BPMN Extension for Business Process Monitoring[*]

Anne Baumgrass, Nico Herzberg, Andreas Meyer, Mathias Weske
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
{Anne.Baumgrass,Nico.Herzberg,Andreas.Meyer,
Mathias.Weske}@hpi.de

Abstract: The execution of business processes generates a lot of data representing happenings (also called events) that may be utilized for process monitoring and analysis. This, however, is not supported by typical Business Process Management Systems (BPMSs). Especially, in manual executing business process environments, i.e., not driven by a BPMS, the correlation of events to processes for monitoring and analysis is not trivial. At design-time, Process Event Monitoring Points are used in process models to specify the locations, where particular events are expected. Therewith, occurring events can be assigned to a process during run-time. In this paper, we introduce an extension to BPMN, which implements this connection between process models and events. We show applicability of this extension by applying it to a logistics scenario taken from an EU project.

1 Introduction

Nowadays, the availability and creation of process-relevant information in terms of events[1] increases substantially, e.g. through the Internet of Things, Big Data as well as new and faster in-memory and data streaming technologies. Furthermore, sensors such as GPS receivers, RFID chips, transponders, detectors, cameras, or satellites enable the depiction of the current status of processes. Although the amount of events would lead to a fine-grained monitoring, mining, and decision support for business processes, a large number of business processes controlled by a BPMS operate without them. Especially, monitoring events of business processes not controlled by a single system and across enterprise boundaries is valuable for gaining insights about business process execution [Luc02, Luc11], e.g., to ensure a business process is executed as expected. However, enabling business process monitoring through events is not trivial [RLM$^+$12, HMW13, DGD12].

Central to Business Process Management (BPM) [Wes12] are process models as they specify business processes and, therefore, capture the operations that need to be carried out to achieve a business goal. Therewith, process models are used, among others, for documentation, verification, enactment, process monitoring, and process analysis. Thus,

[*]The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service).

[1]In this paper, we refer to events as real-world happenings (and not as the modeling elements in BPMN).

the event information of different sensors and produced by novel technologies need to be aggregated over these process executions to see the overall performance of a defined business process, e.g., the consumed resources, the time spend for the operations, and the arisen cost of the business process execution. Especially for process analysis but also for monitoring, it is valuable to have the process context, i.e., the process model, connected to the business process execution data – the events.

In [BBH$^+$13a], we introduced an approach to allow model-driven business process monitoring by automatically creating Complex Event Processing (CEP) queries from appropriately annotated process models. In this paper, we present the corresponding extension to the Business Process Model and Notation (BPMN) metamodel, which formally describes the association of events to process models for business process monitoring and analysis. Associating event information to business processes is supported by the concept of Process Event Monitoring Points (PEMPs) [HMW13] that specify where and when which event is expected during business process execution. We extend BPMN to integrate PEMPs into process models verified by a prototypical implementation of the connection of PEMPs and process models.

The remainder of this paper is structured as follows. Section 2 motivates and demonstrates our BPMN extensions with a scenario from the logistics domain. We elaborate on event modeling in business process models in general in Section 3, before we present the BPMN extension with process event monitoring points to establish process monitoring and analysis in Section 4. Afterwards, an application and implementation of the BPMN extension is shown in Section 5. In Section 6, major related literature is discussed before we conclude this paper in Section 7.

2 Motivating Scenario

Especially for logistics, monitoring (resp. tracking and tracing) the complete logistics process from client to customer is essential to locate the shipment, reveal the progress in delivery, and to inform other stakeholders as well as the customer about arrivals and departures. Furthermore, monitoring a logistics process allows to replan and avoid penalties in case of deviations. However, the ongoing globalization including an internationalization of production and distribution [SP09] leads to more and more complex logistics processes that get harder to monitor because of the increasing amount of transportations and transshipments between countries, as well as the increasing amount of often distributed and loosely coupled systems to support these. Below, the example of a container pick-up handling is used to describe our approach as well as show the necessity of a BPMN extension enabling the connection of processes and event information.

Figure 1 shows a business process model for the container pick-up handling at a terminal modeled in BPMN. After receiving the container location, a driver needs to drive the corresponding truck to the pick-up location (activity `Drive to container location`) Afterwards, a container is assigned to the driver, which then needs to be checked for appropriateness (sufficient capacity, special capabilities as cooling support if required) and

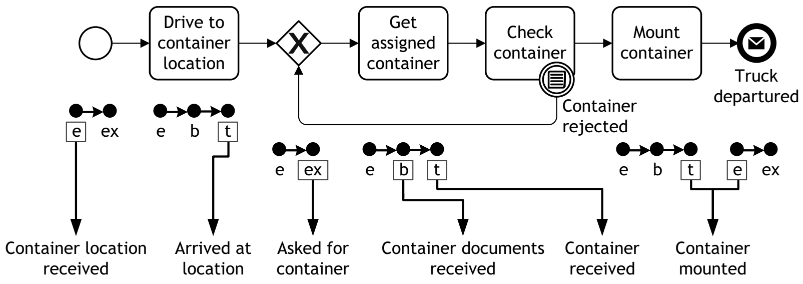


Figure 1: Business process model of a container pick-up process modeled in BPMN with a connection to event information via PEMPs associated with state transitions ((e)nable, (b)egin, (t)erminate, and (ex)ecute) occurring during node execution (see Section 3 for terminology).

mint condition (including tidiness and integrity). While the appropriateness checks are rather guaranteed due to the booking in advance of containers, the mint condition checks are critical and fail sometimes. If any of the checks fails, the activity `Check container` gets canceled and the corresponding intermediate event *Container rejected* is raised. This triggers another container assignment to the driver followed by another check. If all checks succeed, the driver mounts the container to the truck and departs from the pick-up location

An unobstructed process of the container handling is very important for the logistics company, as this is one of their core business processes. Any delays while picking up the container will affect schedules, contracts with customers, and partners as well as resources, i.e., drivers and truck availability. Therefore, it is necessary to monitor and analyze the process as detailed as possible. This allows for the measurement of activity execution times, delays, and resource consumption for instance. Establishing basic monitoring capabilities is possible in BPMN, however, the definition of process monitoring and analysis in a fine-grained and by the industry required manner is not possible. Establishing valuable process monitoring and analysis requires the capturing of events at certain locations within a process and the correlation of them to the process context specified by process models.

In this paper, we present a BPMN extension that implements the framework proposed by Herzberg et al. [HMW13] connecting event and process information via PEMPs associated to transitions describing state changes of nodes occurring during execution of a node For the example, we stick to simplified node life cycles. An activity may be in states enabled, running, or terminated with the corresponding transitions *(e)nable*, *(b)egin*, and *(t)erminate* while for gateways and BPMN events only transitions *(e)nable* and *(ex)ecute* may be triggered; for details about terminology, see Section 3. Potentially, each of these transitions may be connected to a PEMP [HKRS12] used for process observation. However, this does not mean that each transition is observable and therefore monitored during process execution.

In practice, the start of the process may be observed, i.e., the receipt of the location details can be tracked. Therefore, an event called 'Container location received' is associated to the transition *(e)nable* of the start event in the process model in Figure 1. For activity `Drive to container location`, the model associates Global Positioning System (GPS) coordinates aggregated to identify when the truck arrived at its destination, i.e., modelling when an activity termination will take place. In contrast, we model activity `Check container` as not observable, because it is a manual task without any IT system interaction. Likewise, the intermediate conditional event cannot be captured, because it represents a manual interaction between the driver and the pick-up location worker. However, indirectly, the happening of both may be derived whenever another container is requested. Altogether, at design-time, we model six different events as observable and correlate them to the correct locations in the process model and utilize this information for subsequent process analysis during process execution. If the PEMPs would have not been specified, events still occur, but correlation to the process (model) becomes a time-consuming and probably manual task depending on the information existent within the events.

3 Modeling Events in Business Processes Models

In this section, we formally introduce the concepts for Complex Event Processing (CEP) in the context of BPM and correlate both worlds. Figure 2 provides an overview about all concepts and their relations. The Process Event Monitoring Point (PEMP) is the connection between process models and events by relating events to transitions of the node life cycle defined for each node of a process model. The structure of a process model aligns with Definition 4 discussed below.

An event is a real-world happening occurring in a particular point in time at a certain place in a certain context [Luc02]. Processing an event requires to first store it in an information system. Thereby, the event gets transformed into an *event object* which structure is defined by its corresponding *event object type* . In accordance with [BBH$^+$13a] and [HMW13], we define both concepts in the context of BPM as follows.

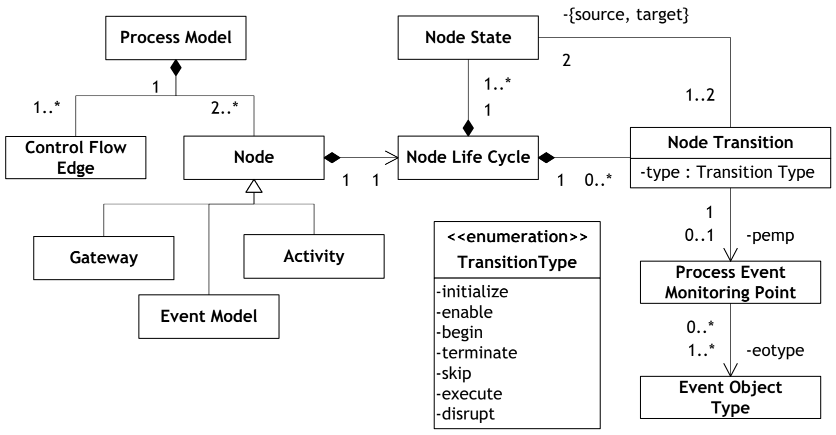


Figure 2: Model of relations for process models and events.

Definition 1 (Event object type) An *event object type* $\mathcal{ET} = (name, cd)$ refers to a unique *name* indicating the object type identifier and has a *content description cd* that specifies the structure of a particular event being of this event type. $\diamond$

Definition 2 (Event object) An *event object* $\mathcal{E} = (\mathcal{ET}, id, P, timestamp, \mathcal{C})$ refers to an event object type $\mathcal{ET}$, has a unique identifier *id*, refers to a set P of *process instances* being affected by the event object, has a *timestamp* indicating the occurrence time, and contains an additional *event content* $\mathcal{C}$. $\diamond$

Definition 3 (Event content and event content description) A *content description* $cd = \{(attributename, datatype)\}$ of an event object type $\mathcal{ET}$ is a collection of key-value-pairs specifying the available attributes by name *attributename* and by the data type *datatype* in which the corresponding attribute may be given. The datatype may be a complex, i.e., allowing hierarchical structures of attributes. For example, the coordinate which is identified by its latitude and longitude. Consequently, the *event content* $\mathcal{C} = \{(attributename, value)\}$ of an event object $\mathcal{E}$ is also a collection of key-value-pairs containing the specific *values* for the *attributename* that references the attribute specified in the corresponding event object type. $\diamond$

Event objects exist on different levels depending on the information provided as described in [HMW13]. We refer to an event object containing at least one reference to a process instance as *process event*; the corresponding structure definition is named *process event type*. The execution of business processes is described by *process instances* which follow instructions specified in *process models*. As indicated above, events affect the execution of business processes in multiple ways. They may, for instance, trigger the execution of specific activities or indicate a specific state of the business process meaning that some activity has been successfully or unsuccessfully executed.

Definition 4 (Process model) A *process model* $M = (N, F, \psi)$ consists of a finite non-empty set $N \subseteq A \cup E \cup G$ of nodes being *activities* A, *event models* E, and *gateways* G $F \subseteq N \times N$ represents the *control flow relation* over nodes. Function $\psi : G \rightarrow \{xor, and\}$ assigns a type to each gateway. $\diamond$

During process execution, each node follows its own *node life cycle* that specifies the states the node may be in and the transitions between pairs of states. The control flow relation and these node states influence each other such that the control flow triggers state transitions and some state transitions trigger the control flow. In the context of this paper, the life cycle L_A for activities consisting of states *initialized*, *ready*, *running*, *terminated*, *disrupted* and *skipped* as well as the initial state visualized by a blank circle. These node states are connected by transitions *initialize*, *enable*, *begin*, *terminate*, *disrupt*, *skip*, and *(ex)ecute* For events and gateways, we utilize a subset of these states removing states *running* and *disrupted* and the transitions leading to them. Upon process instantiation, i.e., start of the process instance, all nodes transition into state *initialized*. If the control flow reaches a node, it transitions into state *ready*. Completing the execution of a node (reaching the state *disrupted*, *terminated*, or *skipped*) triggers the control flow. Formally, we define the underlying state transition net as follows.

Definition 5 (Node life cycle) A *node life cycle* $L = (S, T, \varphi)$ consists of a finite non-empty set S of node states and a finite set $T \subseteq S \times S$ of node state transitions. Let $\mathcal{L}$ be the set of all node life cycles defined for the nodes N of process model M. Then, there exists a function $\varphi : N \rightarrow \mathcal{L}$ assigning a node life cycle to each node $n \in N$ of M. $\diamond$

Process monitoring deals with capturing events based on node execution. We define process event monitoring points (PEMPs) [HKRS12] to specify on model level where we expect events to occur. Assigning PEMPs on node level is possible but results in issues identifying whether the start, end, or some other happening during node execution shall be captured Assume, there exist waiting times between the enablement and the start of an activity Then, the enablement of the activity does not indicate the immediate start (state *running*) Capturing the time passed between activity enablement and activity begin is only possible, if both state transitions are captured. Therefore, we assign each PEMP to one specific state transition in the life cycle of the corresponding node. We distinguish node state transitions into the ones observable by occurring events and the ones requiring the context of the process instance to deduce their triggering. For a given node of a process model, each state transition belongs to either group. As utilizing PEMPs for process monitoring is independent from the process instance execution, a PEMP can only be attached to directly observable node state transitions. A PEMP is defined as follows.

Definition 6 (Process event monitoring point) Let M be a process model, L a node life cycle, and $O_L \subseteq T_L$ the set of state transitions not requiring process instance information Then, a *process event monitoring point* is a tuple $PEMP = (M, n, t, et)$, where M is the process model it is contained in, $n \in N$ is the node of the process model it is created for, $t \in O_L$ is a state transition within the node life cycle L it is created for, and $et \in \mathcal{ET}$ is an event object type specifying the event object to be recognized. $\diamond$

Recapitulating the concepts introduced in this section, there exist two direct connections between CEP and BPM. At design-time, PEMPs specify where in the process model which event is supposed to happen by linking a node state transition and a process event type. At run-time, each event object is or can get related to a set of process instances by utilizing the framework described in [HMW13].

4 Extending BPMN with Process Event Monitoring Points

Typically, business process execution is monitored by BPMSs on the basis of a process model. In distributed environments, however, parts of this execution can only be monitored via CEP techniques. To combine BPM and CEP, process models must include the node life cycles and the definition of PEMPs (see Section 3).

A widely accepted standard for process modeling is BPMN. Although it does not provide native language elements to model node life cycles and PEMPs (see Definitions 5 and 6), it benefits from its organizational maintenance through the Object Management

Group (OMG), builds upon the standardized metamodel Meta Object Facility (MOF)[2], and provides its own extension mechanism [OMG11]. Thus, using metamodel extensions, we are able to transform process models to platform-specific models and code.

In this paper, we provide a BPMN extension definition for modeling the life cycle and PEMPs as defined in Section 3 for each node in a BPMN model. This extension can, for example, be used to transform a process model to CEP-specific code for event detection and, thus, enable the monitoring of events in distributed systems that are associated with a single business process (cf. [BBH+13a]).

In particular, we extend the BPMN metamodel based on MOF to define new domain-specific language concepts used to serialize BPMN models in the XML Metadata Interchange (XMI) specification without contradicting the semantics of any BPMN element. This BPMN metamodel extension implements node life cycles and PEMPs in BPMN models. Figure 3 illustrates our extension (shown in white) to the existing BPMN metamodel elements (shown in gray). We associate a life cycle with a *FlowNode* of the BPMN specification This life cycle includes states and transitions that can be associated with a PEMP that is defined by an event object type (see Section 3).

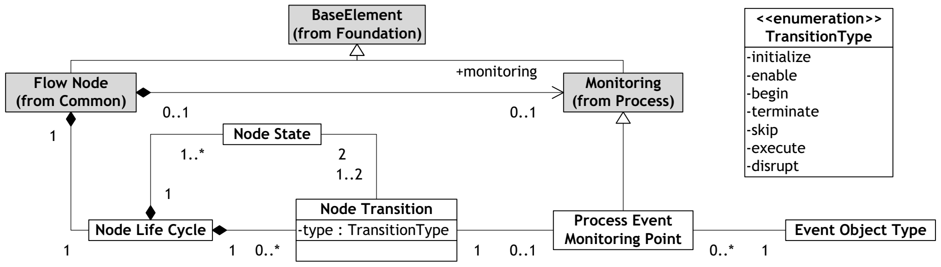


Figure 3: BPMN metamodel extension for including PEMPs and life cycles in BPMN models.

Complementary, in a BPMN specification, new attributes and elements have to be specified in an *ExtensionDefinition* that can be bound to a model definition by an *Extension* to any *BaseElement* (see [OMG11]). As this paper concentrates on the general concept of automating the derivation of CEP queries from business models, we use BPMN and its own eXtensible Markup Language (XML) format for demonstration purposes. This is also because the transformation from BPMN's own extension mechanism to a direct extension of the BPMN metamodel and MOF's own interchange format called XMI can be found in [SCV11b].

For the representation of BPMN models, BPMN defines a set of XML Schema documents specifying the interchange format for BPMN models. Thus, to model and exchange BPMN models that include nodes' state transitions and PEMPs, we derive an own XML Schema for our extension (see Listing 1). In particular, we developed a BPMN+X model [SCV11b] to specify our extension as shown in Figure 4. It allows the attachment of a transition element to a *FlowNode*. This transition element references a PEMP and defines its type by enumeration (see Figure 5). In this way, any state transition of an activity, gateway, or event

[2]http://www.omg.org/mof/

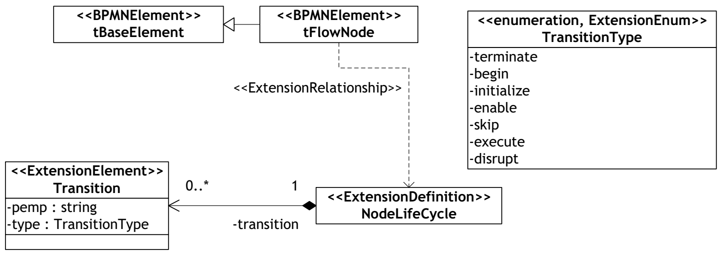

<<BPMNElement>> tBaseElement
<<BPMNElement>> tFlowNode
<<enumeration, ExtensionEnum>> TransitionType
-terminate
-begin
-initialize
-enable
-skip
-execute
-disrupt
<<ExtensionRelationship>>
<<ExtensionElement>> Transition
-pemp : string
-type : TransitionType
0..*
1
-transition
<<ExtensionDefinition>> NodeLifeCycle

Figure 4: BPMN+X model for including PEMPs and life cycles in BPMN models.

models including a PEMP that references an event type can be represented in a BPMN model.

Applying the approach described in [SCV11b] to our model shown in Figure 4 generates the XML Schema that is conform with BPMN's extension definition shown in Listing 1 The node life cycle is an *ExtensionDefinition* introduced as `<xsd:group>` element that includes a transition (*ExtensionAttributeDefinition*) as `<xsd:element>` (Listing 1 Lines 6-10). The type of a transition is externally defined by an `<xsd:complexType>` element (Listing 1 Lines 12-15). Finally, we added the literals for node's transitions as enumeration (Listing 1 Lines 16-21) that can be made available for the type attribute definition in the transition itself. Although possible, we did not explicitly specify the event types that can be bound to a transition as we assume those are defined differently in each CEP environment However, we allow referencing the event type via its identifier (e.g., its name) for a binding of event types in CEP implementation (see Section 5).

Listing 1: XML Schema for the definition of PEMPs and life cycles in BPMN models.

```
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <xsd:schema
 3   xmlns:cep="http://www.extensions.bpmn/cep/deriviation"
 4   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 5   targetNamespace="http://www.extensions.bpmn/cep/deriviation">
 6   <xsd:group name="NodeLifeCycle">
 7    <xsd:sequence>
 8     <xsd:element ref="transition" maxOccurs="unbounded" minOccurs="0"/>
 9    </xsd:sequence>
10   </xsd:group>
11   <xsd:element name="transition" type="tTransition"/>
12   <xsd:complexType name="tTransition">
13     <xsd:attribute name="pemp" type="xsd:string" use="required"/>
14     <xsd:attribute name="type" type="TransitionType" use="required"/>
15   </xsd:complexType>
16   <xsd:simpleType name="TransitionType">
17    <xsd:restriction base="xsd:string">
18     <xsd:enumeration value="initialize"/>
19     ...
20    </xsd:restriction>
21   </xsd:simpleType>
22  </xsd:schema>
```

Finally, using our XML schema definition we can define BPMN-conform models in XML format. The transitions, we defined above via our schema, are added as elements in a

BPMN-element through its `<extensionElements>`. For instance, Listing 2 shows an example of a BPMN task and its two transitions *begin* and *terminate* for which the PEMP requires events from the event types `ContainerDocsReceived` and `ContainerReceived` respectively to monitor the state transitions of this task. These event types must be available in the CEP system of usage. An example derivation of a query is shown in the next section

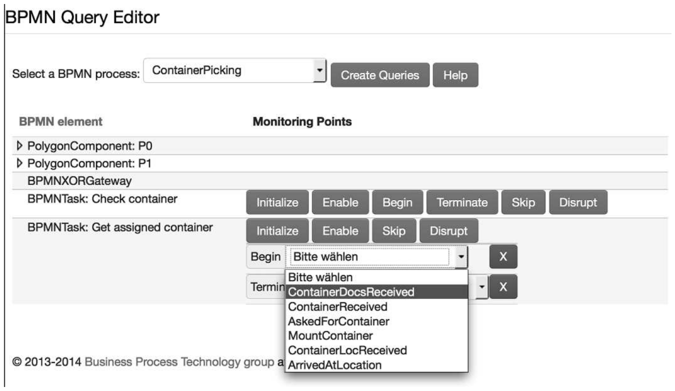
Listing 2: Definition of a PEMP for a BPMN activity with XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:cep="http://www.extensions.bpmn/cep/deriviation"
  xsi:schemaLocation="http://www.extensions.bpmn/cep/deriviation CEPDerivation.
      xsd"
  targetNamespace="">
  <import importType="http://www.w3.org/2001/XMLSchema" location="CEPDerivation.
      xsd"
    namespace="http://www.extensions.bpmn/cep/deriviation" />
  <extension mustUnderstand="true" definition="cep:NodeLifeCycle" />
  <process id="Process_1">
    <task id="Task_2" name="Get assigned container">
      <extensionElements>
        <cep:transition pemp="ContainerDocsReceived" type="begin" />
        <cep:transition pemp="ContainerReceived" type="terminate" />
      </extensionElements>
      ...
    </task>
    ...
</definitions>
```

5 Experimental Evaluation

While the scenario in Section 2 introduced the benefits of the modeling extension at design time, this section discusses how we use the extension during process execution of the scenario for monitoring the current progress of its process instances. In particular, we implemented a service that is able to import the BPMN models specified in Section 4. The service decomposes the BPMN models and generates interleaving CEP queries with which we are able to monitor business process executions. Furthermore, this service may be used to detect execution violations related to a process model specification. This service is directly implemented in the Event Processing Platform (EPP) [HMW13, BBH+13b][3]. In general, the EPP provides services to capture real-world events from different sources, to process these events, e.g., by aggregation or transformation, to correlate the events to its corresponding business process, and to provide and to manage these events for event consumers, e.g., business process monitoring applications. Especially for the event correlation to the right business process and the distribution of the events to the responsible

[3]The whole project is currently under development. Downloads, tutorials, and further information of the stable version can be found at: `http://bpt.hpi.uni-potsdam.de/Public/EPP`. To see the preliminary application of this paper, we refer the interested reader to our screencast available at `https://www.youtube.com/watch?v=doAFKwIEp6w`, starting from minute 7:37.


BPMN Query Editor

Select a BPMN process: ContainerPicking

BPMN element	Monitoring Points
▸ PolygonComponent: P0	
▸ PolygonComponent: P1	
BPMNXORGateway	
BPMNTask: Check container	Initialize, Enable, Begin, Terminate, Skip, Disrupt
BPMNTask: Get assigned container	Initialize, Enable, Skip, Disrupt
	Begin: Bitte wählen
	Termin…

© 2013-2014 Business Process Technology group a…

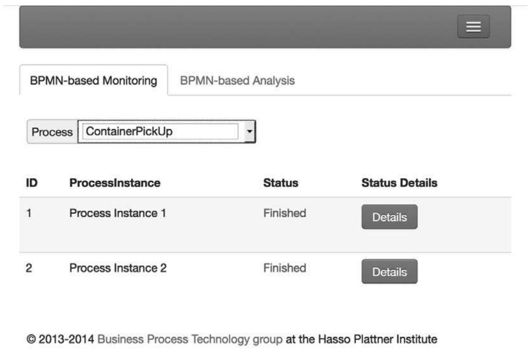
Figure 5: EPP implementation showing the manipulation opportunities of PEMPs in a BPMN process

party of this business process, the platform not only stores event object types but also business process models and event queries.

A series of actions is necessary to use our extension for event processing and business process monitoring in the EPP. First, the user has to define the available event object types by name and its content description (see Definition 1). These event object types are determined by the real-world event sources that the EPP is connected to. The EPP is able to upload event object type descriptions defined as XML Schema. Second, event types must be correlated to group event objects to process instances. This is usually done by using one or more attributes defined in the content description of an event object type. In the introduced example in Section 2, this grouping to process instances could be based on the driver's identification and the truck she drives. It is also possible to define such a group based on the booking number of the transportation order that each event object references. Subsequently, the BPMN model must be created. In this BPMN model, each BPMN element that we want to monitor should include the definition of its state transitions including the event object types referenced in its PEMP as specified in Section 4.

The EPP enables the import of business process models specified in the BPMN-conform XML format including PEMPs. Furthermore, the user gets the possibility to change the association of event types to state transitions of each imported BPMN model directly in the user interface of the EPP as shown in Figure 5. Thus, we are able to upload arbitrary BPMN-specific models and even associate event objects types that are stored in the EPP to PEMPs in a specific BPMN model. Next, the uploaded models can be used to derive CEP queries. Queries in the EPP are implemented using the Event Pattern Language (EPL) provided by Esper [BV07]. The EPP registers each CEP query in Esper via listeners. These listeners get informed if the query matches observed events with the specified conditions defined by the query. For instance, Listing 3 shows an example query in Esper query language for our scenario process model in Section 2; in total, four queries could be derived

As not every activity is observable, the derived queries are restricted to those events that are observable. Listing 3 shows the *St1* query that monitors the sequence of two monitoring



ID	ProcessInstance	Status	Status Details
1	Process Instance 1	Finished	Details
2	Process Instance 2	Finished	Details

© 2013-2014 Business Process Technology group at the Hasso Plattner Institute

Figure 6: EPP implementation showing the monitoring of process instances.

points with the event types of the `Get assigned container` activity.

While the definition of the sequential ordering of the events for this query is enclosed in `PATTERN[...]` in the `FROM`-clause, the `WHERE` clause checks whether the events from both event types have occurred in the same process instance. In particular, this query is activated for each *ContainerDocsReceived* event object imported in the EPP that is followed by a *ContainerReceived* event object belonging to the same process instance. Similar queries are generated for all other PEMPs of the example (cf. [BBH+13a]). Altogether, we are now able to monitor all instances of the process and their status as shown in Figure 6.

Listing 3: Example of a monitoring query using Esper.

```
St1:
SELECT *
FROM PATTERN [((EVERY S0=ContainerDocsReceived -> EVERY S1=ContainerReceived)]
WHERE SetUtils.isIntersectionNotEmpty({S0.ProcessInstances, S1.ProcessInstances})
```

6 Related Work

With the proposed BPMN extension, the basis for high quality process monitoring and analysis is established. Besides process monitoring and analysis, prediction, control, and optimization are further major disciplines of Business Process Intelligence (BPI) [MR06, GCC+04]. [MR06] describes a reference architecture for BPI containing an integration, a functional, and a visualization layer. However, business process models are not integrated into that architecture. We argue that correlating events and the process monitoring and analysis results need to be done in the context of the process, which is described with process models. Process mining [vdA12] discovers this in the same direction by profiling process models from event logs using process discovery techniques [vdAea12] and enriching that models by execution information. Process Mining is utilizing information that is available in a well-defined event log, while our BPMN extension allows the definition of PEMPs that describe which event respectively information is required at which point in the process for process monitoring and analysis purposes.

Another area of BPI focuses on Business Activity Monitoring (BAM). [DWC11] presents an overview of BAM and gives a four class-categorization of BAM systems: pure BAM, discovery-oriented BAM, simulation-oriented BAM, and reporting-oriented BAM. All of these classes benefit from the BPMN extension as the resulting process events will have high information content and BAM techniques and methods can be applied to these process events to provide valuable monitoring results.

Del-Río-Ortega et al. [dRORRC10] present the concept of Process Performance Indicators (PPI), the process related form of key performance indicators, to enable process evaluation They propose a metamodel called PPINOT that supports the graphical notation of PPIs with process models, e.g., modeled with BPMN [dRORCRC13].

Furthermore, we focused on the automation of BPI with the help of CEP. [BPG12] introduces techniques to automatically generate Esper queries by taking a choreography model as a formalization of the process. In [WZM+11], BPMN models are taken as a basis to create EPL statements to monitor process violations but not to monitor the process execution in general as it is our goal. Barros et al. [BDG07] present a set of patterns describing relations and dependencies of events in business processes that have to be captured in process models to observe the overall process context. Their assessment of the modeling languages BPMN and Business Process Execution Language (BPEL) resulted in their language proposal called Business Event Modeling Notation (BEMN) [DGB07], a graphical language for modeling composite events in business processes. BEMN allows to define event rules, e.g., specific combinations of events, that are to be used in stand-alone diagrams or as integration into BPMN. In the CEP context, Rozsnyai et al. [RLM+12] introduce an approach to monitor the execution of semi-structured processes. Our extension of the metamodel complements these approaches by providing an interchange format that may be used by various systems to combine BPM and CEP.

Several approaches have extended BPMN to represent their domain specific requirements [RFMP07, ZSL11, SCV11a]. For example, Rodríguez et al. [RFMP07] present a metamodel extension of the diagram specification of BPMN 1.0 in order to model security requirements, while Stroppi et al. [SCV11a] specify resources in business processes using BPMN's own extension mechanisms. In addition, Stroppi et al. [SCV11b] presented a generic approach to transform metamodel extensions into BPMN's own interchange format Furthermore, Zor, Schumm, and Leymann [ZSL11] graphically extend BPMN models for the manufacturing domain. Complementary, our approach used the tools presented in [SCV11b] and extended the BPMN metamodel to consider CEP concepts.

7 Conclusion

In this paper, we introduced a BPMN extension to facilitate the concept of Process Event Monitoring Points (PEMPs) within process models. PEMPs are utilized to specify at which place in the process model a specific event is expected. Applying this concept, process monitoring and analysis based on events occurring during business process execution is enabled. This is important, especially in manual executing business process environments, as the event information is usually not directly correlated to the business process execution In essence, we used PEMPs to relate the domains of BPM and CEP. Based on the proposed BPMN extension, we modeled and implemented a scenario taken from the logistics domain and have shown the applicability and value of the approach.

In future work, we will investigate how PEMPs can be graphically represented in BPMN models and how this can be supported with appropriate implementations. Furthermore, we will apply the BPMN extension to process monitoring and analysis tasks in different domains, e.g., to measure run-time or process cost analysis. Further, we will adapt the approach to analyse process event occurrences.

References

[BBH+13a] M. Backmann, A. Baumgrass, N. Herzberg, A. Meyer, and M. Weske. Model-driven Event Query Generation for Business Process Monitoring. In *International Conference on Service Oriented Computing Workshops*. Springer, 2013.

[BBH+13b] S. Bülow, M. Backmann, N. Herzberg, T. Hille, A. Meyer, B. Ulm, T. Y. Wong, and M. Weske. Monitoring of Business Processes with Complex Event Processing. In *Business Process Management Workshops*. Springer, 2013.

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

[BPG12] A. Baouab, O. Perrin, and C. Godart. An Optimized Derivation of Event Queries to Monitor Choreography Violations. In *International Conference on Service Oriented Computing (ICSOC)*. Springer, 2012.

[BV07] T. Bernhardt and A. Vasseur. Esper: Event Stream Processing and Correlation. Published at `http://onjava.com/` by O'Reilly Media, 2007.

[DGB07] G. Decker, A. Grosskopf, and A. Barros. A Graphical Notation for Modeling Complex Events in Business Processes. In *Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2007.

[DGD12] M. Daum, M. Götz, and J. Domaschka. Integrating CEP and BPM: How CEP realizes Functional Requirements of BPM Applications (Industry Article). In *Distributed Event-Based Systems (DEBS)*, 2012.

[dRORCRC13] A. del Río-Ortega, M. Resinas, C. Cabanillas, and A. Ruiz-Cortés. On the Definition and Design-time Analysis of Process Performance Indicators. *Information Systems*, 38(4), 2013.

[dRORRC10] A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Defining Process Performance Indicators: An Ontological Approach. In *On the Move (OTM)*. Springer, 2010.

[DWC11] A. Dahanayake, R.J. Welke, and G. Cavalheiro. Improving the Understanding of BAM Technology for Real-time Decision Support. *International Journal of Business Information Systems (IJBIS)*, 7(1), December 2011.

[GCC+04] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M. Shan. Business Process Intelligence. *Computers in Industry*, 53(3), April 2004.

[HKRS12] N. Herzberg, M. Kunze, and A. Rogge-Solti. Towards Process Evaluation in Non-automated Process Execution Environments. In *Services and their Composition (ZEUS)*, 2012.

[HMW13] N. Herzberg, A. Meyer, and M. Weske. An Event Processing Platform for Business Process Management. In *Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2013.

[Luc02] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2002.

[Luc11] D. C. Luckham. *Event Processing for Business: Organizing the Real-time Enterprise* John Wiley & Sons, 2011.

[MR06] B. Mutschler and M. Reichert. Aktuelles Schlagwort: Business Process Intelligence *EMISA Forum*, 26(1), January 2006.

[OMG11] OMG. Business Process Model and Notation (BPMN), Version 2.0, 2011.

[RFMP07] A. Rodríguez, E. Fernández-Medina, and M. Piattini. A BPMN Extension for the Modeling of Security Requirements in Business Processes. *IEICE transactions on information and systems*, 90(4), 2007.

[RLM+12] S. Rozsnyai, G. T. Lakshmanan, V. Muthusamy, R. Khalaf, and M. J. Duftler Business Process Insight: An Approach and Platform for the Discovery and Analysis of End-to-End Business Processes. In *SRII Global Conference (SRII)*. IEEE, 2012.

[SCV11a] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. A BPMN 2.0 Extension to Define the Resource Perspective of Business Process Models. In *XIV Iberoamerican Conference on Software Engineering*, 2011.

[SCV11b] L. J. R. Stroppi, O. Chiotti, and P. D. Villarreal. Extending BPMN 2.0: Method and Tool Support. In *Business Process Model and Notation*, volume 95. Springer, 2011

[SP09] Frank Straube and Hans-Christian Pfohl. *Trends and Strategies in Logistics - Global Networks in an Era of Change: Environment, Security, Internationalisation, People* DVV Media Group, Dt. Verkehrs-Verlag, 2009.

[vdA12] W. M. P. van der Aalst. Process Mining: Overview and Opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 3(2), 2012.

[vdAea12] W. M. P. van der Aalst and et al. Process Mining Manifesto. In *Business Process Management Workshops*. Springer, 2012.

[Wes12] M. Weske. *Business Process Management: Concepts, Languages, Architectures Second Edition*. Springer, 2012.

[WZM+11] M. Weidlich, H. Ziekow, J. Mendling, O. Günther, M. Weske, and N. Desai. Event-Based Monitoring of Process Execution Violations. In *Business Process Management*. Springer, 2011.

[ZSL11] S. Zor, D. Schumm, and F. Leymann. A Proposal of BPMN Extensions for the Manufacturing Domain. In *Proceedings of the 44 th CIRP International Conference on Manufacturing Systems*, 2011.