A Flexible Event Handling Model for Using Events in Business Processes

Sankalita Mandal [1]

Abstract: Business process management (BPM) enables modeling, implementing and monitoring organizational processes to achieve certain business goals. As organizations continue to strive for agility, they have started taking advantage of the digitization and bring flexibility in their processes by several means. One of these is to integrate complex event processing (CEP) with business processes. Event handling specifies how a process communicates with its environment and how this environment influences the execution of the process. Though highly expressive and feature-rich languages like BPMN exist for process specification, they still lack an unambiguous semantics for event handling in different situations. In this work, an event handling model is proposed that take into account the possibilities of subscribing to an event at different point in time with respect to process execution. The model is grounded with formal semantics using Petri Nets and trace analysis to ensure correct execution of process behavior as well as the temporal dependencies among event subscription, event occurrence, event consumption and event unsubscription.

Keywords: Process execution; Event handling; Event subscription; BPMN

1 Introduction

Process model defines a set of activities to achieve certain business goals [We12] in an organizational context. These activities are connected with causal and temporal dependencies using control-flow. A process model also specifies how a process is supposed to interact with the environmental occurrences, represented as events [EN10]. Process engines subscribe to event sources and react to events emitted by these sources according to the process specification. Often, a complex event processing engine is used to abstract from the complexities of connecting to different event sources emitting events in different formats and aggregating them to receive the business level event [HMW13].

Business Process Model and Notation (BPMN) [OM11] is the industry standard for modeling and executing processes. BPMN includes explicit event constructs to depict the production and consumption of events, message flows to link external process participants, and control-flow routing based on events such as event-based gateways or boundary events. However, the event handling semantics in BPMN is pretty limited while it comes to specify when

[1] Business Process Technology Group, Hasso Plattner Institute, University of Potsdam, Germany
sankalita.mandal@hpi.de

to subscribe to an event source or when to stop listening to an event stream [MWW17]. According to BPMN, when the control-flow reaches the event construct, the node is enabled and the process instance waits for the event occurrence. Once the event occurs, the control-flow is passed to downstream activities. The above semantics is a severe limitation as in a distributed setup, the event sources are mostly unaware of the process execution status and therefore the events can occur anytime irrespective of the process being ready to consume it.

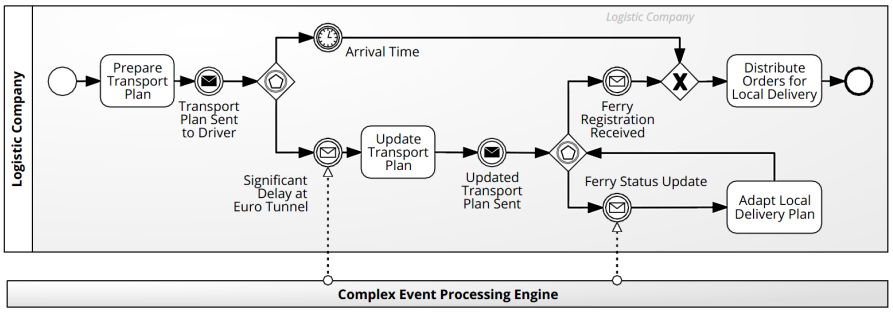

Fig. 1: Motivating example showing the need of flexible event subscription.

In Fig. 1, a process from logistics domain has been shown. The *Logistic Company* manages a process engine to control several transportation where trucks carry goods via Euro Tunnel following the transport plan sent by the company. However, when the truck is still driving (before `Arrival Time`), the engine gets notification if there is a `Significant Delay at Euro Tunnel`, e.g. caused by technical difficulty or traffic congestion. In this situation, the alternative route ferry is taken to cross the Strait of Dover. The ferry status update is considered periodically to adapt the local delivery plan at the destination and the orders are distributed once the ferry is taken. Now, Euro Tunnel or ferry do not know about the processes run by the logistic companies and publish the status update events whenever there is a change of situation which might not coincide with the time when the process is ready to receive the update. As a result, a process instance may not react to an event which occurred before its control-flow reached the respective event construct, even if the event is still relevant for process execution. The process execution might even get stuck waiting for an event which has already occurred. On the other hand, the process engine does not need to listen to an event stream if at certain point of process execution the event becomes irrelevant, e.g. Euro Tunnel updates become irrelevant after ferry is chosen.

So far, there is no clear semantics for all the possible scenarios supporting flexible event handling from a business process perspective. The work in this thesis proposal addresses these shortcomings of event handling semantics and aims to offer a flexible event handling model accommodating the possibilities of event-process interactions starting from the initiation of process engine, via process execution, to the engine termination. The next section discusses the steps followed so far as well as the future plans to achieve this.

2 Research Plan: Past, Current & Future Steps

The goal of the research is to build a flexible event handling model from the perspective of business process execution. The event handling semantics should be grounded formally and provide conceptual as well as technical analysis of using events in business processes. The rest of the section describes the steps required to reach the goal.

Integration of Real-World Events and Business Processes. The whole research is predominantly motivated considering the importance and advantages of using event information flexibly during process execution. There can be several applications (such as [Pu17]) of using event processing techniques that can improve process flexibility and efficiency. The first step was, therefore, to build an integrated architecture enabling event-process interaction. Though the current process engines have provisions for receiving events from external sources, there was no end-to-end framework considering the conceptual and technical challenges for integrating real-life events in processes. In our work associated with a project with industry partners [Be16], we explored the basic aspects to consider while integrating the two worlds of BPM and CEP [MHW17]. This framework identifies three requirements: 1) Separation of concerns between the logic of process execution and event processing, 2) Representation of event hierarchies to show the connection among event sources, low-level events generated from the sources and higher-level business events aggregated from them, and finally 3) Execution of event integration that deals with the technical challenges such as binding events, receiving events and reacting on events. Based on these requirements, a conceptual framework is provided. Also, an integrated architecture consisting of the process engine Chimera[2], the event processing platform Unicorn[3] and an extended version of Camunda process modeler[4] is implemented to realize the concepts.

Early Event Subscription and Event Buffering. The basic framework defines the concept and technicalities required to subscribe to an event, get notified when a matching event occurs and to react on the event according to process specification. However, in a distributed IT setup, the event sources might be unaware of the process execution status. Therefore, the BPMN assumption of event occurrence only when the process is ready to consume it seemed unrealistic and restricting. Hence, the notion of early event subscription using an explicit subscription task is introduced in later work [MWW17]. To accommodate early event occurrences, an event buffer is discussed in this context. The buffer comes with specific policies to store, retrieve and reuse the events.

Flexible Event Handling Model with Petri Net Mapping. The next step prescribes the formal semantics for handling events considering the fact *events can occur anytime – before, during or after the process is ready to consume it*. Based on the grounding semantics, we first

[2] https://bpt.hpi.uni-potsdam.de/Chimera
[3] https://bpt.hpi.uni-potsdam.de/UNICORN/WebHome
[4] http://bpmn.io/

analyze different possibilities of modeling an external event in a process flow, represented as intermediate catching message events. We assume that the subscription does not have any unresolved data dependency, i.e. the subscription is not dependent on any data that must be generated during process execution. The subscription, occurrence and consumption possibilities of events are considered based on a timeline starting from the initiation of process engine and ending at the engine shut down, as shown in Fig. 2.

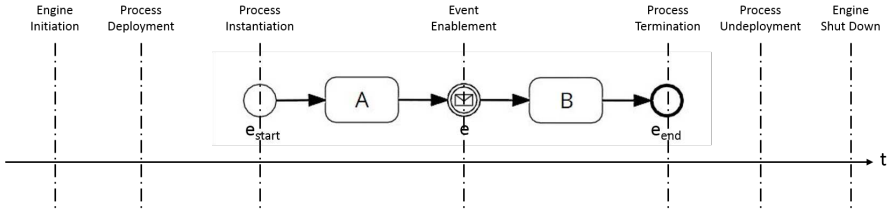

Fig. 2: Process execution timeline.

The CASU Framework proposed in [DM09] talks about (un)-subscription patterns with respect to process instantiation semantics whereas we focus on the environmental occurrences modeled as intermediate catching events. Fig. 3 is an extended version of the causality proposed by [BDG07] that said an event can only be consumed if there is a subscription for this event and the event has actually occurred. We proposed in [MWW17] that to make an event relevant for a process, the subscription should be done before the event occurrence. An unsubscription can be done only when a subscription exists, but it is independent of the event consumption, or even event occurrence, e.g. if at certain point of process execution, the event becomes irrelevant, unsubscription can be done. The dependencies show that along with maintaining the causality of process execution, the event subscription, occurrence, consumption and unsubscription should also follow certain temporal ordering. Currently we are working on mapping the concepts required for flexible event handling model to Petri Net modules in order to specify the implementation semantics and analyzing the traces to ensure correct execution.

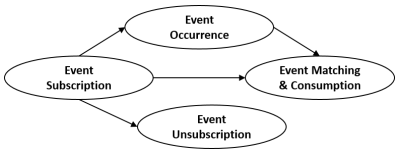

Fig. 3: Dependencies among event subscription, event occurrence, consumption, and unsubscription.

3 Conclusion

Event processing informs business processes about the environmental occurrences, such that the process can adapt to the changed environment as required. This improves process execution with respect to flexibility and efficiency. Hence, we presented a basic framework to realize the integration of event processing in business processes. Further, we developed the notion of early event subscription and event buffering to accommodate the need of flexible event-process communication in a distributed IT system. To this end, a formal event

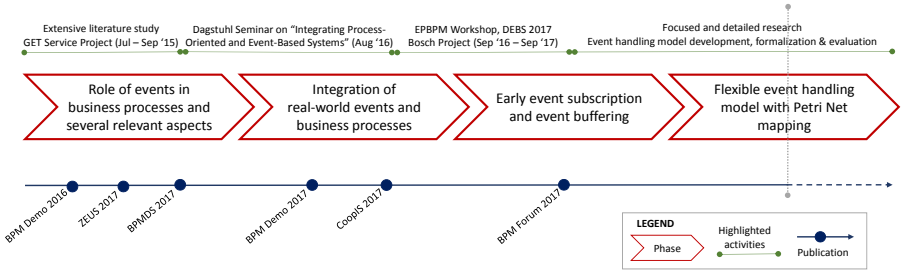

Fig. 4: Research progress and planned steps.

handling model is proposed that specifies the semantics for event subscription, occurrence, consumption and unsubscription taking into account the complete runtime of a process engine and several points in time when subscription for an intermediate event can be made. Next steps include correctness check of process specification in the light of the interactions with environmental events. Fig. 4 summarizes the research progress reflecting on the phases discussed, the associated publications for each phase and the activities that inspired further research directions.

References

[BDG07] Barros, A.; Decker, G.; Grosskopf, A.: Complex Events in Business Processes. In: Business Information Systems. Springer, 2007.

[Be16] Beyer, J.; Kuhn, P.; Hewelt, M.; Mandal, S.; Weske, M.: Unicorn meets Chimera: Integrating External Events into Case Management. In: BPM Demo Session. CEUR-WS.org, 2016.

[DM09] Decker, Gero; Mendling, Jan: Process instantiation. Data Knowledge Engineering, 68(9):777–792, 2009.

[EN10] Etzion, O.; Niblett, P.: Event Processing in Action. Manning Publications, 2010.

[HMW13] Herzberg, N.; Meyer, A.; Weske, M.: An Event Processing Platform for Business Process Management. In: EDOC. IEEE, 2013.

[MHW17] Mandal, S.; Hewelt, M.; Weske, M.: A Framework for Integrating Real-World Events and Processes in an IoT Environment. In: CoopIS. Springer, 2017.

[MWW17] Mandal, S.; Weidlich, M.; Weske, M.: Events in Business Process Implementation: Early Subscription and Event Buffering. In: BPM Forum. Springer, 2017.

[OM11] OMG: , Business Process Model and Notation (BPMN), Version 2.0. `http://www.omg.org/spec/BPMN/2.0/`, January 2011.

[Pu17] Pufahl, L.; Mandal, S.; Batoulis, K.; Weske, M.: Re-evaluation of Decisions based on Events. In: BPMDS. Springer, 2017.

[We12] Weske, M.: Business Process Management - Concepts, Languages, Architectures, 2nd Edition. Springer, 2012.