GET Controller and UNICORN: Event-driven Process Execution and Monitoring in Logistics

Anne Baumgrass[1], Claudio Di Ciccio[2], Remco Dijkman[3], Marcin Hewelt[1], Jan Mendling[2], Andreas Meyer[1], Shaya Pourmirza[3], Mathias Weske[1], and Tsun Yin Wong[1]

[1] Hasso Plattner Institute, University of Potsdam, Germany
[2] Institute for Information Business at WU Vienna, Austria
[3] Eindhoven University of Technology, The Netherlands

Abstract Especially in logistics, process instances often interact with their real-world environment during execution. This is challenging due to the fact that events from this environment are often heterogeneous, lack process instance information, and their import and visualisation in traditional process engines is not sufficiently supported. To address these challenges, we implemented GET Controller and UNICORN, two systems that together enable event-driven process execution and monitoring. Their application is shown for a logistics scenario.

1 Overview

Process instances often interact with a real-world environment during their execution. This especially holds true in the context of logistics, where transportation activities geographically move goods around the globe. Thus, the monitoring of such processes should leverage on the observation of phenomena that happen outside the typical scope of Business Process Management Systems (BPMS), in order to have a clear understanding of their evolvement. We refer to such events as *external events*. They not only comprise information updates about passive resources [5] such as position and speed of transport vehicles like trucks or aeroplanes, but also pertain to facts occurring outside of process boundaries, yet affecting the process instance, such as alerts on strikes, traffic congestions, or adverse weather. To date, traditional BPMSs are not able to correlate those external events to the events of their execution logs [5].

This paper presents an implemented approach that is capable of connecting external events from various heterogeneous sources with the transportation execution, promptly notifying about disruptive conditions at run-time when dangerous situations occur. We devised and implemented the presented software in the context of the GET Service project,[4] a European research project aiming at the realisation of a platform for green transportation and smart logistics.

2 Implementation

Due to the separation of concerns principle [6], we implemented two interacting systems: *(i) UNICORN*, which is responsible for collecting, processing, and distributing

[4] `http://getservice-project.eu`

events relevant for logistics from various heterogeneous sources, and *(ii) GET Controller*, which executes and monitors logistics process instances thereby considering events provided by *UNICORN*. The interaction between these components is event-based, and their communication follows the publish-subscribe paradigm. To this extent, process models are encoded in a dialect of BPMN [5], which contains ad-hoc annotations that specify both the subscription and publishing mechanisms for events relevant to the process. Our adapted BPMN modeling language is detailed in [3].

UNICORN[5] is the system handling heterogeneous events in different formats. It is implemented in Java 7 and can be accessed through a web service (SOAP) as well as through a web-based user interface (UI) written using the Apache Wicket framework. Events can be published to UNICORN in XML, Excel, CSV, and JSON formats. As a prerequisite for handling events, the corresponding event types must be defined, either via the UI or via the web service importing its XSD. In UNICORN, Esper [2] is used for real-time processing of events, while MySQL is used to store events for historical analysis. UNICORN implements the publish-subscribe paradigm through the Java Message Service (JMS) implementation of ActiveMQ. Furthermore, it offers a generic interface to periodically request or receive events from different sources such as service providers for traffic and weather information.

To address logistics-specific requirements, UNICORN stores routes and its waypoints (locations a vehicle has to pass), which are relevant in logistics-specific event processing. In this way, we are able to determine the distances between the current position of vehicles or events and their waypoints, to estimate remaining travel times, and to filter traffic incidents in the vicinity of routes. Thus, UNICORN provides high-level, transportation-related event notifications, such as expired deadlines, delays, or missing connections to its subscribers [1].

The logistics-specific process execution and monitoring is done in *GET Controller*[6]. For the implementation, the Activiti[7] engine has been adopted. Activiti is a BPMN-based open-source process engine, encoded in Java. Given a BPMN process model as input, it enacts the corresponding process instances. The embedded task handler *(i)* allows granted users to execute tasks in the UI, and *(ii)* monitors the status of tasks in execution (e.g., *executing* or *completed*). The task handler of GET Controller is illustrated in Figure 1. On the upper-left side, all running instances are shown in a list. In the lower-left corner, the events relevant for the currently selected process instance (*case 54*) are shown: a diverted flight and a strike in Frankfurt. Both are also visualized on the map in the middle of the screen. On the right hand side, we provide the tasks of the process including their current status. Their execution may be triggered by external events but can also be performed by the user, which is then done through the offered buttons or forms in the lower-right corner.

Following the requirements identified in the GET Service project for integrating process and event processing [7], we extended the core of Activiti with the capability to read and interpret annotated BPMN models. Thereupon, we are able to guarantee *(i)* event-driven execution and *(ii)* monitoring capabilities. For guaranteeing both

[5] `https://bpt.hpi.uni-potsdam.de/unicorn`
[6] `http://is.ieis.tue.nl/research/getservice/`
[7] `http://www.activiti.org/`

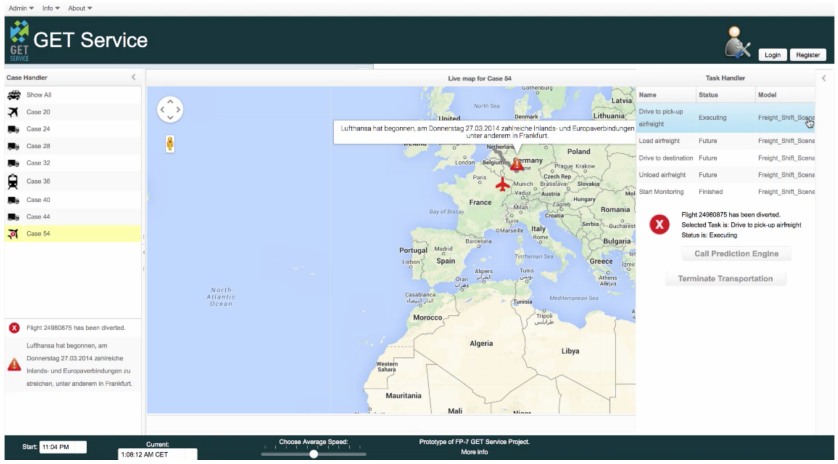

Figure 1. A screenshot of GET Controller

features, GET Controller receives process-execution-relevant events published by UNICORN. They can be external events as well as events signalling the task execution from process clients, e.g., a driver loading goods. Generally, the subscription to events is derived from the annotations in the deployed process model. We distinguish between event subscriptions for *(i)* a task, *(ii)* a group of tasks, or *(iii)* the whole process. Furthermore, these annotations determine for how long subscriptions are valid, i.e. *(i)* from the enabling to the completion of a specified task, *(ii)* as long as at least one task in the given group is running, and *(iii)* along the whole process execution repectively.

3 Execution and Monitoring in GET Controller

GET Controller interacts with UNICORN as shown in Figure 2. The communication comprises five steps:

1. GET Controller deploys a process model which is then used for registering subscriptions to UNICORN. A route can be provided in addition to localise events relevant along this route for the user. For instance, to determine the routes of trucks or to pick up airfreight at Frankfurt airport.
2. The execution of activities can be done via GET Controller (as shown in the panel on the upper-right corner of the screenshot in Figure 1). However, in case of logistics scenarios, GET Controller often derives progress of activities from events received by UNICORN. For instance, rules in UNICORN can be used to correlate the movement of a truck via GPS position events and to evaluate whether it would finally reach its destination as specified (based on information contained in the annotated process model or the corresponding route). The same rules can be defined for flight events or any other movement and activity.
3. Each time an event is published to UNICORN (either by GET Controller or other event sources), its match to a subscription is evaluated. For example, users can subscribe to strike warnings at Frankfurt airport for their transportation process.

This has to be done using the annotation mechanism for process models described in [3]. In the same way, users may subscribe to flight positions as well as diversions that are aggregated by corresponding rules and extensions in UNICORN.

4. In case of a match, an event notification is sent to GET Controller. This notification is then visualised for the affected process instance(s) to the user. The affectedness is computed through the scope and the parameters of a subscription. For example, all processes monitoring a flight will receive that information, while the movement of a single truck will only be shown for the corresponding transportation task.
5. Finally, if the transportation process is completed, GET Controller may decline its corresponding subscriptions and must unsubscribe from UNICORN.

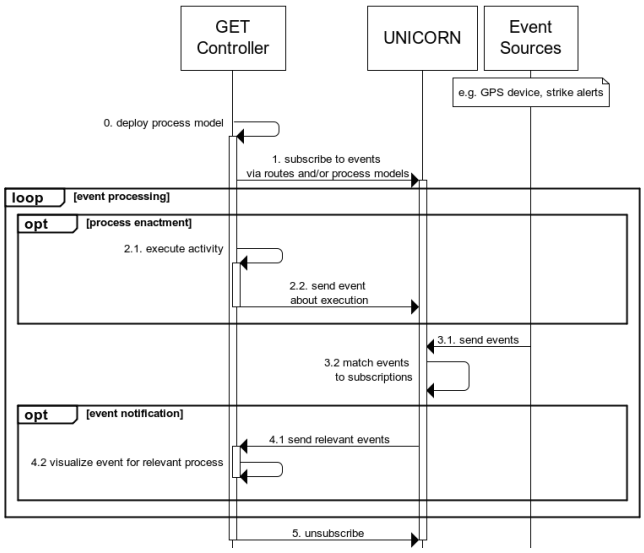

Figure 2. Interaction between UNICORN and GET Controller.

4 Application

The application of both UNICORN and GET Controller is shown using a scenario stemming from the EU-FP7 GET Service project. The scenario pertains to a multi-modal transportation managed by a Logistics Service Provider (LSP), involving a first leg via aircraft and a subsequent truck-based one. In particular, we envision a situation in which the aeroplane lands in a different location than expected, thus requiring the LSP to cancel (or reroute) the reserved trucks, while reserving other vehicles to pick up the cargo at the new airport. We refer to such a situation as *freight shift*. Occasionally, one shift can cause re-planning and re-positioning of 20-100 trucks within 4 hours, with the clear consequence of longer waiting times and empty miles run by the trucks. In order to introduce corrective actions as soon as possible and avoid such situations, it is crucial to constantly monitor freight execution activities, involved resources as well as all external events. Furthermore, mechanisms are required to correlate all this information, and identify which events are currently relevant for the running process instances.

For our demonstration, we used data stemming from a real-world diversion of an aeroplane on the route from Nice to Frankfurt, landing in Brussels. We extracted the data out of a set of examined 6,495 European flight tracks (1,765,172 events) gathered in July 2013. Flights diverted in 49 cases (0.7%). These diversions were automatically detected by the approach presented in [4], based on machine learning algorithms. Specifically for the freight shift used in this paper, this algorithm was able to forecast the diversion of the flight to Frankfurt 1 hour and 18 minutes ahead of the expected landing. For the sake of the demonstration, a Java-based applications was implemented to replay the real-world flight events, along with the event announcing the strike at Frankfurt airport, as reported by Streikradar[8]. This allowed us to better control the occurrences of the right events at the right time as shown in the screencast `https://www.youtube.com/watch?v=JE2Df7iaERk`. The screencast shows how the user is able to execute logistics-specific processes and at the same time get notified about external events that influence its transportation at the time they happen or are predicted automatically.

In summary, we are able to execute and monitor single-modal and multi-modal transportations as sketched through the example above. The progress of the flight and trucks as well as all happenings at the landing airport, such as an announced strike, are monitored. Furthermore, once the prediction of a flight diversion is made, UNICORN automatically forwards the corresponding event notification to GET Controller.

Acknowledgement. The research leading to these results is part of the GET Service project and has received funding from the European Commission under the 7th Framework Programme (FP7) for Research and Technological Development under grant agreement 2012-318275.

References

1. A. Baumgrass, M. Hewelt, A. Meyer, A. Raptopoulos, J. Selke, and T. Wong. Prototypical implementation of the information aggregation engine. GET Service Deliverable report D6.3, 2014.
2. T. Bernhardt and A. Vasseur. Esper: Event stream processing and correlation, 2007.
3. M. Botezatu and H. Völzer. Language and meta-model for transport processes and snippets. GET Service deliverable report D4.1, 2014.
4. C. Cabanillas, C. Di Ciccio, J. Mendling, and A. Baumgrass. Predictive task monitoring for business processes. In *BPM*, pages 424–432. Springer, 2014.
5. M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management*. Springer, 2013.
6. D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.
7. S. Treitl, P. Rogetzer, M. Hrušovský, C. Burkart, et al. Use cases, success criteria and usage scenarios. GET Service deliverable report D1.2, 2014.

[8] `http://streikradar.de`

References

Baumgrass, Anne, Claudio Di Ciccio, Remco M. Dijkman, Marcin Hewelt, Jan Mendling, Andreas Meyer, Shaya Pourmirza, Mathias Weske, and Tsun Yin Wong (2015). "GET Controller and UNICORN: Event-driven Process Execution and Monitoring in Logistics". In: *BPM (Demos)*. Ed. by Florian Daniel and Stefan Zugal. Vol. 1418. CEUR Workshop Proceedings. CEUR-WS.org, pp. 75–79. URL: http://ceur-ws.org/Vol-1418/paper16.pdf.

BibTeX

```
@InProceedings{   Baumgrass.etal/BPMDemos2015:GETControllerUNICORN,
  author        = {Baumgrass, Anne and Di Ciccio, Claudio and Dijkman, Remco
                   M. and Hewelt, Marcin and Mendling, Jan and Meyer, Andreas
                   and Pourmirza, Shaya and Weske, Mathias and Wong, Tsun
                   Yin},
  title         = {{GET} Controller and {UNICORN}: Event-driven Process
                   Execution and Monitoring in Logistics},
  booktitle     = {BPM (Demos)},
  year          = {2015},
  pages         = {75--79},
  publisher     = {CEUR-WS.org},
  crossref      = {BPM2015Demos},
  url           = {http://ceur-ws.org/Vol-1418/paper16.pdf}
}
@Proceedings{     BPM2015Demos,
  title         = {Proceedings of the {BPM} Demo Session 2015 Co-located with
                   the 13th International Conference on Business Process
                   Management ({BPM} 2015), Innsbruck, Austria, September 2,
                   2015},
  year          = {2015},
  editor        = {Florian Daniel and Stefan Zugal},
  publisher     = {CEUR-WS.org},
  series        = {{CEUR} Workshop Proceedings},
  volume        = {1418},
  url           = {http://ceur-ws.org/Vol-1418}
}
```