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. Data collected during business process execution are used for deriving the key performance indicators (KPI) that allow continuous tracking of the process behavior and measurement of process-specific goals. KPIs evaluation leverages on business process monitoring solutions that can be embedded into the BPM execution framework or integrated as additional facilities. This paper presents an integrated framework that allows 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. An example of implementation of the proposed execution and monitoring framework as well as its validation on a real case study in the learning domain are also presented.**

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, wherever it is possible 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]. The generic and adaptable nature of this methodology makes easy its application into different environments such for instance the clinical one, for managing the patient treatment, the financial sector for ruling the bank processes, or the leaning context, for guiding the training activity. In all these application contexts, a key role is played by the data collected during the business process execution, which lets the possibility of reasoning about and/or improving the overall performance of the business process itself.

Indeed new corporate strategies rely more and more on specific key performance indicators (KPIs), i.e. properly defined values useful for evaluating the 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 [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 integrated environment enabling the modeling, execution, monitoring and analysis of business process.
3) an instantiation of the proposed architecture on a real case study relative to the learning context.

The remainder of this paper is structured as follows: Section II presents the proposed business process execution and monitoring framework whereas Section III presents a first validation of this framework considering a proof-of-concept learning system. Section IV addresses related works on monitoring of business process and their application in learning context. Finally, Section V 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 [6], Camunda [7] and jBPM [8]. In this paper we rely on Activiti [6].
- 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 [5] infrastructure 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 components framework 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.

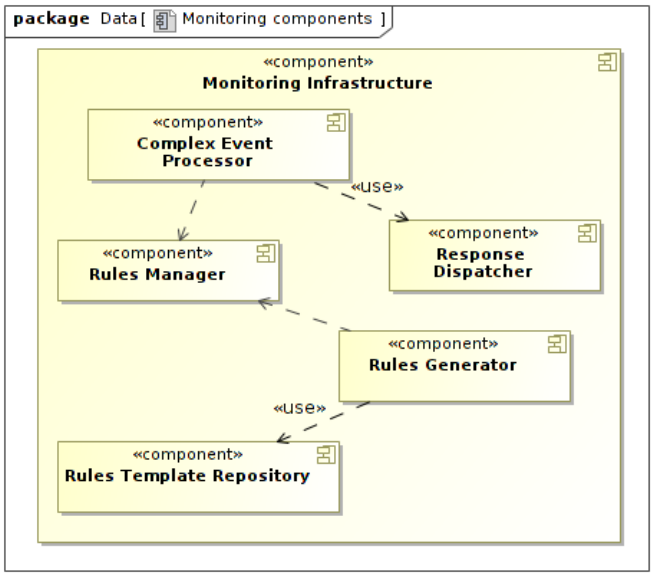

Fig. 1. Multi-source Monitoring Package Diagram

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 probes and correlates 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[11], VisiRule, RuleML. In the current framework implementation, the Complex Event Processor is realized using Drools Fusion.

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. a 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 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. An example of monitoring rule will be presented in Listing 1.

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 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 [5] for a more detailed description of the monitoring architecture.

III. Case Study

In this section we present an example of instantiation and use of the monitoring framework depicted in Section II.

In particular, in the following subsections we briefly introduce the case study considered (Sec. III-A) and the KPIs considered (Sec. III-B), then we present the implementation of the monitoring infrastructure (Sec. III-C) and finally we present and discuss the results collected (Sec. III-D).

A. Business Process Description

The case study considered in this paper is a prototyped version of a learning training systems, that will be used by students during the training for the practical examination of a software engineering university course. Figure 2 presents the reduced version of the overall Business Process Model (BPM) considered in this paper. Specifically, the BPM models the interaction of two different stakeholders: the `student` who makes the request of a learning session simulation and the `Registration system`, which is in charge of managing the students' interactions and evaluating the learning session performed by the students. In detail, the student requests to the registration system to start a learning session,(`Send examination request`) activity, thus the system automatically generates a questionnaire (`Generate questionnaire`) that the student should fill. Once the student completes the questionnaire the collected responses are sent to the registration system (`Send Responses`) and analyzed by a teacher (`Assess examination assisted by Monitoring Dashboard`). He/she is in charge to examine the elaboration of the students and provide them feedback and suggestions (`Send Results/KPI`). On the basis of collected monitoring data the teacher can also derive statistics and evaluations of the questioners themselves so to modify or improve them when necessary.

In Figure 3 the subprocess relative to the `Examination` activity is further detailed.

In this model during a learning simulation a student has to solve problems relative to two different topics. For each topic the student can choose from three different prefixed replies: one of them is *wrong*, another is *partially correct*, and the remaining one is *correct*. In particular, by means of the activity labeled `Training Step One` the student starts solving the first question relative to the first topic. In this case the BPM models four different situations:

- In case of a *wrong* reply a different question on the same topic is proposed to the student by means of the activity (a script task) labeled `Question Regeneration`;
- If the reply is classified *partially correct* a simpler question is proposed for the second topic by means of the activity labeled `Training Step Two:Low`;
- If the reply is classified *correct* a standard question is proposed for the second topic by means of the activity labeled `Training Step Two:Normal`;
- The student can in any moment decide to abandon the learning session by means of the sequence flow labeled `Abandon session`.

In the second and third case, the student starts solving the question relative to the second topic. Also in this case the above four situations are modeled in the BPM. Learning session is concluded by the activity labeled `Collect responses` that collects the student's responses in a structured format.

B. KPIs Definition

In the considered case study, the monitoring infrastructure is responsible to provide inputs for the learning process KPIs evaluation. In particular three different macro categories of KPIs are considered:

i) the *learning adequacy*: the degree of correctness of the replies provided by the students to the questionnaire; ii) the *completion time*: the time necessary to the students for completing each single task or the overall questionnaire; iii) the *learner competency*: the cardinality of reloads of the question associated to the same topic.

The measures associated to the evaluation of each KPI are listed below and are differentiate at user level (i.e. student or teacher level):

Student level

- *Correctness score* is assigned depending on the correctness of the replies provided by the students. The score is calculated considering the number of replies classified as *correct/partially correct*. In Table I in the last column an example of KPI measurement is presented considering the extract of BPM of Figure 3.
- *Student Learning Adequacy* measured in terms of degree of correctness of the replies and number of

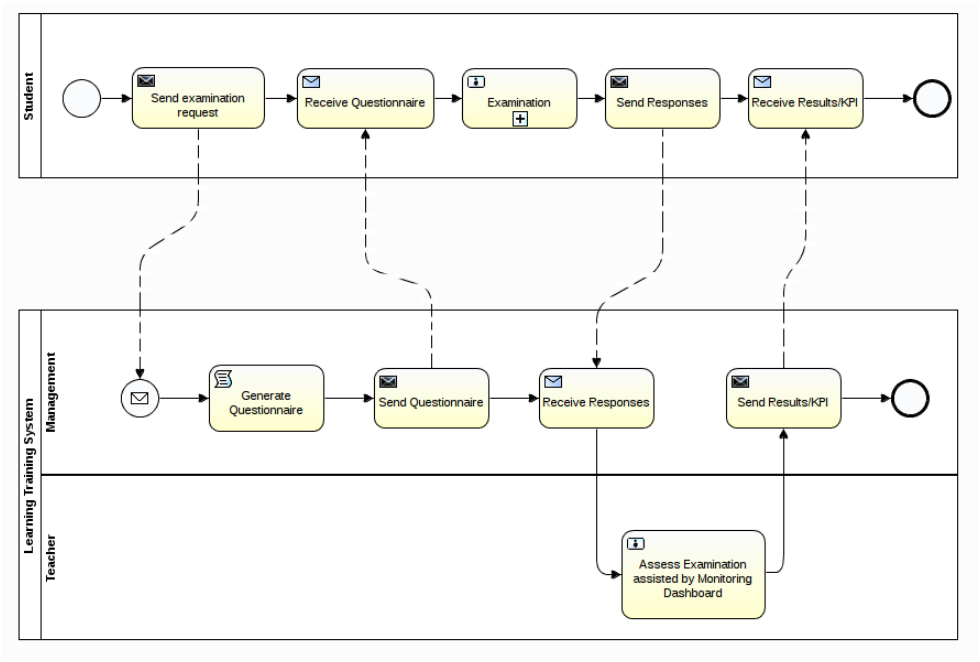

Fig. 2. The BPMN Main Process

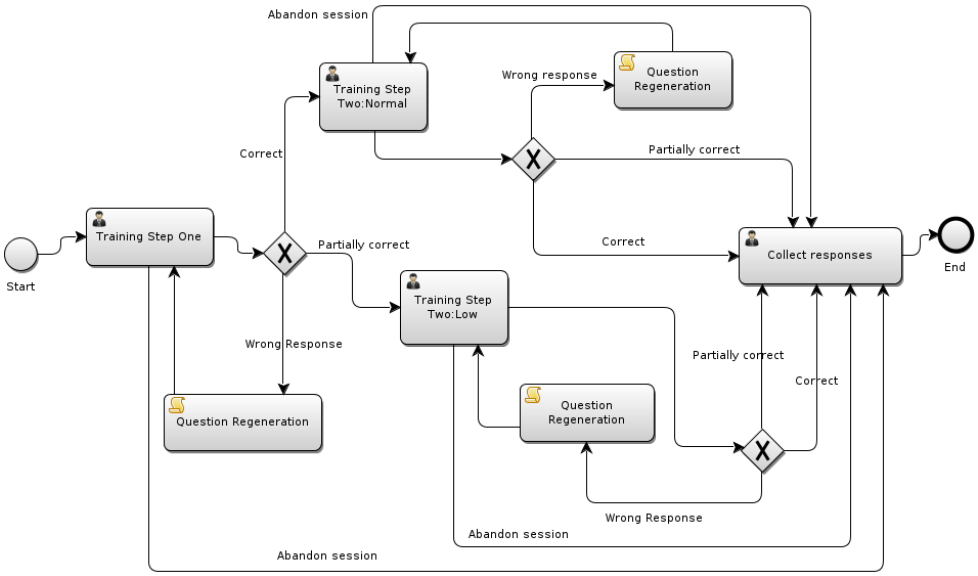

Fig. 3. The Examination subProcess

reloads of the questions. In particular, the metric used is the following:

$$C.score - \sum_{k=1}^{j} \left\lfloor \frac{N}{3} \right\rfloor$$

where *C.score* is the *Correctness score*, N is the number of reloads per training steps and j is the number of steps in each questionnaire. Example of computation of this KPI are provided in Table II last column.

Teacher level

- *Correctness Score per Training Step (CSTS)* is computed as the number of replies considered *correct* over the total amount of students who have executed the considered *Training Step*. For this KPI, values close to 1 are better. An example of computation of this KPI is provided in Table IV.
- *Correctness Score per Learning Session (CSLS)* is computed as the mean value of the *CSTS* calculated for a Learning Session. For this KPI, values close to 1 are better. An example of computation of this KPI is provided in Table V.
- *Reload Cardinality per Training Step (RCTS)* is com-

TABLE I. KPI CORRECTNESS SCORE EVALUATION

	Tr.Step 1	Tr.Step 2:Normal	Tr.Step 2:Low	Correct.score
Correct	A	A		A
Correct	A	B		B
Part. Correct	B		A	C
Part. Correct	B		B	D

puted as the number of reload per Training Step. For this KPI, values close to 0 are better. An example of computation of this KPI is provided in Table IV.

- *Reload Cardinality per Learning Session (RCLS)* is computed as the mean value of the *RCTS* calculated for a Learning Session. For this KPI, values close to 0 are better. An example of computation of this KPI is provided in Table V.
- *Session Learning Adequacy per Training Step (SLATS)* is computed as the sum of *CSTS* + *RCTS* for each Training Step in a Learning Session. For this KPI, values close to 1 are better. An example of computation of this KPI is provided in Table IV.
- *Session Learning Adequacy per Learning Session (SLALS)* is computed as the sum of *SLATS*s encountered in each path of a questionnaire divided by the number of the Traning Steps composing a path.
  In particular, the metric used is the following:

$$\frac{\sum_{j=1}^{n} SLATS(j) * h}{T}$$

  where *h* is equal to 1 if the Training Step belongs to the path *j* in the questionnaire and 0 otherwise, and *T* is the total amount of Training Step composing a path. For this KPI, values close to T are better. An example is provided in Table V.
- *Completion Time per Training Step (CTTS)* is computed as the mean completion time for each Training Step in a Learning Session. Boundary values for each Training Step are provided. An example of computation of this KPI is provided in Table IV.
- *Completion Time per Learning Session (CTLS)* is computed as the mean value of *CTTS* for each path of a questionnaire. Boundary values for Session are provided.
  In particular, the metric used is the following:

$$\frac{\sum_{j=1}^{n} CTTS(j) * h}{T}$$

  where *h* is equal to 1 if the Training Step belongs to the path *j* in the questionnaire and 0 otherwise and *T* is the total amount of Training Step composing a path. An example is provided in Table V.

C. Implementation

As described in Section II, the execution and monitoring framework involves different components. In this section we detail the implementation used for executing, monitoring and managing the BPM presented in Section III-A. In particular, as depicted in Figure 4, for the framework implementation three different machines have been considered.

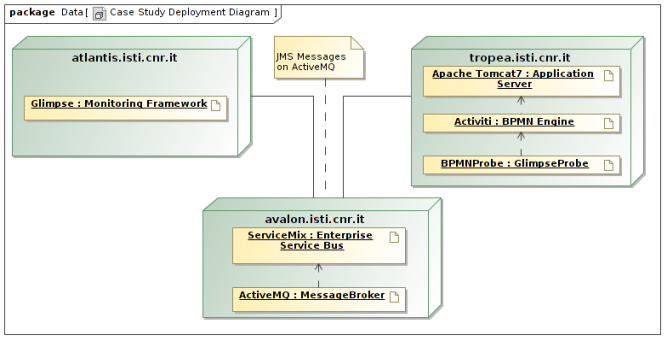

Fig. 4. Deployment diagram

In particular, the Glimpse monitoring infrastructure, already detailed in Section: II, is deployed on *atlantis.isti.cnr.it* and connected to *avalon.isti.cnr.it* where the Message Broker (ActiveMQ) is running on top of the ESB (ServiceMix). Finally, the BPMN Engine, that has been identified in Activiti, is running on *tropea.isti.cnr.it* and it is accessible to the students by means of a web service interface running on the same machine. For aim of simplicity the KPI editing and management has been performed using spreadsheet while a BPM has been annotated using the integrated editor provided by Activiti Framework. During the learning session, Activiti interacts with the Monitoring Framework by means of notification messages triggered by the student BPM execution. In particular, the notification is forwarded by means of probe that is running on the Activiti engine. A probe, is a java code integrated into Activiti using the facilities provided by the Activiti ExecutionListeners [12]. Specifically, each activity expressed into the BPM, involves at least two or more ExecutionListener, which execute actions when certain events occur at runtime. The set of the events traced are: taking a transition, start and ending of a process, activity or gateway. All the parameters annotated on the KPI using the integrated BPM Editor provided by Activiti will be encapsulated by the probe in a GlimpseBPMNEvent. Figure 5 provides an example of the GlimpseBPMNEvent class adopted in this case study. The probe will send the event to the Complex Event Processor (CEP) on a JMS [13] channel through the ESB. The CEP will infer which of the rule templates already available into the Rule Template Manager should be instantiated with the data contained into the GlimpseBPMNEvent.

Once the instantiated rule-template is ready, the Rule Manager loads it into the CEP. As soon as the rule constraints are satisfied by the events monitored during the BPM execution the rule is considered fired and the proper data will be stored for the KPI evaluation.

In Listing 1 an example of rule-template which calculates an activity completion time is provided.

D. Results collection and analysis

In this section the results collected during a session learning are presented. In particular, we ask 20 students of a Software Engineering Course to use the prototyped version of the

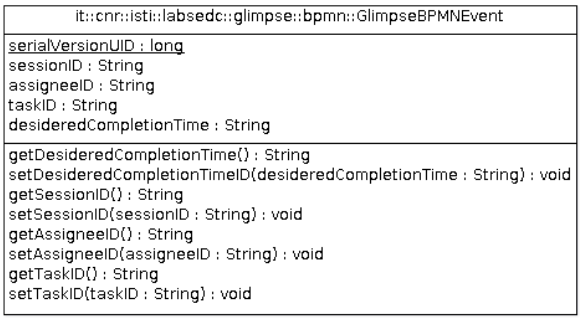

Fig. 5. GlimpseBPMNEvent class

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 ",
            "\nThe completion time is:" + bEvent.getTimeStamp() - aEvent.getTimeStamp());

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

Learning Training System presented in Section III. In this simplified version the BPM models only two Training Steps for each questionnaire (Figure 3). For evaluation purposes we considered a learning session of about one hour in which all the students in parallel reply to the various questions. Once the learning session finished the KPIs have been computed using the monitoring data collected and shown to the students and teacher. In this experiment the boundary values associated to the various CTTSs have been fixed to *180*, *300*, *480* seconds for the *Training Steps One, Two:Low, Two:Normal* respectively and 570 second for the CTLS.

Table II reports the monitoring data and the KPI for each student, calculated according to the formulas of Section III-B. In particular, the column labeled *TS1*, (*TS2N*. *TS2L* respectively) reports the correctness score (CSTS) of the *Training Step One* (*Training Step Two:Normal* and *Training Step Two:Low* respectively). The column labeled *ReloadTS1* (*ReloadTS2N*, *ReloadTS2L* respectively) reports the reload cardinality (RCTS) of the *Training Step One* (*Training Step Two:Normal*, *Training Step Two:Low* respectively). The columns labeled *Corr.score* and *Learning Adequacy* report the reload cardinality (RCLS) and the correctness score (CSLS) respectively of the overall training session. As evidenced in the table there are cases in which the values of CSLS obtained by the student are lower than that of RCLS due to the high number of reload performed for the same question (see for instance Student 1). This indicates that the student has some difficulties in the specific topic and needs to improve his/her preparation.

TABLE II. CORRECTNESS SCORE AND STUDENT LEARNING ADEQUACY

	TS1	Reload TS1	TS2N	Reload TS2N	TS2L	Reload TS2L	Corr. Score	Learning Adequacy
Stud.1	B	3			A	7	C	F
Stud.2	A	0	B	2			B	B
Stud.3	A	0	A	0			A	A
Stud.4	A	3	A	0			A	B
Stud.5	B	2			A	0	C	C
Stud.6	A	4	A	0			A	B
Stud.7	A	0	B	0			B	B
Stud.8	A	0	A	0			A	A
Stud.9	A	0	B	2			B	B
Stud.10	B	0			A	2	C	C
Stud.11	B	0			A	1	C	C
Stud.12	A	0	B	2			B	B
Stud.13	A	0	A	0			A	A
Stud.14	A	1	A	3			A	B
Stud.15	A	4	B	1			B	C
Stud.16	A	0	B	2			B	B
Stud.17	A	0	A	3			A	B
Stud.18	A	0	A	1			A	A
Stud.19	A	1	B	0			B	B
Stud.20	A	1	B	4			B	C

Table III reports for each student the collected monitoring data relative to the completion time of each Training Step. These data have been used to compute the teacher KPIs as reported in Tables IV and V.

Considering in particular Table IV in the rows there are the three Training Steps (One, Two:Normal, Two:Low) considered in the BPM of Figure 3 labeled respectively (TS1, TS2N, TS2L). In the columns there are the cumulative KPI values associated to each training step during the executed learning session, i.e. the correctness score (CSTS), the reload cardinality (RCTS), the learning adequacy (SLATS) and the completion time (CTTS). Data reported in the Table IV reveals to the teacher information about the concluded training session.

In particular, considering the Training Step One the values computed for CSTS and CTTS are close to the established best values, indicating that the students seem to have a good preparation on this topic. Even if the overall evaluation (SLATS) is acceptably close to one (reported value is 1.75), the number of reloads (RCTS) is quite close to 1 (reported value is 0,95). This evidenced that most of the times the students try to change the question proposed, that is quite in contradiction with the level of correctness measured. However a deeper investigation of the punctual data of Table II reveals that this is not the case. Indeed most of the students (12 over 20) solve the questions without reloads, and just few of them have a high number of reloads (3 or 4). The anomalous situation evidences us that probably additional KPIs, that could take in consideration also the standard deviation of the data collected, should be included in the final release of the execution and monitoring framework so to better tracing the students learning activity.

Analysis similar to that performed for Training Step One

TABLE III. COMPLETION TIME OF EACH TRAINING STEP

	Boundary:180s	*Boundary:480s*	*Boundary:300s*
	TS1 time	**TS2N time**	**TS2L time**
Stud.1	389		100
Stud.2	121	820	
Stud.3	89	295	
Stud.4	169	511	
Stud.5	140		344
Stud.6	121	373	
Stud.7	116	491	
Stud.8	155	471	
Stud.9	160	60	
Stud.10	209		280
Stud.11	116		388
Stud.12	150	525	
Stud.13	98	422	
Stud.14	225	621	
Stud.15	420	212	
Stud.16	188	690	
Stud.17	192	833	
Stud.18	95	622	
Stud.19	147	321	
Stud.20	167	735	

can be considered also for the other steps. In particular for the Training Step Two:Normal even if the global SLATS value is equal to that of Step One (1,75), a different situation is highlighted from the remaining KPIs. Indeed the value of CSTS (0,5) reveals that the students had a certain difficulty in solving the questions about this topic. This could be due either to a low level of preparation of the students or to a poor clarity of the questions proposed. This is also confirmed by the CTTS value (500.125 seconds) that is over the boundary established (480 seconds) and by the number of reloads (RCTS) that is above 1. In this case the punctual data of Table II confirmed these conclusions. Thus KPIs analysis could be a hint for the teacher to perform further investigation with his/her students to better understand what happened and/or to improve the questions quality.

Finally, analyzing the KPIs relative to the Training Step Two:Low, even if the students seem to reveal a very good preparation on this topic (CSTS equal to 1 and CTTS equal to 278), the number of reloads (RCTS equal to 2.5) shows that students frequently change the proposed question, probably trying to find the one that is more easy to reply. This is also confirmed by the SLATS value that is quite far from the best value 1.

Considering instead the overall training session the analysis of the KPIs, calculated according to the formulas of Section III-B, reveal that in average students have not a proper preparation on the examination topics. This is confirmed both by the correctness levels (CSLS equal to 0,767), by the number of reloads (RCLS equal to 1.567) and by the overall adequacy level (SLALS equal to 4,375)) that are quite far from the best values 2. The overall completion time (CTLS) is instead under the boundary level of 570 seconds.

The preliminary results of this case study reveal that the calculated KPIs, derived by using the monitoring data collected during the learning session, are a very good mean for the evaluation of the level of preparation, both from students and teacher point of view.

TABLE IV. KPI FOR TEACHER PER TRAINING STEP (TS)

	CSTS	**RCTS**	**SLATS**	**CTTS**
TS1	0,8	0,95	1,75	173,35
TS2N	0,5	1,25	1,75	500,125
TS2L	1	2,5	3,5	278

TABLE V. KPI FOR TEACHER PER LEARNING SESSION

CSLS	0,767
RCLS	1,567
SLALS	4,375
CTLS	562,413

IV. RELATED WORK

Nowadays, in many application domains, modeling of business process and KPI evaluation is increasingly gaining importance. However, monitoring that represents the main element for the management of communication networks and distributed systems [14] is assuming a key role for tracking the states of a business process and for evaluating the execution performance [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. The work of [4] provides a model-driven approach for generating a monitoring infrastructure based on events. The goal of this work is to show how KPIs are modeled and transferred into event rules by a model-driven approach.

Existing works [3] 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. Finally, other approaches [15] focus on monitoring business constraints at runtime by means of temporal logic and colored automata. They allow continuous compliance with respect to predefined business process constraint model and recovery after the first violation. Differently from these approaches, the proposed 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.

In the context of learning, that is the application domain of the proposed case study, monitoring solutions can be used for providing feedbacks on training sessions and allow KPI evaluation. Popular web based Learning Content Management Systems (LCMSs) such as WebCT [16], Moodle [17] and Blackboard [18], require teachers to constantly adapt structure and content of their courses according to the feedback of the learning process in order to assure comprehensibility, high performance and learning efficiency of their students. However, contemporary LCMSs provide rather basic feedbacks about the learning process, such as simple statistics on technology usage or low-level data on students activities (e.g., page view).

Some tools have been developed for providing feedbacks on the learning activities by the analysis of the user tracking data. The authors of [19], for instance, propose LOCO-Analyst, an educational tool aimed at providing educators with feedback on the relevant aspects of the learning process taking place in a web-based learning environment such as the usage and the comprehensibility of the learning content or contextualized social interactions among students (i.e., social networking). The main goal of these tools is to support educators for creating courses, viewing the feedback on those courses, and modifying the courses accordingly. Differently from these solutions, our proposal focuses on model-based learning and monitoring of business process execution.

V. 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.

In this paper an example of implementation of the proposed execution and monitoring framework has been presented as well as its validation on a real case study in the learning context. In particular, by using opportunely defined KPI measurement processes, values of interest have been monitored and computed for the performance evaluation of a learning session executed by a group of students of a software engineering university course.

Even if preliminary the experimentation evidenced the effectiveness of the proposed framework in supporting the evaluation at run-time of important KPIs and highlighted the importance of the combination of business events occurring on a BPM execution with information related to timing and paths. However the validity of the experiments and the amount of confidence on the reported results are limited by two different aspects: i) the business process model used as a case study. The BPM is a simplified version of the overall business process model of a training system offered to students of a software engineering course. Thus the number of activities modeled as well as the complexity of the questionnaire management could be extremely increased so to consider other realistic situations. The purpose in this paper was to provide a simpler vision of the BPM to better focus the reader attention to the implementations details of the proposed framework. ii) KPI definition. It is out the scope of the paper an accurate definition of the KPIs values and related measures. The intention was to show the use of monitoring data to derive interesting parameters for BPM management and control. It is likely that other more appropriate KPIs may produce different evaluation results. iii) Questionnaire management. In this case study only two topics are considered and three different prefixed replies (*wrong*, *partially correct*, and *correct*) modeled. This is a very simplified vision of a training system, and an higher variability of replies needs to be considered to guarantee the scalability of the proposed approach.

In this paper we presented a first implementation of the execution and monitoring framework proposed. As a future we would like to validate the proposed framework considering more complex business process models related to different application domains. 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*. Springer, 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*. Springer, 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 *Software Engineering for Resilient Systems*. Springer, 2011, pp. 130–144.

[6] I. Alfresco Software, "Activiti BPM Platform," http://activiti.org/.

[7] "Camunda," http://camunda.org/.

[8] "jBPM," http://www.jbpm.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] Alfresco Software, Inc., "Activiti Execution Listener," http://docs.alfresco.com/activiti/topics/executionListeners.html.

[13] Oracle, "JMS Overview," http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html.

[14] M. Mansouri-Samani and M. Sloman, "Network and distributed systems management," M. Sloman, Ed., 1994, ch. Monitoring Distributed Systems, pp. 303–347.

[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] "WebCT," www.webct.com.

[17] "Moodle," http://moodle.org/.

[18] "Blackboard," www.blackboard.com/.

[19] L. Ali, M. Hatala, D. Gašević, and J. Jovanović, "A qualitative evaluation of evolution of a learning analytics tool," *Computers & Education*, vol. 58, no. 1, pp. 470–489, 2012.