303–347.
[13] L. Baresi and S. Guinea, "Towards dynamic monitoring of ws-bpel processes," in *Third International Conference of Service-Oriented Computing*, 2005, pp. 269–282.
[14] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-time monitoring of instances and classes of web service compositions," in *International Conference on Web Services*, Sept 2006, pp. 63–71.
[15] F. M. Maggi, M. Montali, M. Westergaard, and W. M. Van Der Aalst, "Monitoring business constraints with linear temporal logic: An approach based on colored automata," in *Business Process Management*. Springer, 2011, pp. 132–147.
[16] "IBM Business Monitor V7.5," http://www.ibm.com/developerworks/websphere/bpmjournal/1106_alcorn2/1106_alcorn2.html.