Artifact-driven Process Monitoring: Dynamically Binding Real-world Objects to Running Processes

Giovanni Meroni[1], Claudio Di Ciccio[2], and Jan Mendling[2]

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

Abstract. Monitoring inter-organizational business processes requires explicit knowledge about when activities start and complete. This is a challenge because no single system controls the process, activities might not be directly recorded, and the overall course of execution might only be determined at runtime. In this paper, we address these problems by integrating process monitoring with sensor data from real-world objects. We formalize our approach using the E-GSM artifact-centric language. Since the association between real-world objects and process instances is often only determined at runtime, our approach also caters for dynamic binding and unbinding at runtime.

Keywords: E-GSM, Artifact-driven Process Monitoring, Dynamic Binding

1 Introduction

The monitoring of business processes is well understood and readily supported by modern Business Process Management Systems (BPMSs) [5]. In an intra-organizational setting, the process engine of the BPMS keeps track of the state and state changes of individual process instances, and the start and completion of activities is directly recorded in the system. The information captured by the BPMS is in such a case sufficient to the monitoring purposes.

Monitoring becomes more challenging when inter-organizational processes have to be monitored and activities are not automated, i.e., performed by humans. Typically several stakeholders are involved, each one controlling only a portion of the process. In that case, the way to inform a BPMS on when activities are executed is to manually send a notification. Therefore, human operators have to notify the BPMS about the start and completion of each activity. Such a task disrupts their work and can be easily forgotten or postponed, thus negatively affecting the reliability of the monitoring.

Another disadvantage of a centralized BPMS is the difficulty to deal with variations in the distributed control flow. Should activities not be executed in the right order, the typical reaction of a BPMS is to raise an exception and consequently abort the process at once. This is undesirable: Because the process is distributed among different stakeholders, some of them may choose to ignore the exception and continue running the process fragment under their own control. This would mean that any activity performed

X. Franch J. Ralyté, R. Matulavi iųs, C. Salinesi, and R. Wieringa (Eds.):
CAiSE 2017 Forum and Doctoral Consortium Papers, pp. 105-112, 2017.

after the exception would not be tracked. A conventional approach to solve this issue is to suppress the BPMS default process abort at violations, and resort on process mining techniques to detect which portion of the process was affected. However, such techniques are meant to be applied a posteriori, thus permitting the stakeholders to know how the process actually unfolded only after it finished.

To overcome these issues, in this paper we propose an approach that relies on the information coming from the artifacts involved in the process, rather than on the stakeholders, to understand how the process evolves. By adopting the artifact-centric language Extended-GSM (E-GSM) [1], extension of Guard-Stage-Milestone (GSM) [9], it is possible to monitor the process even when the control flow is not respected. Also, our approach caters for the dynamic binding of the real-world objects impersonating the artifacts with the process instances at run-time.

The remainder of this paper is structured as follows. Section 2 introduces a motivating example used to describe our approach in Section 3. Finally, Section 4 reports on the related work and Section 5 concludes the paper outlining possible future research.

2 Motivating Example

To better understand the motivations behind our work, in this section we present an example process taken from the logistics domain. The process is depicted with the Business Process Model and Notation (BPMN) diagram in Fig. 1. The upper portion represents the overall process, where each leg is modeled as a subprocess. The process has five main stakeholders: A producer (henceforth, P), the wholesale customer (W), and the transportation companies operating via road with trucks (T), via open-sea with cargo-ships (O), and via rail with trains (R). P provides its products to W, and to do so it relies on the following plan: The process starts when the goods are inside a shipping container (start event Process started). Then, the container is shipped by truck from the warehouse of P to terminal X (first mile). We indicate the first leg of the transport as activity Ship to X. If the container arrives at X during a workday, it is then shipped to terminal Y by rail (Ship to Y), otherwise it is shipped to terminal Z via open-sea (Ship to Z). Finally, the container is shipped to W from either Y or Z by truck. The transportation leg along the last mile is denoted as Ship to customer. Once W receives the goods, the process ends (end event Process ended).

The lower portion of Fig. 1 depicts the expanded subprocess Ship to X, carried out by T via truck. The container is firstly loaded onto the means of transport (Load container), which subsequently delivers the container to the planned destination (Deliver container). Here the container is inspected for damages occurred during the shipment (Inspect container), and finally it is unloaded from the means of transport (Unload container). The expanded subprocesses of Ship to Y, Ship to Z, and Ship to Customer are not drawn here for the sake of space. The activities involved are the same as Ship to X, although they differ for the source and destination of the shipment, the stakeholder involved (resp. R and O), and the means of transport adopted. Note that T has control only on the first and last mile, whereas the intermediate shipment is performed either by R or O. P and W, who are the only stakeholders interested in knowing how the whole process is being executed, have no direct control on the shipment.

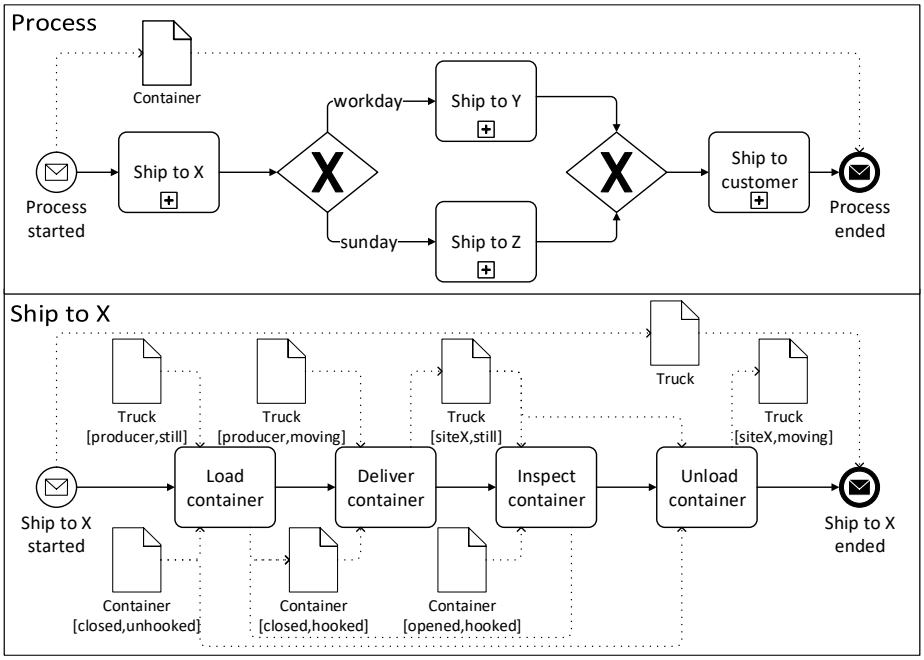

Fig. 1. BPMN diagram of the running example: High-level process model (top), and expanded subprocess Ship to X (bottom).

3 Approach

To autonomously monitor the process, our E-GSM-based monitoring solution needs to know *(i)* how the process is structured and which artifacts interact with the process, and *(ii)* which real-world objects impersonate the artifacts.

The artifacts are used to identify which activities are executed. The underlying idea is that when an activity is running, it alters the state of one or more artifacts. We can thus infer which activities are being carried out by detecting a change in the state of the artifacts. Note that, for an activity to be autonomously monitorable, it must alter the state of at least one physical artifact. Otherwise, explicit notifications are still needed to determine its activation or termination. On the other hand, the structure of the process is used to detect if compliance violations occur: When the execution of a process instance differs from the one defined in the process structure, we can report such a discrepancy. Thus, we can identify which activities are affected, and mark them as non-compliant.

Because different real-world objects embodying the same artifact exist (e.g., different trucks), a binding among each artifact and the impersonating real-world object must be defined. To maximize the flexibility of our solution, such a binding is definable at runtime: Oftentimes it is possible to know which objects participate in a specific process instance only after that process instance started. Since the same object may participate in multiple process instances, at some point in time the information on its state may be relevant to some running process instances and not to other ones. Therefore, it should

be possible to unbind the object from the process instances once its state becomes no longer relevant.

To easily produce all such information, we propose a three-steps procedure. It starts from a BPMN process diagram representing the process to be monitored. The first step requires the designer to enrich the BPMN diagram by including information on the artifacts participating in the process. The second step automatically translates the BPMN diagram into an E-GSM process, suited for monitoring distributed processes. The third step automatically defines criteria to map real-world objects to the artifacts at runtime.

3.1 Enrichment of the BPMN process model with artifacts

A BPMN process diagram specifies a.o. the activities of a process and their control-flow relationships. However, to be able to infer when activities start or end based on the state of the artifacts, the diagram must capture this information. To this extent, we resort on the standard BPMN *data objects*, rather than introducing yet another extension of BPMN. Data objects traditionally serve for documentation purposes, yet we use them to model the artifacts and their interactions with the process. In particular, the artifacts and their states specified in input and output data objects indicate the triggers for the activities to start and end.

Furthermore, the following binding and unbinding mechanisms among artifacts and real-world objects must be specified in the diagram: *(i)* When an artifact starts interacting with the process, *(ii)* how the object impersonating the artifact is notified to the process, and *(iii)* when an artifact is no longer related to the process. To do so, we rely on the following rules:

- For each artifact, at least one output data object with no data state must be defined in the diagram and associated to a start event. The artifact is supposed to begin interacting with the process when that event occurs. Beforehand, the artifact and its state are ignored. The payload of the event indicates the object that instantiates the artifact. For instance, Truck interacts with the process once Ship to X started occurs.
- For each artifact, zero or more input data objects with no data state can be defined in the diagram and associated to an end event. The artifact is supposed to become unrelated to the process when the event occurs (after such an event, the artifact and its state will be ignored when the process is executed). For instance, Truck will be no longer related to the process once Ship to X ended occurs.

Notice that the same set of input (output) data objects should be associated to multiple activities only if all such activities are expected to start (end) simultaneously. Similarly, the artifacts specified in input and output data objects are expected to assume the indicated states only when the associated activity starts or ends, respectively.

Example. Let us consider again the BPMN process model shown in Fig. 1. The input and output data objects of Unload Container indicate the preconditions and postconditions for that activity to be executed. To execute Unload Container, the container must be closed and unhooked from the truck, and the truck must already be parked in terminal X. When Unload Container finishes, the truck will leave terminal X. As the container participates in the whole process, its data object is associated to the start and

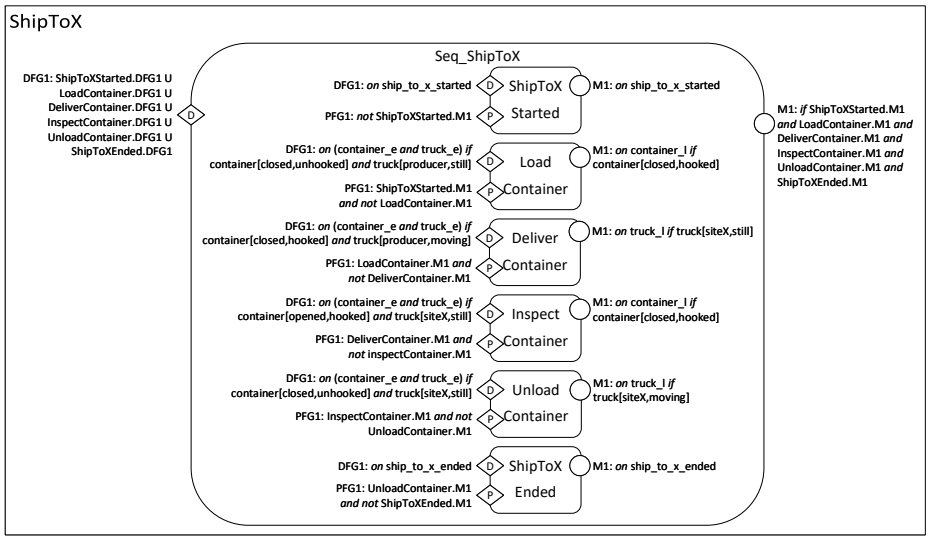

Fig. 2. E-GSM model derived from the subprocess Ship to X.

end events of the process. On the other hand, a truck is needed only for the Ship To X and Ship to Customer subprocesses – the shipments carried out by the trucks. As such, the data object representing the truck is associated only to the start and end events of these subprocesses.

3.2 Generation of the E-GSM process model

Due to its imperative nature, BPMN treats the control flow information in a prescriptive way: The only possible executions of the process are the ones that comply with the control flow. Therefore, any other execution cannot take place. To overcome this limitation, we make use of the E-GSM language [1], an extension of the Guard-Stage-Milestone notation [9] expressly suited for monitoring. E-GSM treats the control flow in a descriptive way, and as such it can monitor any possible execution of a process: When a deviation from the control flow is detected, an E-GSM engine simply marks the portion of the process that caused such a deviation as non compliant, without interrupting the monitoring.

Starting from the enriched BPMN process model obtained in the previous step, an E-GSM model of that process can be automatically produced. To do so, we extend the translation rules defined in [2] by taking into account also the data objects associated to activities. In particular, for every stage S derived from an activity A, its Data Flow Guard (Milestone), responsible for detecting the activation (termination) of S, is evaluated whenever any of the artifacts Ar associated to each input (output) data objects of A changes state. S starts (ends) when the state assumed by all Ar's is the one indicated by the input (output) data objects of A.

Example. Fig. 2 shows the E-GSM process model derived from the BPMN process model of Fig. 1. To mark `UnloadContainer` as opened (i.e., the container is currently being unloaded from the truck), `unloadContainer.DFG1` requires that Truck is siteX, still, and Container is closed, unhooked. To mark `UnloadContainer` as closed (i.e., the unloading of the container finished), `UnloadContainer.M1` requires that Truck is siteX, moving. Finally, to ensure that `UnloadContainer` is executed at the right time, `UnloadContainer.PFG1` requires that `UnloadContainer` has not already been executed (thus requiring `UnloadContainer.M1` not to be achieved). Also, `UnloadContainer.PFG1` requires that `InspectContainer`, which directly precedes `UnloadContainer`, has already been executed (thus requiring `InspectContainer.M1` to be achieved).

3.3 Generation of the artifact-to-object mapping criteria

The E-GSM model generated in the previous step allows us to detect when activities are executed based on the state of the artifacts participating in the process. However, the E-GSM model does not indicate which real-world object will impersonate each artifact (e.g., the artifact `Truck` is impersonated by the physical truck having license plate "AB123XY"). We capture the mapping criteria among artifacts and objects in a separate document. This choice allows us to decouple the process logic, which is carried out by an E-GSM engine, from the artifact instantiation logic, carried out by a separate software module, named Events Router.[3] Based on the mapping criteria, the Events Router notifies the E-GSM engine only of those events related to objects that participate to the process being monitored. We remark here that through this approach the binding of real-world objects with information artifacts in the model is dynamically established at run-time.

Starting from the enriched BPMN process model obtained in the first step, the criteria to map real-world objects to the artifacts can be applied in an automated way. To do so, the following rules are applied:

- Each data association between a BPMN start event and a data object is translated to a mapping criterion. The criterion states that, whenever the start event is detected, the artifact represented by the data object is bound to the real-world object identified in the payload of the event. Should the artifact be already bound to a different real-world object, the new binding would replace the existing one. For instance, when the event ship_to_x_started occurs, Truck is bound to the real-world truck whose license plate is specified in the payload of ship_to_x_started, e.g., "AB123XY".
- Each data association between a data object and a BPMN end event is translated to a mapping criterion. The criterion states that, whenever the end event is detected and the artifact represented by the data object is bound to a real-world object, it gets unbound. For instance, when the event ship_to_x_ended occurs, no real-world truck is bound anymore to Truck.

Example. Fig. 3 shows an excerpt of the artifact-to-object mapping criteria derived from the BPMN process model of Fig. 1. Because the Container artifact interacts with the

[3] Source code at https://bitbucket.org/polimiisgroup/eventsrouter.

```
<Mapping>
        <Artifact name="Container">
            <BindingEvent id="process_started"/>
            <UnbindingEvent id="process_ended"/>
        </Artifact>
        <Artifact name="Truck">
            <BindingEvent id="ship_to_x_started"/>
            <BindingEvent id="ship_to_customer_started"/>
            <UnbindingEvent id="ship_to_x_ended"/>
            <UnbindingEvent id="ship_to_customer_ended"/>
        </Artifact>
[...]</Mapping>
```

Fig. 3. Artifact-to-object mapping criteria.

whole process, the binding is expected to occur when the process starts, and the unbinding to occur once the process finishes. Therefore, to bind a physical container to Container, the event process_started should occur. Once process_started is detected, Container is bound to the container whose unique identifier (e.g., its serial number) is equal to the one specified in the payload of process_started. To unbind Container, process_ended should occur. The `Truck` artifact, on the other hand, interacts with both the subprocesses Ship To X and Ship To Customer. Therefore, to bind a physical truck to Truck, either the event ship_to_x_started or ship_to_customer_started should occur. Similarly, to unbind Truck, either ship_to_x_ended or ship_to_customer_ended should occur.

4 Related Work

To monitor the execution of processes based on external data, Baumgrass et al. [3] integrate a BPMN Engine with a Complex Event Processing (CEP). Cabanillas et al. [4], on the other hand, annotate activities with constraints monitored when the process is executed. However, none of these solutions deals with deviations in the execution flow.

Methods to translate imperative languages to GSM are proposed a.o. in [10,7,11,14,6] All these solutions treat control flow information in a prescriptive way. On the other hand, our solution extends [2], and as such it treats control flow in a descriptive way.

A GSM-based collaboration hub to ease the coordination of logistics processes is proposed in [12]. However, it relies on explicit notifications to determine when activities are executed. [8] overcomes this limitation by adopting the Internet of Things (IoT) paradigm: they rely on sensor data coming from smart objects to activate and deactivate stages. However, the GSM model is expected to be modeled from scratch. Also, they lack mechanisms to detect deviations among the execution of the process and the model.

Concerning the enrichment of process models with information on the data manipulated during execution, Meyer et al. [13] also propose the adoption of BPMN data objects to model such information. With respect to our work, they focus on process execution rather than monitoring. Also, they rely on an extension of the BPMN syntax.

5 Conclusions and Future Work

This paper has presented an approach based on E-GSM to monitor the execution of distributed business processes based on the status of the manipulated artifacts. Mechanisms

to dynamically bind and unbind to a process execution the participating real-world objects have also been defined.

Currently, our approach supports only one-to-one mappings among real-world objects and artifacts. Furthermore, the user has to manually check if the BPMN diagrams are correctly and sufficiently annotated for the process to be autonomously monitored. Future work will aim to address these limitations by including one-to-many and many-to-many mappings, and by introducing tool support to evaluate and improve the annotations.

Acknowledgments

This work has been partially funded by the Italian Project ITS Italy 2020 under the Technological National Clusters program.

References

1. 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)
2. 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)
3. Baumgrass, A., Herzberg, N., Meyer, A., Weske, M.: BPMN extension for business process monitoring. In: EMISA 2014, pp. 85–98. GI (2014)
4. Cabanillas, C., Di Ciccio, C., Mendling, J., Baumgrass, A.: Predictive Task Monitoring for Business Processes. In: BPM 2014, pp. 424–432. Springer (2014)
5. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)
6. Eid-Sabbagh, R.H., Hewelt, M., Meyer, A., Weske, M.: Deriving Business Process Data Architectures from Process Model Collections. In: ICSOC 2013, pp. 533–540. Springer (2013)
7. Eshuis, R., Van Gorp, P.: Synthesizing data-centric models from business process models. Computing pp. 1–29 (2015)
8. Gnimpieba, Z.D.R., Nait-Sidi-Moh, A., Durand, D., Fortin, J.: Using internet of things technologies for a collaborative supply chain: Application to tracking of pallets and containers. Procedia Computer Science 56, 550 – 557 (2015)
9. 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)
10. Köpke, J., Su, J.: Towards quality-aware translations of activity-centric processes to guard stage milestone. In: BPM 2016. Springer (2016)
11. Kumaran, S., Liu, R., Wu, F.Y.: On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: CAISE 2008, pp. 32–47. Springer (2008)
12. Meijler, T.D., Stollberg, M., Winkler, M., Erler, K.: Coordinating variable collaboration processes in logistics. In: MITIP 2011 (2011)
13. Meyer, A., Pufahl, L., Fahland, D., Weske, M.: Modeling and Enacting Complex Data Dependencies in Business Processes. In: BPM 2013. Springer (2013)
14. Meyer, A., Weske, M.: Activity-Centric and Artifact-Centric Process Model Roundtrip. In: BPM 2013 Workshops, pp. 167–181. Springer (2014)