Artifact-Driven Process Monitoring: A Viable Solution to Continuously and Autonomously Monitor Business Processes

Giovanni Meroni(✉)

Politecnico di Milano, Milan, Italy
giovanni.meroni@polimi.it

Abstract. Business process monitoring aims at identifying how well running processes are performing with respect to performance measures and objectives. By observing the execution of a process, process monitoring is also responsible for creating process traces, which can be subsequently used by process mining algorithms to gain further insights on the process.

Among the various monitoring solutions, artifact-driven monitoring has been proposed as a viable solution to continuously and autonomously monitor business processes. By monitoring the changes in the physical and virtual objects (i.e., artifacts) participating in the process, artifact-driven monitoring can autonomously generate traces that include events related to semi-automatic and manual tasks. Also, by relying on a declarative representation of the process to monitor, artifact-driven monitoring can detecting violations in the execution flow as soon as they occur. In addition, artifact-driven monitoring can identify the process elements affected by a violation, and it can continue monitoring the process without human intervention.

This tutorial paper will firstly provide an introduction to process monitoring, and the recent advancements in this field. Then, an overview on how artifact-driven monitoring works will be provided.

Keywords: Business process monitoring · Artifact-driven · Conformance checking

1 Introduction to Process Monitoring

As discussed by Dumas et al. in [2], business process monitoring consists in methods and techniques aiming at collecting and analyzing information on the way business processes are executed. Process monitoring plays a key role in the Business Process Management (BPM) lifecycle, as it allows to verify how well a business process is executed in reality and if the real behavior differs from the one being modeled. The outcome of process monitoring can then be used by the subsequent phases in the BPM lifecycle to optimize the process or to discover undocumented behaviors.

According to the classification proposed by [7] and [6], process monitoring techniques can be classified in the following groups:

- **Event data logging**. Such techniques identify and record in a so-called execution log events related to a specific process instance being executed. Such events can be related to the activities being executed, the artifacts (i.e., the physical or virtual objects) manipulated by the process, or to the resources (i.e., the human operators or software components responsible for executing activities) participating in the process. Since several other monitoring techniques require event data to work, this technique is often seen as a prerequisite for them.
- **Business Activity Monitoring (BAM)**, also known as "monitoring" [7]. Such techniques analyze real-time information on the activities being executed (e.g., response time and failure rate) in order to measure Key Performance Indicators (KPIs) relevant for the process and to determine how well activities are performed.
- **Runtime Performance Analysis.** Such techniques analyze performance information on the processes being executed and identify bottlenecks or resource allocation problems. Unlike BAM, which focuses on single activities, Runtime performance analysis focuses on process runs, thus accounting for dependencies among activities.
- **Conformance Checking.** Such techniques compare the modeled process behavior with the one evidenced by execution data, in order to detect inconsistencies. To do so, they typically replay events in the execution log and see if they fit the process model.
- **Compliance Checking.** Such techniques verify that constraints representing regulations, guidelines, policies and laws are fulfilled by the process. With respect to conformance checking, compliance constraints focus on specific portions of the process, rather than on the entire model. Also, constraints can predicate both on the structure and on non functional aspects, such as execution time and resource allocation.

1.1 Challenges in Process Monitoring

To cope with the ever changing needs of the market, more and more organizations tend to externalize - either partially or completely - their internal business processes, and to establish short-term collaborations. This causes organizations to no longer have full control on how the process is being executed. Thus, process monitoring plays a critical role in this setting. Nevertheless, being able to monitor processes that span among multiple participants is far from trivial. Most monitoring solutions rely on information coming from Business Process Management Systems (BPMSs) or other corporate information systems, which are typically confined within the premises of an organization. Therefore, to monitor collaborative processes, organizations may have to federate their infrastructure, a complex task that may become problematic especially when the organizations need to collaborate only for a short period of time.

Another relevant challenge in process monitoring consists in ensuring that events related to process executions are accurate, timely, and reliable. When the process is fully automated by a single software component, such as a BPMS, it may be sufficient to collect and analyze the execution logs produced by that component. However, when the process is composed of manual activities, obtaining reliable monitoring information becomes more difficult. Indeed, the human operators responsible for such activities have to input information on how the activity was executed. Thus, the operator may forget to send this information, may make mistakes, or may deliberately introduce misleading information.

Finally, being able to continuously and autonomously determine if the execution differs from the model is a challenging task. Most conformance checking techniques operate off-line with complete event logs, thus can provide monitoring information only after the process ended. Conversely, compliance checking techniques can operate in real-time with partial event logs. However, most of them can only indicate if a constraint was satisfied or violated [3]. Therefore, if the process model is treated as a single constraint, compliance checking techniques can only indicate if the execution adheres to the model or not, but they cannot point out where it differs (e.g., an activity could be skipped). Breaking down the model into multiple constraints may address this issue. However, if constraints modeling all the possible discrepancies that may arise are modeled, the complexity of the compliance checks grows exponentially.

2 Artifact-Driven Monitoring in a Nutshell

Artifact-driven process monitoring [4] is a novel technique aiming at addressing the aforementioned challenges. The key idea behind artifact-driven monitoring is that, by observing the evolution of the artifacts participating in a process, it is possible to infer how the process is being executed. In particular, the Internet of Things (IoT) paradigm is exploited to make the physical artifacts in the process smart. Being equipping with sensors, a computing device (e.g., as a single board computer) and a communication interface, physical artifacts can autonomously collect and exchange with each other information on their conditions and on the environment. In addition, by providing them a representation of the process they participate in, physical artifacts can autonomously keep track of how the process is being executed.

The main advantages of artifact-driven monitoring are thus the following:

- **Manual Activities can be Automatically Monitored.** When executed, a manual activity changes the conditions of one or more artifacts. For example, delivering a package changes the position of that package. Therefore, if the conditions of the artifacts involved in that activity are automatically monitored thank to the IoT, the operator responsible for that activity is no longer required to provide information on when and how that activity was executed.

- **Collaborative Processes Can be Easily Monitored.** Physical artifacts are in close contact with the process they participate in, even when such process spans among multiple organizations. Therefore, they can autonomously collect all the information relevant for the process, without having to be federated with the information systems of the external organizations.
- **Deviations from the Modeled Process can be Immediately and Continuously Identified.** Artifact-driven monitoring relies on an artifact-centric representation of the process to monitor, which treats dependencies among activities as descriptive rather than prescriptive. In this way, it is possible to immediately detect when the execution deviates from the model and which portion of the process is affected. When a deviation is detected, monitoring is not stopped and is capable of detecting subsequent deviations.

2.1 E-GSM Modeling Language

To represent the process to monitor, Artifact-driven Monitoring makes use of Extended-GSM (E-GSM), an extension of the Guard-Stage-Milestone (GSM) notation [1]. In particular, activities and process blocks (e.g., exclusive blocks) are modeled as Stages, which are decorated with Data Flow Guards, Process Flow Guards, Milestones and Fault Loggers.

A Data Flow Guard contains an expression that, when evaluated to true, causes the associated stage to become *opened*, meaning that the process portion represented by that stage was started. Similarly, a Milestone contains an expression that, when evaluated to true, causes the associated stage to become *closed*, meaning that the process portion represented by that stage completed its execution.

A Process Flow Guard defines a prerequisite dependency among other stages (e.g., another stage must be closed). It is evaluated when the associated stage becomes *opened* and, if the dependency is not satisfied, it means that the process portion represented by the associated stage is not compliant with the expected execution flow (e.g. an activity was executed before the previous one was finished). Therefore, the associated stage is marked as *outOfOrder*.

A Fault Logger contains an expression that is evaluated as long as the associated stage is *opened*. If that expression evaluates to true, it means that the process portion represented by the associated stage was incorrectly executed (e.g., an activity failed). Therefore, the associated stage is marked as *faulty*.

Expressions contained in Data Flow Guards, Milestones and Fault Loggers can predicate on the conditions of the artifacts. In this way, when a change in the conditions of an artifact is detected, the corresponding expression is triggered, and the execution of the process can be monitored.

2.2 From BPMN to E-GSM

E-GSM is an expressive but complex modeling language. Also, some of the processes one would like to monitor may already have been modeled in Business Process Model and Notation (BPMN), which is a widely adopted standard in process modeling. To address those issues, a semi-automatic method to transform BPMN collaboration diagrams in E-GSM has been proposed. In this way, process designers do not have to learn E-GSM to monitor a process, and can reuse existing process models and modeling tools.

The first step consists in enriching BPMN collaboration diagrams with information on the artifacts participating in the process and their conditions. This way, it is possible to indicate which artifacts are required for an activity to start, and how such an activity alters the artifacts. To do so, BPMN data objects are used to represent the artifacts, data states to represent the conditions of an artifact, and data associations to represent the artifacts required for activities to start and the ones being produced when it finishes.

The second step consists in transforming the BPMN collaboration diagrams into a BPMN process diagram that represents the view artifacts have on the process. To do so, pools are removed and message flows are transformed in process flows. Indeed, as artifacts can travel along different organizations and participate in activities carried out by different organizations, it no longer makes sense to distinguish activities and dependencies based on the organizations.

The final step consists in transforming the BPMN process diagram in an EGSM model. To do so, translation rules that map BPMN elements and patterns into their corresponding BPMN counterparts. As long as the BPMN process diagram is well-structured, translation rules can be automatically applied with no user interaction required [5].

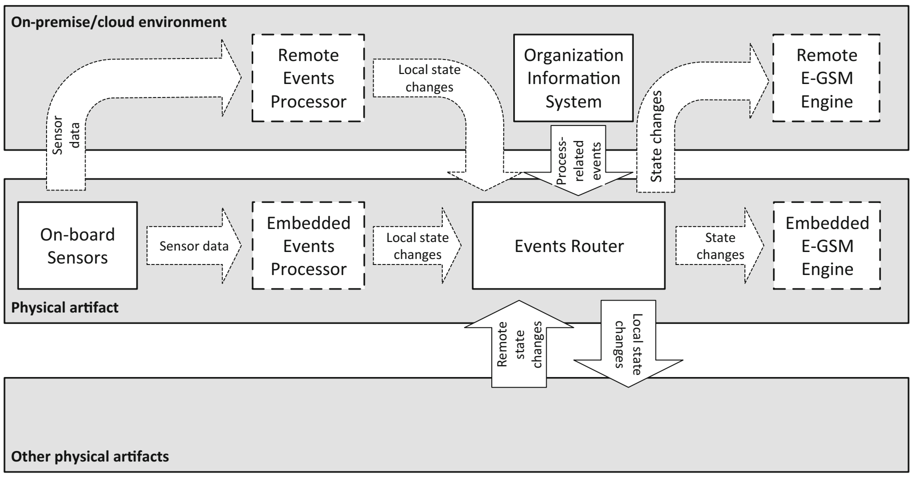

Fig. 1. Reference architecture of our artifact-driven monitoring platform.

2.3 SMARTifact: An Artifact-Driven Monitoring Platform

To implement an artifact-driven monitoring platform, the reference architecture shown in Fig. 1 has been proposed. This architecture is organized along four main modules:

- **On-board Sensors Gateway.** This module runs on each physical artifact, and is responsible for periodically collecting the values coming its sensors.
- **Events Processor.** This module takes as input the data collected by the On-board Sensors Gateway, analyzes them, and determines the state of the artifact. Depending on its complexity and on the computing capabilities of the smart objects, the Events Processor can either run on top of them or remotely in an on-premise or cloud environment.
- **Events Router.** This module runs on each physical artifact, and is responsible for exchanging information with all the physical artifacts, the information systems and the software components involved in the same process execution.
- **E-GSM Engine.** This module contains the E-GSM model of the process to monitor, which is used to determine when activities are executed and if the process deviates from the expected behavior. Whenever the Events Router forwards a new event, the E-GSM Engine examines the event and triggers the expression in the E-GSM model predicating on that event.

This reference architecture was implemented in the SMARTifact platform. In particular, the Events Processor was implemented with the Node-RED flow engine, the events router relied on the Message Queue Telemetry Transport (MQTT) protocol, and the E-GSM engine was implemented in Node.js. The computing requirements were modest enough for the platform to be deployed in an Intel Galileo single board computer.

Acknowledgments. The author would like to acknowledge the input from his colleague and former supervisor Pierluigi Plebani, the Information Systems research group in Politecnico di Milano, and the colleagues Marco Montali and Claudio Di Ciccio in this multi-year research work.

References

1. Damaggio, E., Hull, R., Vaculín, R.: On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. Inf. Syst. **38**(4), 561–584 (2013)
2. Dumas, M., La Rosa, M., Mendling, J., Reijers, A.: Fundamentals of Business Process Management (2013)
3. Ly, L.T., Maggi, F.M., Montali, M., Rinderle-Ma, S., van der Aalst, W.M.P.: Compliance monitoring in business processes: functionalities, application, and tool-support. Inf. Syst. **54**, 209–234 (2015)
4. Meroni, G.: Assessing and improving process monitorability. In: Artifact-Driven Business Process Monitoring. LNBIP, vol. 368, pp. 93–106. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32412-4

5. Meroni, G., Baresi, L., Montali, M., Plebani, P.: Multi-party business process compliance monitoring through iot-enabled artifacts. Inf. Syst. **73**, 61–78 (2018)
6. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)
7. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. **2013**(507984), 37 (2013)