mArtifact: an Artifact-driven Process Monitoring Platform

Luciano Baresi[1], Claudio Di Ciccio[2], Jan Mendling[2],
Giovanni Meroni[1], and Pierluigi Plebani[1]

[1] Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Italy
{luciano.baresi,giovanni.meroni,pierluigi.plebani}@polimi.it
[2] Institute for Information Business
Vienna University of Economics and Business, Austria
{claudio.di.ciccio,jan.mendling}@wu.ac.at

Abstract. Traditionally, human intervention is required to monitor a business process. Operators notify when manual activities are executed, and manually restart the monitoring whenever the process is not executed as expected. This paper presents mArtifact, an artifact-driven process monitoring platform. mArtifact uses the E-GSM artifact-centric language to represent the process. This way, when a violation occurs, it can flag the affected activities without halting the monitoring. By predicating on the conditions of the physical artifacts participating in a process, mArtifact autonomously detects when activities are executed and constraints are violated. The audience is expected to be familiar with business process monitoring and artifact-centric modeling languages.

Keywords: Process monitoring, artifact-centric processes, GSM

1 Contribution to the BPM field

Traditionally, organizations rely on a Business Process Management System (BPMS) to monitor their process. The tight integration between the process execution engine and the monitoring component allows a BPMS to both automate the execution of business activities composing the processes, and keep track of when activities are executed at the same time [5]. This approach works particularly well for intra-organizational processes consisting of completely or semi-automated activities. In this case, the BPMS directly controls when activities are executed. However, when inter-organizational or human-centric processes have to be monitored, a BPMS may experience difficulties.

In inter-organizational processes, organizations directly control only a portion of the process. Consequently, their BPMS cannot enforce the whole process to be executed as defined in a model. Additionally, organizations have visibility of all the activities composing the process only if either a centralized BPMS is deployed, or their BPMSs are federated. Another shortcoming that comes when monitoring inter-organizational process with BPMSs is the way they deal with violations in the execution order of activities. When a violation occurs, the BPMS typically halts until an operator marks the violation as resolved. This behavior is fine for intra-organizational processes, since the BPMS controls both the execution and the monitoring of the process. In inter-organizational processes, on the other hand, this behavior may cause activities executed after the violation from other parties not to be tracked. This issue can be mitigated by forcing the BPMS to ignore violations, and resorting to post-mortem techniques, such as process mining, to detect them. However, mining techniques are meant to be applied after the process ends, thus not permitting organizations to promptly react to violations.

Human-centric processes are characterized by activities completely performed by human operators (e.g., loading a shipping container onto a truck). In these activities, the only interaction with a BPMS occurs to notify when such activities start or end. Consequently, operators can easily forget or postpone such notifications, thus negatively affecting the reliability of the monitoring.

These issues are addressed in mArtifact, a monitoring platform capable to continuously and autonomously monitor business processes. By predicating on the conditions of the physical artifacts interacting with the process, mArtifact can autonomously detect when activities are started and concluded. Also, mArtifact relies on the Extended-GSM (E-GSM) artifact-centric language [3], extension of Guard-Stage-Milestone (GSM) [6], to represent the dependencies among activities without enforcing them. This way, when activities are not executed as specified in the model, mArtifact can detect such a violation and continue monitoring the process without requiring any intervention.

2 Usage of mArtifact on a Use Case

To demonstrate the capabilities of mArtifact, we will rely on a simplified example taken from the logistics domain, depicted in Fig. 1. A manufacturer located in London relies on a shipping company to send a container to a customer in Amsterdam. First, the container of the manufacturer is loaded onto a truck of the shipping company (activity Load container). Then, the truck travels in the UK, takes the Channel tunnel, and travels in the European continent (activity Travel in EU) until it reaches the premises of the customer. Finally, the container is unloaded. Optionally, the truck can take a break while driving in the UK or in the European continent (Take break in UK, Take break in EU). The process will be henceforth indicated as Manufacturer-to-Customer (M-to-C) shipping process.

To monitor this process, mArtifact relies on two software tools: *(i)* the configuration tool, which automatically generates all the information required to monitor the process, and *(ii)* the artifact-driven process monitor.

Configuration Tool. To monitor a process, mArtifact relies on the E-GSM artifact-centric language [3]. Information on the execution order of the activities composing the process is considered descriptive, and not prescriptive as in activity-centric languages. This allows mArtifact not to stop monitoring the process when activities are executed without respecting the defined execution order. Instead, such activities are either flagged as *out of order*, meaning they were executed when they should have not, or as *skipped*, meaning they were not executed while they should have. The configuration tool takes care of translating an input Business Process Model and Notation (BPMN) 2.0 diagram representing the process model in the E-GSM language. Data Objects are required to annotate the diagram to denote the involved artifacts as depicted in Fig. 1. They are connected to activities to indicate through their states the conditions under which they

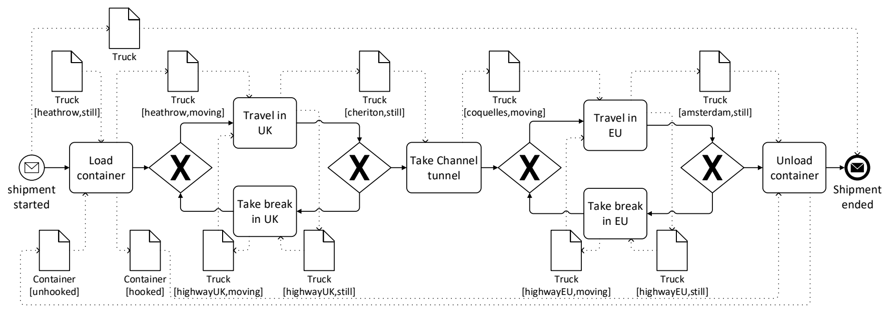


Fig. 1: BPMN diagram of the M-to-C shipping process annotated with artifacts.

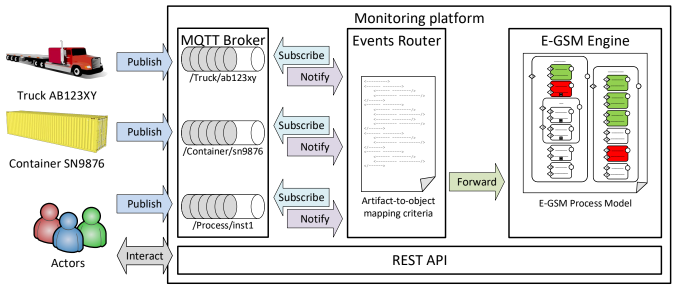
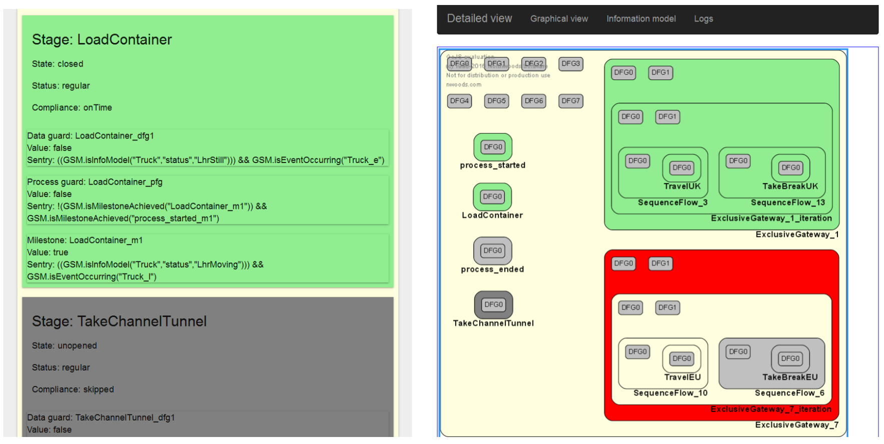


Fig. 2: Architecture of the process monitor.

signal an activity start or end. For instance, the diagram of Fig. 1 indicates that if Truck gets in the [heathrow,still] state or Container gets [unhooked], then Load container starts. The link to start- and end-events in the diagram denotes when the process begins and stops interacting with a specific artifact. For instance, the Truck artifact is meant to be monitored along the execution of the whole M-to-C shipping process. Given the BPMN process model, the configuration tool automatically converts it to E-GSM by using ATLAS Transformation Language (ATL) [7] rules.

Additionally, mArtifact requires to know which physical entities will participate in each specific process execution, e.g., the truck having plate number “AB123XY” representing the real-world instance of the Truck artifact. Since such an information is often known only after the process starts, mArtifact relies on binding criteria which allow for a dynamic association of artifact instances to running processes at run-time, i.e., during the monitoring itself. The configuration tool of mArtifact takes care of the automatic generation of those run-time bindings. Details on the transformations performed are described in [4,8].

Process Monitor. Once the E-GSM model of the process and the binding criteria have been defined, the artifact-driven process monitor comes into play. Fig. 2 shows the architecture of the process monitor. To make the monitoring completely automated, we assume that the physical artifacts can autonomously infer their conditions and submit such an information to the monitor. This is a feasible assumption in the context of a


Stage: LoadContainer

State: closed

Status: regular

Compliance: onTime

Data guard: LoadContainer_dfg1
Value: false
Sentry: ((GSM.isInfoModel("Truck","status","LhrStill"))) && GSM.isEventOccurring("Truck_e")

Process guard: LoadContainer_pfg
Value: false
Sentry: !(GSM.isMilestoneAchieved("LoadContainer_m1")) &&
GSM.isMilestoneAchieved("process_started_m1")

Milestone: LoadContainer_m1
Value: true
Sentry: ((GSM.isInfoModel("Truck","status","LhrMoving"))) &&
GSM.isEventOccurring("Truck_l")

Stage: TakeChannelTunnel

State: unopened

Status: regular

Compliance: skipped

Data guard: TakeChannelTunnel_dfg1
Value: false

Fig. 3: Screenshot of the process monitor showing an incorrect execution.

Wireless Sensor Network (WSN) [1] or the Internet of Things (IoT) [2], where environmental data can be collected by the artifacts.

To allow the artifacts to communicate with the monitor, a *Message Queue Telemetry Transport (MQTT) Broker* is used. MQTT is a queue-based publish/subscribe protocol, which is particularly suited for applications where computing power and bandwidth are constrained.[3] The *E-GSM Engine* is the component responsible for monitoring the execution of the process. This component takes as input the E-GSM model, and *(i)* keeps track of which activities are ongoing, *(ii)* detects whether they follow the execution flow defined in the model and, if not, *(iii)* marks them as not compliant. To support late binding and unbinding among artifacts and running processes, the *Events Router* component is introduced. By receiving as input the binding criteria, the Events Router forwards to the E-GSM Engine only the events produced by the artifacts that effectively take part in that process execution. Finally, a *REST API* offers a service-oriented component for the organizations to interact with the process monitor. It is the software interface through which *(i)* the E-GSM Engine is provided with the E-GSM model, *(ii)* the Events Router is instructed with the binding criteria, and *(iii)* the organizations control that the processes are correctly executed.

Fig. 3 shows a screenshot of the process monitor graphical interface, which displays an incorrect execution of the M-to-C shipping process. In this case, the truck took a ferry from Dover to Calais instead of taking the Channel tunnel. Therefore, the monitor marks Take channel tunnel as *skipped* (dark gray). On the other hand, Load container was executed according to the execution flow, and as such it is marked as *on time* (green). Since the truck has not yet taken a break in the European continent, Take break in EU is not executed yet (light gray). Finally, as the truck is still traveling in the European continent, Travel in EU is still being executed (yellow). Note that, even though a violation occurred, the monitoring is still running.

[3] See `http://mqtt.org/`

3 Platform Maturity

The accuracy of the mArtifact platform was validated by using real-world processes and data from a logistics company. 77 process executions were monitored based on the position and speed of the trucks participating to that process. mArtifact was capable of correctly determining the execution of such processes for 93.13% of all cases. For what the system requirements are concerned, mArtifact is capable of running on very modest hardware. We were able to successfully run mArtifact on different Single-board computers (SBCs), such as the Intel Galileo[4] and Raspberry Pi[5] boards, thus paving the path towards a distributed monitoring performed directly by the physical artifacts taking part in the process.

4 Additional resources

The source code of the mArtifact platform is available at `https://bitbucket.org/account/user/polimiisgroup/projects/MARTiFACT`. A screencast showing a usage demo can be watched at `https://purl.org/polimi/martifact/screencast`.

Acknowledgments. This work has been partially funded by the Italian Project ITS Italy 2020 under the Technological National Clusters program, and the EU H2020 program under MSCA grant agreement 645751 (RISE_BPM). The authors thank Marco Spelta for his help with the implementation of the E-GSM engine.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393 – 422 (2002)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks 54(15), 2787–2805 (2010)
3. Baresi, L., Meroni, G., Plebani, P.: A GSM-based Approach for Monitoring Cross-Organization Business Processes using Smart Objects. In: BPM 2015 Workshops, pp. 389–400. Springer (2016)
4. Baresi, L., Meroni, G., Plebani, P.: Using the guard-stage-milestone notation for monitoring bpmn-based processes. In: BPMDS EMMSAD 2016, pp. 18–33. Springer (2016)
5. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
6. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath, Fenno(Terry), I., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculin, R.: Introducing the Guard-Stage-Milestone Approach for Specifying Business Entity Lifecycles. In: WS-FM 2010, pp. 1–24. Springer (2011)
7. Jouault, F., Allilaire, F., Bézivin, J., Kurtev, I., Valduriez, P.: Atl: A qvt-like transformation language. In: OOPSLA '06 Companion. pp. 719–720. ACM (2006)
8. Meroni, G., Di Ciccio, C., Mendling, J.: Artifact-driven process monitoring: Dynamically binding real-world objects to running processes. In: CAiSE '17 Forum. pp. 105–112. CEUR-WS.org (2017)

[4] See `https://software.intel.com/en-us/iot/hardware/galileo`.
[5] See `https://www.raspberrypi.org`.