From Event Streams to Process Models and Back: Challenges and Opportunities[☆]

Pnina Soffer[a], Annika Hinze[b,*], Agnes Koschmider[c], Holger Ziekow[d], Claudio Di Ciccio[e], Boris Koldehofe[f], Oliver Kopp[g], Arno Jacobsen[h], Jan Sürmeli[i], Wei Song[j]

[a] *University of Haifa, Israel*
[b] *University of Waikato, New Zealand*
[c] *Karlsruhe Institute of Technology, Germany*
[d] *Hochschule Furtwangen, Germany*
[e] *Vienna University of Economics and Business, Austria*
[f] *Technische Universität Darmstadt, Germany*
[g] *University of Stuttgart, Germany*
[h] *Technical University of Munich, Germany*
[i] *Humboldt-Universität zu Berlin, Germany*
[j] *Nanjing University of Science and Technology, China*

Abstract

The domains of complex event processing (CEP) and business process management (BPM) have different origins but for many aspects draw on similar concepts. While specific combinations of BPM and CEP have attracted research attention, resulting in solutions to specific problems, we attempt to take a broad view at the opportunities and challenges involved. We first illustrate these by a detailed example from the logistics domain. We then propose a mapping of this area into four quadrants – two quadrants drawing from CEP to create or extend process models and two quadrants starting

[☆] This paper is an outcome of discussions and collaborations that were initiated at the Dagstuhl seminar 16341 on "Integrating Process-Oriented and Event-Based Systems"

[*] Corresponding author

Email addresses: spnina@is.haifa.ac.il (Pnina Soffer), hinze@waikato.ac.nz (Annika Hinze), agnes.koschmider@kit.edu (Agnes Koschmider), zie@hs-furtwangen.de (Holger Ziekow), claudio.di.ciccio@wu.ac.at (Claudio Di Ciccio), boris.koldehofe@kom.tu-darmstadt.de (Boris Koldehofe), kopp@ipvs.uni-stuttgart.de (Oliver Kopp), arno.jacobsen@msrg.org (Arno Jacobsen), suermeli@hu-berlin.de (Jan Sürmeli), wsong@njust.edu.cn (Wei Song)

from a process model to address how it can guide CEP. Existing literature is reviewed and specific challenges and opportunities are indicated for each of these quadrants. Based on this mapping, we identify challenges and opportunities that recur across quadrants and can be considered as the core issues of this combination. We suggest that addressing these issues in a generic manner would form a sound basis for future applications and advance this area significantly.

Keywords: complex event processing, event-based systems, business processes, event-driven business process management

---

1. Introduction

Complex event processing (CEP) and Business Process Management (BPM) have traditionally been focusing on relatively distinct application areas. In recent years, newer application scenarios increasingly involve aspects of both areas. Particularly, in the context of Internet of Things (IoT) [1] the combination of both areas would be of great benefit and would allow implementing novel scenarios in smart homes [2], smart cities [3], connected cars [4], or logistics [5]. BPM and CEP have their origins in the relative isolation in which their concepts and formalisms were developed. Specifically, BPM concerns the level of activities within a process (e.g., to ship goods, to allocate goods) and intends to bring them in an order that follows a (business) process model. In contrast, CEP is often concerned with lower-level input events that may relate to activities (e.g., the location of a good via reading RFID tags at a certain reader). As a consequence, the logic of BPM and CEP is often defined at different abstraction levels and no explicit link is provided to bind the two approaches. Furthermore, current process modeling languages do not support the expressiveness of complex event processing systems and thus concepts of both domains are specified in two separate computational environments. To provide mutual benefits for a combination of BPM and CEP, several challenges need to be attended based on a careful and sound alignment.

Consider a scenario in the logistics domain. Relevant activities are to store goods, to manage transportation, to plan the distribution of workloads, and to handle (order and shipment) documents. Addressing this scenario from a BPM perspective would involve arranging and coordinating the activities through a (graphic or formal) representation in a way that meets customer's

needs and enabling execution by a logistics management system. Well-known examples of languages providing a graphical notation are Business Process Model and Notation (BPMN) [6], Event-driven Process Chains (EPC) [7], and Petri nets [8]. The execution is typically logged by the BPM system. The generated event log can later on be mined to gain an understanding of the actual process, which can thus be improved, become more efficient and effective. By contrast, CEP often aims at identifying event patterns in a live trace of events. In particular, CEP offers means to define complex temporal patterns over events as well as to detect those patterns in live event streams. It allows the usage of contextual information that supports the possible choices to enable reaction to occurring events and to monitor ongoing transportation or shipment.

What generally seems to be two disjoint areas, could greatly benefit from each other through a convergence of formalisms and methods from both worlds. While a BPM system would handle, for example, the overall phase of transportation, the implementation-level monitoring of the truck locations and the container conditions would be handed to event-based processing. Furthermore, the CEP component could transmit the status of the current shipment and inform about bad weather resulting in shipment delays. Thus, real-time reactions could be allowed for modified events. CEP can thus play a role in both the monitoring of ongoing events and in real-time decision support, and extend the model-based BPM with responsiveness to its environment and flexibility. On the contrary, BPM provides CEP with an order how to best process the events (e.g., concurrently or alternatively) and thus allowing to obtain complex-events in a more efficient and effective way. Finally, BPM and CEP both provide means to define events and relationships between them. The focus and origins of BPM and CEP are different and the respective formalisms have different strengths and weaknesses. However, we argue that leveraging the concepts from both origins in combination can yield benefits over using the concepts in isolation.

While various attempts have been made in this direction, they mostly address specific applications, solving specific problems [9–11]. We claim that a comprehensive view should be taken at possible interaction areas in order to identify key opportunities, realize benefits, and tackle challenges. Taking such view, this paper explores challenges that we identified during a Dagstuhl seminar on "Integrating Process-Oriented and Event-Based Systems" [12] in August 2016. Existing literature of related work was analysed to define the scope of our challenges in the context of available solutions. For each challenge,

the benefits of a mutual combination of BPM and CEP are described. The two main objectives of this paper are hence to:

- Provide an understanding of essential concepts of CEP and BPM and strengths of both areas, to lay the ground for supporting their automatic alignment.
- Map the directions in which this alignment can be beneficial and the efforts in these directions. We thereby provide guidance for future research and highlight open questions.

The remainder of this paper is structured as follows. Section 2 provides a brief overview of the concepts of BPM and CEP, applies these concepts to a scenario in the logistics domain, and presents key aspects structured in four "quadrants" arising from the BPM lifecylce. Sections 3 to 6 describe in detail each quadrant: using CEP constructs for process mining, enriching expressiveness of process models, executing business processes via event-based rules, and deriving event-based rules from process models. For each of the quadrants, we identify key challenges and existing approaches. Finally, Section 7 combines the insights from our four quadrants and discusses the core issues involved in all application areas as well as issues that are relevant only to some. The paper closes with a brief summary and conclusion in Section 8.

2. BPM and CEP: Background and Interactions

The purpose of this section is to provide sufficient background information of the areas of BPM and CEP.

2.1. BPM-oriented Systems

Business process models are central artifacts in BPM aiming at documentation, analysis, implementation, and execution of business processes. In a nutshell, a business process model consists of *activities* that are executed in a specific order to achieve a certain goal [13]. Activities can be executed by software or humans, and can vary in the number of required resources and the conditions that determine their order. A business process model describes the three intertwined perspectives of control-flow, data, and resources. The control-flow comprises the activities of the process and the order in which they may be executed, distinguishing between regular behavior and exceptions or compensation. A business process model is typically depicted graphically

and can be executed by a process engine [13], also called *Business Process Management System* (BPMS). The data perspective defines the data elements, e.g., the order, delivery note, customer id, required for the execution of the process together with their scope, their visibility and how they are passed on between activities. The resource perspective clarifies who takes part in which activities, how work is assigned to resources, and other organizational issues such as authorization. Given a (business) process model, one can construct its *traces*, that is, the possible executions of one process instance or case from beginning to end or the (temporally ordered) sequence of all events in one log belonging to one case. Based on these notions, we can define formal criteria of correctness, most famously, soundness [14]. Such criteria describe acceptable behaviors in terms of general properties such as deadlock freedom, proper termination, or bounded resource consumption. Existing analysis techniques can verify the correctness of a process model with respect to such criteria [8]. During runtime, *process monitoring* is a mechanism for providing accurate information of the status of business process instances [13]. This information can be used to provide feedback to a customer [13] or to generate combined metrics such as the number of processes carried out per hour [15].

Typically, there exists a gap between process modeling and its execution: while the process model specifies correct process behavior, the enacted behavior may differ. *Process mining* [16] promotes the understanding of the actually observed process behavior. Process mining refers to a set of techniques that analyze (mostly historical) *event logs* in order to derive a model of the process that created these logs. Generally, one assumes that each event has a timestamp and belongs to one case (process instance). Process mining enables discovery of a process model from a log, measuring the conformance of a log to a model, enhancement of a process model based on logs, and predictions of process properties [17, 18].

The feasibility of these tasks depends on the quality of the data, that is, completeness and correctness of the observed events, and properties are given for each event. Often, the application of pre-processing techniques is required to improve the quality of a log, and to bridge the gap between the activities in a process model with events of a log.

2.2. Complex event processing and event-based systems

Complex event processing (CEP) is a core function of *event-based systems* (EBS). CEP provides means for applications to react to happenings in form of

events by triggering the *events*. Events could be, for example, a new temperature sensor value, a new RFID read, or more generally a state transition. The events are typically described in form of an *event query*, while the reactions are described in form of *event rules* (which comprise a query).

CEP offers an abstraction layer that hides the complexity in detecting such events. The business-level application is only notified about the occurrence of the event and can concentrate on realizing appropriate actions whenever a specific event occurs. The basic state of the monitoring infrastructure, like the kind of sensors available and used for detecting a warning, is no longer of relevance. Further, the application layer does not need to worry how to efficiently detect events. This contributes to a significant simplification of the application by decoupling it from the actual deployment of the monitoring infrastructure. Adding/removing infrastructure elements can be dealt with by adapting the behavior of the CEP in detecting events. Not surprisingly, event processing systems have gained many applications, e.g., in the areas of monitoring critical infrastructure, logistics, and financial applications [19].

Systems that realize complex event processing typically offer a specific query language that allows a domain expert to express when for an application a relevant event occurs. These continuous queries are similar to search queries but filter streams of incoming events instead of querying static data. Event streams typically report primitive (atomic) events. Often systems are not interested in simple events but rather require complex event processing that relies on the detection of composite events, which are formed by logical and temporal combinations of events coming from either a single source or many sources. This typically requires several steps. First, an analysis of the set of potential event producers is needed, e.g., available sensors or RFID readers. Second, events of interest are described. Notifications about such events may then be delivered directly to the application or be used in a subsequent step to identify a complex event. For instance, in the context of a logistics scenario a warning may be triggered when a sequence of 5 sensor readings have each exceeded a critical temperature threshold in the medical container. Likewise, a warning may be triggered if a cooling box was open for longer than 5 minutes.

The identification of specific queries and their appropriate abstraction level typically requires both expert domain knowledge and knowledge of CEP mechanisms. Support for this process of abstraction and formalization is an area of ongoing research with contributions ranging from automatic query generation [20, 21] to supportive CEP query interface [22, 23]. However, this

Table 1: Comparison of BPM-oriented systems and event-based systems

	Event-based / CEP	**Process-oriented / BPM**
Construct	Event	Activity
Goal	Detection of event patterns	Documentation, analysis, execution, monitoring, and improvement of business processes
Methods	Query languages for event specification; Methods for the efficient detection of events; Support for distribution, migration, parallelization; Event dissemination	Designing graphical or formal process models; Formal analysis to verify the correctness of a process model; Process mining to discover a process model, measure the conformance of a log to a model, and enhance a process model based on logs; Monitoring and prediction for process instances
Strength	Decoupling of producers and consumers of events; Hiding details of low level event streams and the infrastructure which is subject to monitoring; Support for the efficient detection of complex events	Understanding and analysis in business-meaningful terms (e.g., activities); Top down as well as bottom up (modeling) approaches; Ability to prescribe and execute behavior; Understanding the difference between specified and observed process behavior

is outside the scope of this article.

CEP query languages typically offer constructs to select event streams of relevance that restrict the set of (primary) events and patterns that could detect a new (complex) event. Various event composition languages exist, most of which support a basic set of pattern operators such as sequence of events, enumeration, and temporal windows over sets of events. A detailed discussion of event languages, algebras and their semantics is described by Hinze and Voisard [24]. From the description of the query model specific operators can be compiled that allow processing event streams at run-time.

While the query language provides constructs for the specification of events, a CEP engine supports the efficient detection of specified *event patterns*. This comprises in particular the following aspects: First, CEP engines offer support to access event streams that are required to detect a specific event pattern. Second, it provides means to efficiently execute operators that are needed to detect events. Third, it provides means to define the event query. Approaches towards efficient CEP engines offer, in addition to the detection of events,

methods for the distribution, migration of state as well as parallel execution, and therefore allow for highly flexible processing of event streams that, for instance, allows to minimize bandwidth consumption, the throughput of events that can be processed by a CEP engine, or the latency it takes to detect a complex event. Table 1 provides a summary of the main issues discussed regarding BPM-oriented systems and event-based systems.

2.3. Application of BPM and CEP concepts

This section introduces a running example to demonstrate challenges and opportunities of combining BPM and CEP. The example concerns the shipment of medicines from an industry warehouse to the nearest hospital. The shipment has constraints both in time of consignment[1] and in environmental conditions of the goods during transportation. For simplicity, we consider three main phases (see Figure 1), namely (i) the *preparation phase*, with shipment documentation, reservation of the truck, and the loading of the goods; (ii) the *transportation phase* referring to the physical movement of the truck and the conditions in the container, and (iii) the final *consignment phase*, consisting of arrival, unloading and handling of shipment documents.

Figure 1: A logistic business process model

Figures 2 to 4 refine our logistic business process model. Three main actors are considered in this scenario, namely: (i) the transportation and logistics planner, (ii) the truck and truck driver, who takes part in the physical transportation of the goods, and (iii) a monitoring system as an intermediate layer. The monitoring system is meant to be the novel actor that we envision in the overall event-driven process architecture. It serves as the middleware between the process-driven execution and the event-driven retrieval of information (gathered upon the process execution).

In the first activity, "Prepare transportation" (see Figure 2), the planner starts the process by reserving the truck for the shipment. Thereupon, the

[1] A consignment is the delivery of the smallest portion of goods that is necessary to track and trace after bundling it for shipment.

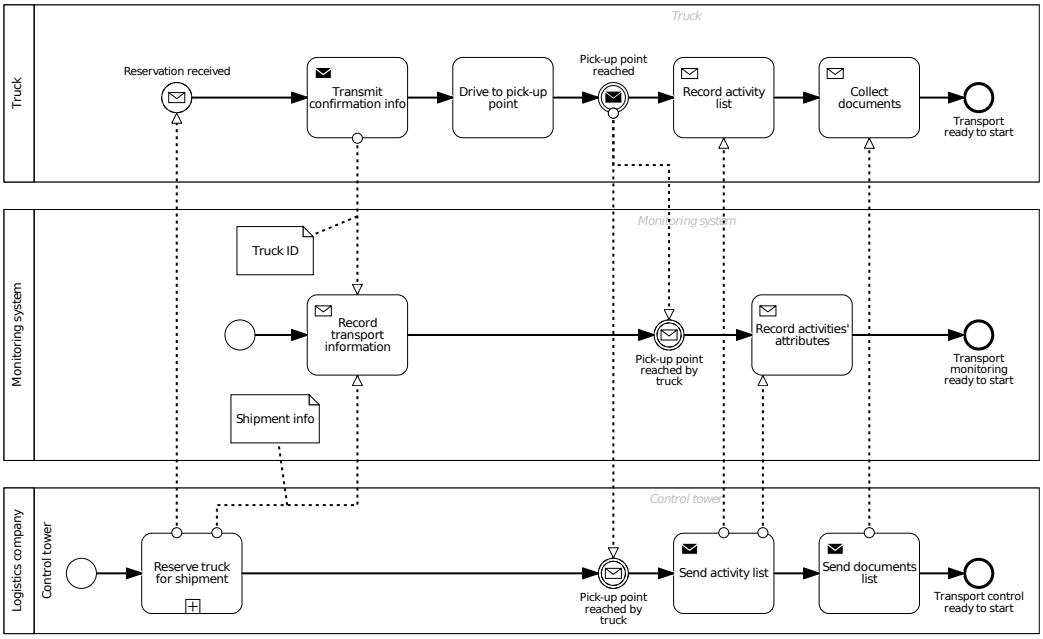

Figure 2: The "prepare transportation" sub-process

shipment information such as ID, estimated times of departure and arrival, etc., are registered in the monitoring system. On the other side, the truck driver communicates the truck ID to the system. The provided coordinates make it possible to link the information stemming from the truck to the shipment, that is the main business object of the process. The details about the list of activities to be carried out, such as the transmission of documents, the gates at which picking up the container and release it, etc., are given directly to the truck driver and recorded in the system as well.

In the second activity, "Transport goods" (see Figure 3), the driving starts after loading the container, in order to consign the goods at the expected destination. Subsequent notifications of the reached positions are automatically communicated via transponder from the truck to the monitoring system, which interprets the positional updates as stages of the ongoing transportation. The planner is thus kept informed on the evolution of the task, until the container is unloaded. We remark here that the monitoring system is meant to transform low-level input events such as the positional updates into more informative events for the planner, such as the beginning, progress stage reached, and end of the transportation.

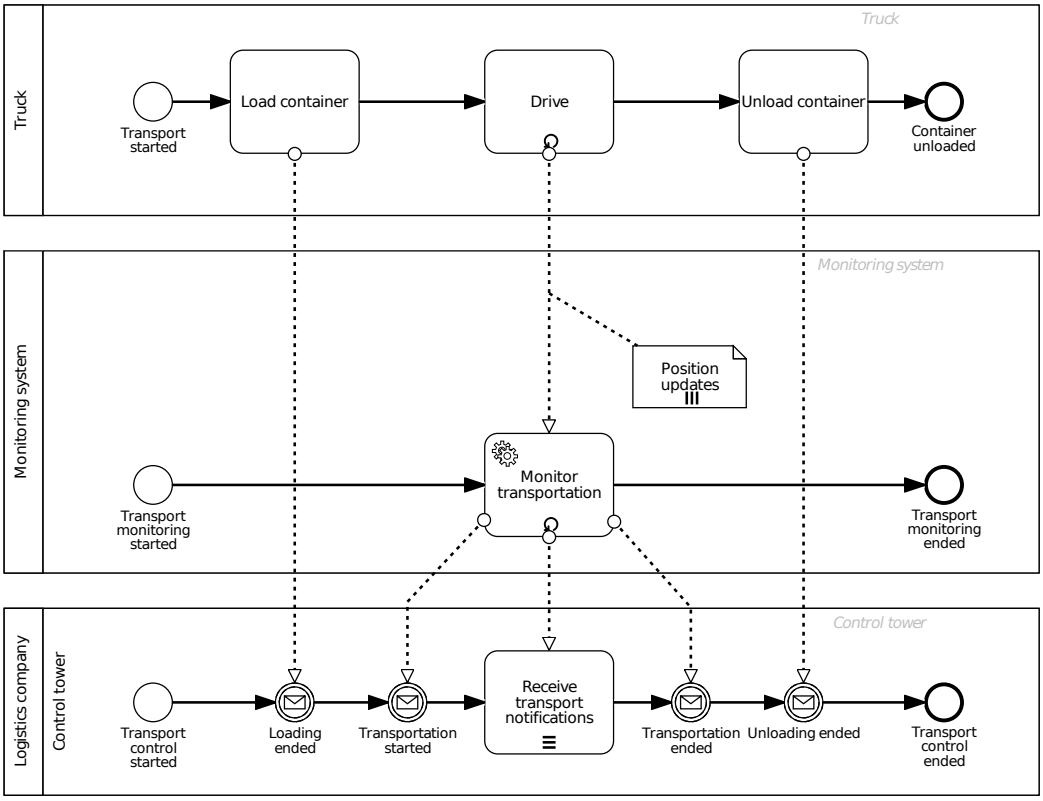

Figure 3: The "transport goods" sub-process

Finally, the confirmation of the receipt and other documents are transmitted from the truck to the planner and the process fragment terminates. The third activity, "Finalise consignment" (see Figure 4), sees the two actors of truck and logistics company exchanging the final documents to confirm the consignment. The monitoring system is no longer involved.

In the example, the domains of CEP and BPM can mutually benefit in several ways:

- In a top-down view, the process model would define the activities and constraints, and determine the events that are to be monitored to ensure quality of service. Thus, it allows for an effective control of the ongoing shipment and processes by processing the status of current shipments.

- In a bottom-up view, the available event monitoring data follows the process model thus allowing for an accurate estimation of the time of

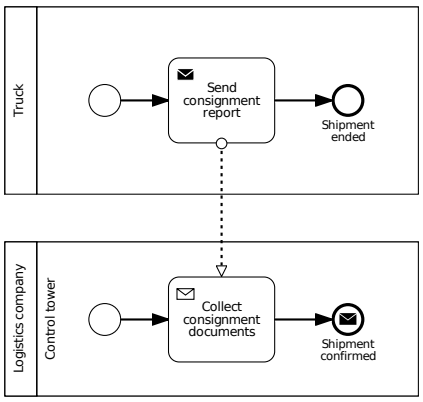

Figure 4: The "consignment phase" sub-process

Table 2: BPM systems and complex event processing in the running example

Activity	Processing the order; Transportation of medicine; Adjusting the temperature in the box where the medicine is shipped
(Complex) event	A movement of the truck determined by a position update; A good is loaded on a truck, e.g., based on a RFID reading
Process Model	Shipping of medicine from a warehouse to a hospital while meeting quality constraints
Event pattern	Determine outliers in transportation and arrival time
Complex event processing	Trigger the event of a warning when the temperature of the shipped medicine has reached a critical threshold

consignment.

Exemplary BPM and CEP elements that can be found in this scenario are listed in Table 2.

2.4. Mapping combinations of BPM and CEP

BPM and CEP follow different purposes and rely on conceptually different foundations, which generally hamper their seamless integration. The scenario above shows how both areas can gain from each other synergetically. Overall, going from raw events to process models and backwards, we identified four specific areas where the combination of process models and CEP forms potential opportunities and challenges. We classified them with respect to

the four phases of the BPM lifecycle [13] into four quadrants of a map, see Figure 5.

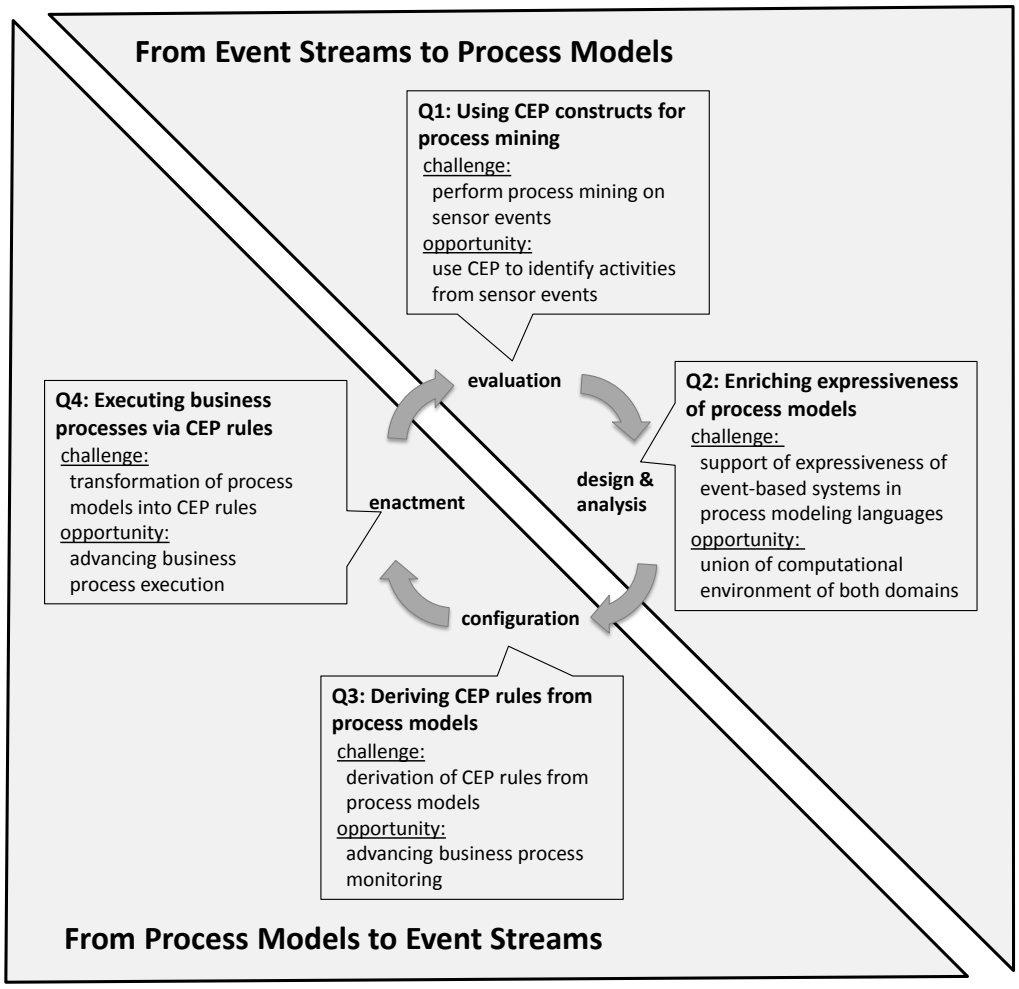

Figure 5: Four key quadrants of combining BPM and CEP concepts

In our map, two quadrants (Q1 and Q2) take the direction from CEP to BPM, and are related to the phases of evaluation and of design & analysis of a process model. In the BPM lifecycle the **evaluation** phase addresses the evaluation of process models using business activity monitoring and process mining techniques. The **design & analysis** phase tackles the creation and validation of business process models. From an event streams to process model perspective the quadrants Q1 and Q2 examine the idea of extending

or creating process models based on CEP and their concepts. Q1 addresses how process models can be created or enhanced through the use of complex events in process mining. Q2 is concerned with enriching the expressiveness of process models through event-based concepts, so that the process model may also benefit from language features available in CEP.

The other two quadrants of the map (Q3 and Q4) take the direction from process models to CEP, and relate to the phases of configuration and enactment. The **configuration** phase within the BPM lifecycle is about technical realization of a process model, including setting specific parameters to tie implementation details to the higher-level model. Therefore, Q3 is related to this phase, exploring how to create monitoring CEP queries and how to derive CEP patterns from a process model. The **enactment** phase is related to the execution of process models. Q4 is about executing a process model through a CEP engine, namely, injecting events and managing event streams in order to realize a business process.

The classification according to the BP lifecycle shows that we consider challenges and opportunities for each phase.

The next sections will discuss each of the four quadrants in turn. Based on our observations, we then discuss the core issues and challenges of the relationship between process models and CEP.

3. Q1 – Using CEP constructs for process mining

This section examines the idea of using CEP constructs to bridge the gap between abstraction levels of event logs. We focus on two classes of event logs: (i) primary event sources, e.g., event logs based on sensor data and (ii) event logs fulfilling the requirements for process mining. Process mining algorithms expect event logs at a certain level of abstraction to produce (business) process models. With primary event sources this abstraction can be obtained only with great effort. In this context, we focus on the expressiveness of CEP languages for defining complex event patterns rather than on the real-time execution with CEP.[2]

Usually, process mining works with recorded events or real-time data of information systems (e.g. truck loading ended) and not with low-level raw data (e.g. raw RFID readings). CEP technologies and the corresponding

[2]Note that CEP is likely run over log files and not over streams in this scenario.

high level languages are designed to support the inference of complex events from raw (input) events. Therefore, the use of complex event processing for process mining has the potential to bridge abstraction levels of events meaning that process mining techniques can also process raw data. On the contrary, process mining techniques could be valuable to IoT data allowing to derive assumptions about the appropriate placement of sensors.

3.1. Motivation and Research Question

To demonstrate the potential of combining CEP technologies and process mining, we consider the item observation in our running example. The example includes loading and unloading of a truck. These activites can be observed by RFID readers and other sensors and reflected in the IT system.

However, the read events that RFID readers and sensors capture are distorted by interferences and other physical effects in the wireless communication [25] as well as by noise and errors in the measurement process itself (e.g. misconfiguration of readers). Hence, instead of a clean event like "person entered loading area A", the system receives a noisy sequence of sensor observations. For instance, Jeffery et al. [26] demonstrate in their experiments how RFID readers and motion sensors capture events when a person with an RFID tag periodically leaves and enters a room. Their experiments show that both sensor types may create events, even when no person is present or fail to report events when a person is present. Therefore, there is a non-obvious mapping between a raw sensor read and the presence of a person.

The experiments by Jeffery et al. [26] show how events like "product placed on shelf B" or "person entered work area A" may enter the system. Approaches for process mining typically assume that the analyzed log files directly reflect activities on the abstraction level of the generated process model. They typically assume that the events in the corresponding logs are on a suitable level of abstraction, i.e., they reflect the activities in the business process. From that perspective, "product removed from a shelf" and "product placed on shelf B" or "person active in work room A" may reflect activities that are suitable inputs for mining a process. However, events that enter real world applications are often captured on a much lower level of granularity, in particular with the ongoing rise of IoT. Bridging such gaps is at the core of CEP query languages. Also more data sources with indirect reflections of business activities become available. A specific activity may often not be directly observed but needs to be inferred indirectly from a multitude of observations. For instance, one may need to infer the activity of loading a

truckraw input streams, so that process mining can incorporate the loading activity in the overall process model. This may involve cleaning of noisy RFID signals and correlating data from multiple RFID readers and/or light barriers. CEP languages hold many constructs that aid such tasks.

The combination of CEP techniques and process mining has the following benefits:

Improving the quality of event logs for process mining. IoT data sources often include noisy and erroneous inputs. Instead of filtering in the stream, the integration of CEP and process mining allows deferring the error correction with CEP to the mining process (no raw information lost). Event patterns may be mined to derive the pre-processing of events.

Bringing context into CEP event logs. Process mining algorithms intend to find behavioral relationships (e.g., sequence, alternative, parallel) between activities and thus bring activities into context (they define their execution order). The application of techniques for finding behavioral properties on IoT data would enhance CEP engines. CEP events could be considered in context (e.g., due to railroad conditions, the track is still unloaded).

Providing a human-understandable interface for event streams using process mining visualizations. Process mining derives a process model from traces providing a structured representation of activities. To visually recognize any changes of activities it is common to use Dotted-Charts [27]. Such a chart representation can be used to event streams to improve the outliers identification.

Following these potential benefits we derive the following research question:

How to enrich process mining with CEP techniques to identify meaningful business (high) level events out of low level (raw) events and how to benefit from CEP techniques to automate low-level logs pre-processing techniques?

3.2. Existing work related to Q1

To date, there exist neither methods for coherently combining CEP and process mining into end-to-end solutions nor a well established tool chain. However, several works address parts of the problem. Related approaches stem from a range of different domains and a complete review is beyond the scope of this paper. Therefore, we discuss relevant approaches along selected examples of existing work.

In general, we see four main steps in generating business process models from sensor event logs or applying process mining to sensor data to bridge level of abstractions. These are: (1) Cleaning and pre-processing of sensor

data, (2) abstracting events, (3) enriching with context information, and (4) finding recurring patterns to derive processes.

Cleaning and pre-processing of sensor data. Many works demonstrate the suitability of CEP and stream processing and related constructs for detecting business relevant events in low level data streams [28–34]. Moreover, a considerable amount of work exists on using CEP and middleware with CEP constructs related to cleaning and pre-processing of IoT data [25, 26, 28, 33]. These works support the technical process of deriving complex events from low level inputs with operations like filtering, aggregation and correlation. However, they do not address how to create process models from the results. Initial approaches can be found that apply process mining to raw data. For instance, a transformation approach is discussed by van Eck et al. [35] applying process mining techniques to sensor data. The available approaches have in common that they intend to map sensor data to activities beforehand.

Abstracting events. The abstraction of events bridges between raw low-level event logs (e.g. sensor observations) and events at process level (e.g. loading a container). CEP is well suited for executing the logic that facilitates this abstraction (i.e. as CEP rules or queries). However, devising the abstraction logic is a different concern. A number of works at the intersection of CEP and machine learning exists, which target this particular aspect. [36] describe a general framework called iCEP for automatic generation of CEP rules. Tax et al. [37] extract features from a window of events to train a model for abstraction with supervised learning (i.e., conditional random fields). Considering differences in abstraction levels of process models [38], Smirnov et al. [39] transform a process model into a simplified version on a higher level of granularity, while preserving certain properties of the original process model. The abstraction is performed either by aggregation or elimination of labels of process activities [40] or process elements [41]. Works of this type are examples for research that addresses how to generate abstraction logic, if labels for training abstraction models are present. However, such approaches are orthogonal to approaches that leverage knowledge of domain experts who can provide explicit abstraction rules (e.g., CEP constructs). In contrast to using e.g., CEP constructs for abstraction, most machine learning techniques yield abstraction logic that is hard to comprehend by humans. The suitability of using machine learning or encoded expert knowledge depends on the requirements of the application domain. Baier et al. [42, 43] exploit

Table 3: Summarizing capabilities of process models and CEP regarding the main issues discussed (Q1)

Issue	Process Mining	CEP	Application to example
Cleaning and preprocessing of sensor data	NA (not a core concept)	Supported in CEP by filter rules and possibly the combination with window constructs	the location of a good via reading RFID tags at a certain reader is not distorted by interferences and no noisy sequence of sensor observations are delivered.
Abstracting events	Aggregation and elimination on the level of process activities	Supported through output definition of CEP queries on any level of abstractions	Fitting sub-processes of the logistic business process model can be derived from event logs.
Enriching with Context Information	Support through alignment of ontological concepts to activities or training data	Execution supported by integration of knowledge bases and joining streams	The interference of exogenous factors (e.g., temperature, humidity, etc.) on transportation monitoring are understood.
Finding Recurring Patterns to Derive Processes	Process mining uses case identification as a basis	Detection of events about the same object as basis	Tracing object in a shipment across different levels of aggregation (i.e. before and after beeing packed in a container/loaded on a truck)

the comparison of the respective behavioral relationships to link the events in the log to the activities in the process model. Behavioral relationships are used to define the constraints for a Constraint-Satisfaction Problem (CSP), which automatically defines the matching. When ambiguities cannot be solved, the knowledge of process analysts is required to discriminate to which activity an event should be associated. The approach has been then extended in [44] by including natural-language processing techniques to compare the labels of events and activities in the model. Similarly, Mannhardt et al. [45] rely on behavioral activity patterns that capture domain knowledge and tie events to activities. Then, they align logged events to process activities. Beyond the question of how to define abstraction logic, abstraction techniques are concerned with the execution model and implementation of the logic. Stocker et al. [46] present an approach that explicitly enables the use of CEP and machine learning based abstraction. They propose a framework with four technical layers that explicitly includes CEP and machine learning as alternatives for implementing the abstraction logic. While Stocker et al. position their work in the context of situational awareness, it shows concepts for bridging abstraction levels with CEP that relate to business processes as well.

Enriching with Context Information. Context enriching in BPM can be implemented through ontologies. Sztyler et al. [47] construct an activity

ontology in the context of health-care. Alternatively, the use of a time-based label refinement is suggested [48]. Further approaches train running instances at run-time with past instances [49] or use context-related execution scenarios [50, 51] in order to enrich the context. However, these approaches do not address the identification of behavioral relationships between event data. Instead, context information is only used for the selection of appropriate events.

Finding Recurring Patterns to Derive Processes. A main assumption underlying process mining techniques is that events have an attribute, which uniquely identifies the process instance they relate to, namely the case ID [16]. Event-based systems may support identification of case IDs though incorporation of knowledge bases and correlation of events. An example is RFID middleware that enriches raw observation events with information about the corresponding process [33].

In Table 3 we provide an overview how CEP and process mining techniques support the above discussed steps. The table stresses upon the fact that CEP and process mining do not support all aspects individually, but rather cover the whole spectrum in combination.

4. Q2 – Enriching Expressiveness of Process Models

This section explores the challenges and opportunities of using CEP concepts to enrich the expressiveness of process models. We use our running example (see Section 2.3) to highlight the gaps between process languages and practical needs, and highlight the opportunities for CEP involvement.

4.1. Motivation and Research Questions

Languages for business process modeling are devised to orchestrate services and to involve humans for the enactment of a process. They offer control-flow constructs to wire the activities together. They further offer the opportunity to include in the workflow the raising (throwing) and elaboration (catching) of events, including the handling of exceptional circumstances. However, the common standard modeling notations hardly utilize advanced models and features in dealing with events. For instance, the business process execution language Business Process Execution Language (BPEL) [52] treats events as messages and does not provide the modeler with the opportunity to define complex events within the model [53]. In contrast, BPMN explicitly includes

the notion of events such as timers, signals or exceptions, but lacks the capability of handling or defining complex events. Thus, the expressiveness of workflow languages is limited with regards to event descriptions.

Capturing of and reaction to complex events. We consider again our running example of medical transportation. Let us assume that the transportation ended, but the current time exceeds the expected time of arrival by more than a predefined threshold. We now wish to model that when a delay occurs, a compensation action needs to be triggered. To date, process modeling languages do not support machine-readable statements that allow for an automatic identification of the arrival of the truck at destination: the consecutive deceleration, stop, and switch-off of the engine in the surroundings of the destination would be a complex event signaling it. Although expressible as a CEP query, the inclusion of such extension points towards complex event processing is not a standardized. Currently, even defining the criteria to classify the completion of the activity as late is not part of the standard process modeling capabilities, let alone the exact way to capture such an exception from low-level input events. Text annotations may be used as a workaround [54]. However, those are not meant to be processed by a BPMS [6]. Alternatively, event-driven gateways may be used, which are triggered by a designated event that receives a message generated by a CEP engine [6, 55]. In this case, the required expressiveness is not supported by current process modeling languages.

As a consequence, the CEP queries and process models are specified in two separate computational environments and no explicit link is provided to bind the two. Bringing the CEP specification to the business process model would decrease such a gap. This would ensure that the person modeling the business process is in control both of the higher level of abstraction of the process model, and the related concrete information extracted from low-level input events. Using such an approach could enhance the expressiveness of the process model.

Managed begin and end of monitoring. From a practical perspective, we can see that CEP is not required to come into play throughout the whole execution of the process. For example, the process might not need the events processing right from the beginning, but rather once they become of interest. In our logistics example, the location updates of the truck may be of little interest before the truck has been actually assigned to the shipment. Furthermore, the

event monitoring may not be needed for the whole duration of the modeled process. In our example, the monitoring of the truck's location updates is only of interest until the unloading of the container is finished.

Engaging CEP monitoring only when needed by the process (i.e., on demand) would allow workload reduction in the CEP engine. Therefore, the CEP engine needs to be aware of when the event monitoring is required by the process.

Dynamic assignment of event parameters. Finally, CEP queries serving for event monitoring need to support parameters that can be specified not at design time (i.e., process modeling time), but rather kept unassigned until the process instance is unfolding, hence at run-time. In our example, such parameters could include for example the truck identification number, the container number, the pick-up point and the destination point, the estimated times of departure and arrival. In light of the aforementioned workload reduction for CEP engines, only the trucks actually involved in the shipment should be monitored in the scope of an ongoing process. However, realizing such *late binding* of values and parameters in CEP engines requires instantiation of respective parameter values and invocation of corresponding information sources at runtime. CEP languages typically provide constructs that allow for the definition of abstract query templates which determine parameters as part of their execution. Furthermore pub/sub mechanisms allow for invocation of event sources at runtime. However, it remains a challenge of integrating CEP with BPM to seamlessly derive the concrete CEP queries that realize the late binding to process instances. For instance, in our running example a filter may be implemented to select relevant trucks based on the information carried by events about trucks reservations from the logistics company. A domain expert can write and deploy the necessary query with available technologies. Better integration of CEP and BPM should support this implementation and reduce the manual effort of domain experts. Ad-hoc solutions can be implemented in BPMSs that allow to record instance-specific data within, e.g., the so-called information artifacts in BPMN [6, 56]. The missing connection between BPMSs and CEP engines makes it however not possible to explicitly assign query parameters with instance-specific information.

In Table 4 we provide an overview on how concepts of CEP query languages can be integrated with process modeling to their mutual benefit. These three aspects lead to the following research question:

Table 4: Summarizing capabilities of process models and CEP regarding the main issues discussed (Q2)

Issue	Process Modeling	CEP	Application to example
Capturing of, and reaction to complex events	Only via non-common extensions	Implementation-level complex event languages exist as well as higher-level algebras	Automatic detection of the late end of transportation based on events from the truck, and start of compensatory activities
Managed begin and end of monitoring	Monitoring per process instance. Subscription to message-based events upon reaching the respective activity. Unsubscription directly upon receipt. Custom extensions may change that behavior.	Based on subscription (upon event registering and unregistering)	Truck positions are of interest to the process execution only after the reservation until the end of the shipment
Dynamic assignment of event parameters	Partially covered by information artifacts	Not available	Pick-up point and destination locations and times are defined on a per-instance basis, hence assigned at run-time.

How to enrich process modeling languages with CEP information while ensuring that CEP queries are executed only when needed and in relation to objects of interest to the process?

4.2. Existing work related to Q2

CEP rules capture the dynamic nature of the evolving process tasks by comparing the observed evolution with expected trends, bounded intervals, and thresholds. It is a common assumption that such information is present at design time [57, 58]. As outlined above, previous knowledge of the referenced values is often not available at such an early stage. Here, we discuss a number of approaches that allow enrichment of processes by event information, some of which consider late specification.

Basic BPMN. Recker et al. [59] conducted a study showing that 26% of the interviewees "indicated that they are limited in capturing events, 80% categorized this limitation as a problem (minor or major)". In BPMN, CEP constructs can be realized building on low-level BPMN tasks such as script tasks or business rule tasks. This leads, however, to somewhat bloated BPMN models, because much more constructs are used. For instance, Barnawi et al. [60] use the expressiveness of BPMN to embed tasks to detect and handle violations of compliance. To enable checking, it is assumed that the process engine provides information such as the completion time of tasks.

Extending BPMN with event specifications. An architecture for the integration of the event-driven analysis of ongoing process instances has been proposed by Baumgraß et al. [54], using parametric expressions for information that should become available upon enactment. The proposed approach is based on the BPMN-T extension of the BPMN language [61], which specifies ad-hoc time annotations, process snippet interfaces, and event subscriptions especially for transportation processes inserted in text comments. However, our objective here is not limited to the scope of a sole application domain such as transportation or logistics. Furthermore, text annotations are not suitable for a standardized formal solution, owing to the fact that they are not meant to be normed by machine-readable grammar rules, but rather used as human-comprehensible further explanation on the details of the process model [6].

Baumgraß et al. [62] introduced a BPMN extension using Process Event Monitoring Points, which specify where and when which event is expected during business process execution. Event and process information are associated with data and activity state changes, such as the begin or end of the process execution. Their approach uses the Esper Query Language queries to include event monitoring in BPMN [63].

The modeling language rBPMN is an extension of BPMN with the aim to "integrate both rule- and process-oriented modeling perspectives" [64]. The rules are written in the REWERSE markup language [65], using business rules to define and constrain the execution of a business processes. They can be divided into rules for derivation of new rules, specifying integrity, production of actions and reaction to processed events. The rBPMN approach then allows to model some parts of the process based on business rules which are to be evaluated at runtime. If those business rules can change during runtime, or are very dependent on data / context / environment-dependent, a CEP engine could be applied to evaluate the rules at runtime.

Mandal et al. [66] recently proposed an extension of BPMN defined as a BPMN+X model [67] to specify the automated services requiring a connection to external CEP engines. The subscription to the external data source can be customized by means of specific directives regarding a.o. the subscription query, subscription point, buffer size and policy. They root the execution semantics of the extended process model in Colored Petri nets (CPNs) [8] to allow for formal verification. A proof-of-concept implementation based on the

open-source process engine Camunda[3] is made available for testing purposes and to demonstrate the viability of the model extension as implementable over off-the-shelf process engines.

Executable event pattern languages and BPEL. A number of authors from the business process community call for the development of executable event query languages. These should implement the expression of event patterns and their relations within process models (see, e.g., [11, 68]).

In their work, von Ammon et al. [69] suggested to extend the workflow execution language BPEL to cover event concepts that influence the process execution. For example, they introduced event subscriptions and a simple pattern constructs, which allowed the start or ending of a process depending on the observance or lack of an event. While considering a joint execution of events and processes, the proposal is not concerned with detailed constructs such as dynamic parameters and CEP on demand.

Similarly, Wieland et al. [70] use an extension of BPEL (called Context4BPEL) to introduce context awareness into workflows using so-called context events. Using context events, processes can be started, query the environment, and execute decisions based on the context. They use the Augmented World Query Language for context queries and the Augmented World Modelling Language for result representation [71]. Although process instance data can be used within the queries, the query language supports geo-locations only and does not offer the expressiveness of CEP languages.

Executable event pattern languages in other languages. Wieland et al. [55] propose the SOEDA method to support both event-driven systems and service-oriented architectures. Business processes are first modeled using Event-driven Process Chains (EPCs) and then transformed into BPEL expressions. The resulting applications can use process-oriented workflows with events that trigger the execution of business activities. A shortcoming is the limited coverage of event patterns and the lack of dynamic adaptations of processes based on events at runtime [72].

Vidackovic [72] also proposes a method for modeling, platform-specific transformation and implementation of dynamic business processes based on CEP concepts. He introduces the Event Processing Model and Notation

[3] https://camunda.org

(EPMN), which has a graphic representation similar to BPMN, and uses XML and the Esper query language [73] for implementation.

Seiger et al. [74] propose extended Petri nets to annotate event queries to transitions. During execution, the event query is sent to an CEP engine, which registers listeners and upon pattern detection leads to an execution of the respective action.

Graphic notations for events in processes. Decker et al. [75] suggested the Business Event Modeling Notation (BEMN) for expressing complex events in business processes. They proposed a graphical notation as well as a formal semantics. The language covers only a somewhat limited range of event concepts (which were developed based on observations by Barros et al. [76]) with little concern for real-time execution and does not yet provide the complexity of native CEP languages.

Kunz et al. [77] describe another graphical approach of integrating complex events, described in Esper query language, into BPMN. They showed how the core query language clauses select, from and where can be assigned to BPMN artifacts. Their aim was to provide better usability and user adoption for Complex Event Processing. The integration of event processing into the process model is limited and no dynamic interactions between events and processes are considered.

Breitenbücher et al. [78] introduce the SitME method (Situation-Aware Workflow Modelling Extension method). The idea is to offer a modeling environment for workflows including of situational events and situational scopes. Situational events mirror situational templates [79, 80], which define a situation based Situation-Aggregation-Trees (SAT) [81]. Breitenbücher et al. also outline runtime aspects: The workflow is then transformed to a standard-compliant workflow. For recognizing situations, either Node-RED flows or CEP engines can be used.

Analyzing events in process models. There are already previous comparisons of existing work on incorporating CEP concepts into process modeling.

Barros et al. [76] present a catalog of 13 requirements for handling complex events in process models, mainly driven by event concepts from the process community. Examples are simple compositions such as as event conjunction and disjunction as well as complex composites, such as dependencies with regards to data and process instances. They argue that both BPEL and BPMN support only a very small selection of event patterns. Similar to our

arguments in this paper, Barros et al. argue that event pattern descriptions need to be closely integrated into executable process definition languages.

Vidackovic [72] analyzed the process modeling languages EPC, BPMN 2.0, UML [82], and WS-BPEL for their support for event concepts. Only EPC and BPMN are found to fully embrace event monitoring. However, none of the four languages are found to sufficiently support complex events. The same work also analyzed selected approaches to event-driven business process management with regard to their support for complex events (based on CEP concepts [83]). They considered complex event patterns as well as execution parameters for sliding windows. Vidackovic observes that existing approaches each address selected aspects, without yet providing integrated or standardized solutions. He particularly notes the lack of execution semantics, insufficient modeling support, and lack of software tools.

Involve CEP on demand. When running a processes, it is assumed that all services are available at the instantiation of the process. This, however, may lead to increased costs as the services might not be always required. As a consequence, the idea is to use the concepts of Cloud Computing. It is possible to provision the required services in the Cloud at the beginning of the execution of the process. In processes, there is the possibility to provision services on demand [84]. Is is also possible to advance even further and to deploy the process middleware on demand [85]. There is also general work for service deployment [86], but it is not aware of business processes execution. This concept can be extended when the business process model is enriched with CEP constructs. The CEP engine itself can be provisioned on process start or even the middleware required to read the sensor data can be provisioned on demand.

5. Q3 – Deriving CEP Rules from Process Models

This section explores the direction of deriving rules, more specifically event processing or CEP rules, from a process model to allow us to monitor the process at runtime. Using rules for monitoring a business process is not a new idea. At the basic level, "normal" rule patterns can be derived in a bottom-up manner by analyzing an event stream (as described in Q1). In contrast, here, we discuss the derivation of rules from a process model in a top-down manner. This raises a number of challenges.

5.1. Motivation and Research Question

In order to explore this direction and understand the challenges, let us consider the running example (see Section 2.3), where medical products transportation requires very special conditions, including a stable temperature and humidity level in the container and a predefined maximum transportation time. Sensors in the container and transponders installed on the truck can help to gather the relevant information.

Considering the process model, the "Drive" activity of the truck (see Figure 3) can be monitored for various purposes. One purpose can be the temperature and humidity within the container. In addition, taking as an example the expected locations for the start and the end of the "Drive" activity makes it possible to monitor the ongoing execution of the activity, displaying the gained and remaining distance, e.g., with respect to the current position of the truck. Deviations of the current position from the expected one can be reported. Predictions of schedule deviations can be made, based on comparing the remaining time left to deliver the container with the estimated time of arrival, on the basis of the current position of the truck and its speed. In contrast, monitoring the "Collect Documents" activity would be less challenging, and entail comparing the already collected documents to a predefined checklist. Monitoring in this case is not too beneficial as the uncertainty is limited and the events are simple, reporting a task performed without the need for a complex query logic.

Monitoring capability can depend on available sources, and these will be relevant for the duration of specific activities. For example, the traffic information is of extreme interest while the shipment is made by trucks, but only during the "Drive" activity and until the goods are delivered. The same rationale holds for a storm forecast along the way. Another issue relates to the granularity level, which is different for a process model and for event-based monitoring, including many details that are instance-specific. As an example, upon instantiation, the "Drive" activity would entail different routes, quantities, products, and so forth for each process instance.

Following the example, we identified two research questions along which we discuss related work: *Which kinds of process parts/activities and what business purposes require CEP-based monitoring or rules? How can CEP queries and rules be derived from process models*

5.2. Existing work related to Q3

Purpose of CEP rules. CEP rules derived from process models can serve a variety of purposes, such as monitoring of the process progress [77], validation of conditions defined over the process [87] or monitoring its compliance [88, 89].

In particular, derived rules should be used for monitoring various aspects of a process. One specific form of monitoring, which has received relatively high attention, is the monitoring of Service Level Agreements (SLAs) [87]. SLAs define the level of service that a service provider commits to deliver. An SLA is a contract between a service provider and a consumer, and it includes appropriate actions to be taken upon violation of the contractual obligations. Muthusamy et al. [87] present a vision to achieve end-to-end SLA management by facilitating the various stages of business process development using formally encoded SLAs. They develop a model to control the provisioning of business processes based on high-level goals described by means of event-based rules that can be specified independently of the implementation details of a process. Furthermore, Chau et al. [90] show how an SLA contract can be modeled and designed to be configurable, reusable, extensible and inheritable. The modeling resorts to rules to express runtime conditions and constraints over process execution events. The resulting rules automatically monitor a business process and evaluate whether the SLA is violated during runtime execution. SLA monitoring can be separated from or combined with general monitoring against compliance requirements, as suggested by Thullner et al. [91].

In the area of business processes a more general form of monitoring which is sought is predictive monitoring of various process aspects. Much attention has been put into predictive monitoring recently, and it appears that CEP provides promising solution directions. Metzger et al. [92] provide a survey of corresponding techniques and mention the use of CEP in some analyzed works. These techniques, however, take a bottom-up approach of learning from the event stream, and do not incorporate knowledge from the process model in a top-down manner.

Derivation of rules from process models. Several authors have considered deriving CEP rules from process models. Here, the term rule has the broadest possible interpretation and refers to antecedent-consequent structures (i.e., if-condition-then-action) [93, 94], patterns (i.e., (complex) event expressions) [94], and subscriptions (i.e., state or non-state baring Boolean expressions over predicates to tie events to process elements) [54] For exam-

ple, the approach proposed by Muthusamy and Jacobsen [95] is based on decomposing a process into expressions that describe sets of events that may occur when the process executes.

How to apply the idea of decomposing a process into constituent rules that are triggered by the events underlying the execution of processes (process instances) to BPEL has been shown by Li et al. [96] and to the Guard-Stage-Milestone (GSM) with Lifecycles by Sadoghi et al. [94]. Some of the difficulties related to the derivation of rules, taking BPMN as a basic process model, are discussed by Bry et al. [93].

Addressing processes based on web-services, Mulo et al. [88] derive CEP rules in two steps: first, they transform process activities into event trails that correspond to the invocation of the respected web services, and then they transform these trails into CEP rules.

Baresi et al. [97] define and formalize an approach based on an extension of the artifact-centric language GSM (E-GSM), to automatically translate process models defined in BPMN into E-GSM and monitor the execution of such processes through the observation of the related objects. Baresi et al. [97] in particular show how a monitoring engine based on the E-GSM specifications can detect anomalies during the execution of the process and classify them according to different levels of severity, that is, with respect to the impact on the outcome of the process. Meroni et al. [98] propose a framework based on [97] that allows for monitoring of processes based on the recorded status of machine-tracked objects instantiating the artifacts in real world. Considering the control flow in imperative process models, Weidlich et al. [99] proposed a two-phase approach. First, to derive behavioral profiles of the process, in particular considering strict order of activities, exclusiveness, and interleaving (parallel) execution, and then to derive CEP queries that correspond to these behaviors.

Approaches have been suggested to support transforming higher-level process tasks into lower-level detailed event rules, addressing the issues of transformation between constructs (activity and gateway to event), differences in abstraction levels, and the specific information that can only become available upon instantiation. For example, Cabanillas et al. [100] proposed an approach aimed at enriching the definition of activities by means of attributes. The attributes in particular were meant to be subject to constraints about the values they can assume during the execution of the activity, in addition to thresholds defining the minimum peaks allowed. A typical example for that is the temperature of the container, with allowed maximum and minimum oscil-

lations. Based upon this model, and specifically targeting context-awareness, a system has been developed and described by Di Ciccio et al. [101], capable of alerting logistics companies in case of predicted diversions during the aerial transportation of goods. Alerts were raised on the basis of the automated classification of ongoing flight data features. A similar use case has been addressed by Baumgraß et al. [54], who annotated BPMN elements with subscription templates, using placeholders for details which specifically relate to the process instance during execution.

In the approaches of Cabanillas et al. [100] and Di Ciccio et al. [101], the devised analysis is purely numerical though, and complex rules beyond the comparison of gathered values with given thresholds cannot be expressed. Furthermore, streamed raw data needs further rework to be effectively tractable. In particular, event correlation needs to be established, tying events to process instances and fixing incorrect ordering of events over time [91]. The interpretation of raw data can be assigned to CEP engines that derive exceptional conditions by the examination of the data streams. To that extent, the specification of monitoring rules by means of a machine-readable language is of utmost importance. They are meant to describe the way in which raw data need to be processed and interpreted in order to provide high-level events describing the progression status and the health status of the ongoing activities in the process. Monitoring rules that co-exist with a process model are suggested by Thullner et al. [91]. In their approach the process definition (model) includes compliance check points, while the compliance rules specify conditions over parameters at these check points and actions to be performed upon violation. The conditions can address the process execution time or its flow, and further relate to specified event attributes (e.g., "gold customer"). A highly generic approach to rule derivation is proposed by Awad et al. [89], with a predefined generic set of patterns concerning process and task instances. They relate to the occurrence of tasks, their ordering and resource assignments in a given process instance. Events are classified to types accordingly. A CEP engine monitors anti-patterns that indicate any violation of the rules derived from the process model.

In summary, the challenging areas of difference between process models and CEP rules generated on their basis are summarized in Table 5.

Table 5: Summary of challenging areas of difference between process models and CEP rules derived from them (Q3)

Issue	Process Model	Derived CEP Rules	Application to Example
Purpose supported	Conceptualization, communication, execution	Monitoring of process progress, compliance, or SLA; Process execution	The process model specifies control and message flow among responsible units and can be executes by an appropriate engine; A CEP rule can monitor time commitments for the process and alert when they are not met
Abstraction level	High level, business meaningful terms and task definitions	Low level, need to be correlated and aggregated to be business meaningful	Model: beginning and ending of the Drive activity; corresponding CEP rule: position and movement of Truck.
Generic vs. specific information	Generic (design time) information	Instance-specific details, available at runtime	Generic (model): "Pickup point"; Specific (CEP): based on specific instance data - address and coordinates of pickup point.
Main constructs	Activities, control-flow gateways, conditions, instances	Events, predicates, antecedent-consequent structures, sources	A gateway in a process model is transformed to a complex event: (TruckID sent AND ShipmentInfo passed) => then Record Transport Information can start.
Context awareness	Can be inferred as a requirement but not explicitly specified	Immediate consequence of the rules	Flight ongoing data as the context of the logistic process [101] - inferred from the process but enabled through CEP

6. Q4 – Executing Business Processes via CEP Rules

This section examines the idea of using CEP engines for executing business processes. In particular, we look for opportunities for advancing business process execution and note the challenges that are posed such an approach.

6.1. Motivation and Research Question

To explore the potential benefit of CEP engines for process execution, let us first examine the feasibility of this idea and what can be gained by it. We do so by focusing on a basic functionality of a process engine, that is managing the state of the process at any given moment. When a process is executed, each work item (namely, an instance of an activity in the process model) can be disabled (before its preconditions are fulfilled based on the process model), enabled (when according to the process model it is possible to start executing it), active, or completed [102]. As work items are completed, the state of the process changes and other work items become enabled. For human-based activities beginning and ending are typically indicated by the operator, enabling the engine to advance the state of the process. For tasks whose execution is continuous, involving non-human resources, and geographically distributed, this monitoring may pose difficulties. We shall consider this in

the context of the running example (see Section 2.3), focusing on two specific activities.

Let us first consider the activity of Monitor Transportation (cf. Figure 3), which receives updates during the Drive activity of the truck. At a business level, we have high-level events which are derivable from low-level input events (not the concerns of a process model) processed by a CEP engine. Assuming the monitor activity is implemented by a CEP engine, relevant high-level events are derived by the CEP engine from the respective low-level input events. *Driving started*: The truck is in the scope of the starting point, goods on the truck, the truck is moving out of the scope of the starting point. *Driving completed*: The truck is in the scope of the destination point, goods off the truck, the truck is moving out of the scope of the destination point. The clear advantage that is observed is the possibility to automatically detect and record the beginning and completion of the driving, as well as unexpected exceptions (e.g., traffic jam) without involvement of the driver. "Traditional"BPM systems react on simple event notifying on the start and completion of a task, whereas CEP engines are capable of inferring this information from a combination of lower-level inputs.

In contrast, let us consider the activity of Receive Transport Notifications (see Figure 3), performed by the logistics company. This is a multiple instance activity, meaning that a work item is created with every notification sent by the control system. We assume the company's activities are performed by human operators, thus receiving each notification instance is enabled as soon as it arrives, but only executed when the human operator attends to it (an event which makes the work item active, followed by an event marking completion of addressing the notification). In this example, the CEP engine does not seem to offer an advantage over a process engine because no complex reasoning to infer the completion of the activity is needed.

Following these two examples, in general terms, the research question is:

What aspects of process execution can benefit from the use of a CEP engine, and what aspects are better handled by traditional process engines?

6.2. Existing work related to Q4

To address this question systematically, we first outline the basic required capabilities of a business process engine [103]. These include (a) management of process instances (cases), (b) basic management of process state; (c) the dimensions of workflow patterns: control-flow, data, resource, and exceptions handling; and (d) the use of a process model. We analyze CEP-based process

execution systems that were found in our literature survey with respect to these issues. In addition, we characterize the motivations that drive CEP-based process execution proposals to identify possible strengths of such systems. In particular, these refer to distributed execution and collaboration, interaction with diverse environments, flexibility, and scalability [104, 105].

Management of process instances. A process engine should be able to create instances of a given process model, where each process instance has its own work items as instances of the process activities. Many instances can be executed in parallel (e.g., many shipments delivered) and controlled by the process engine. In contrast, CEP engines typically process a stream of events, regardless of process instances. A mechanism supporting the creation and management of process instances by a CEP engine is proposed Cicekli and Cicekli [106]. Their proposed process engine, which is based on event calculus, creates a process instance upon a defined external event (e.g., an order received). It creates a unique ID of the process instance, and this ID then marks all the activities related to the process instance. This way parallel instances of the same process can be executed and managed. Appel et al. [107] propose a middleware connecting CEP and process engines. While not directly targeting process execution by a CEP engine, they recognize the need to manage process instances and to "mark" each event by an ID relating it to a specific process instance. Another example, relating to declarative Guard-Stage-Milestone (GSM) models, is the CEP engine proposed by Jergler et al. [108] that creates process instances and monitors their state.

Basic management of process state. This refers to the ability to track the state of work items as demonstrated to the running example above. While this is a basic functionality of process engines, they need to be "notified" about the start and completion of work items, either by the human operators or by signals (for automated tasks). A CEP engine can infer start and completion of work items based on a combination of events, and is hence better suited for activities whose beginning or ending are not specifically marked. A possible approach is to specify events or rules that mark the beginning and the completion of an activity [106, 109]. Besides this, Appel et al. [107] also propose a similar marking for the completion of the entire process instance. Morales et al. [110] use control events and a coordinator function which monitors the execution of activities and the set of enabled activities.

Workflow patterns dimensions. The workflow patterns collection has been developed as a benchmark for workflow systems and process modeling formalisms. The four basic sets of patterns relate to four process dimensions: control flow patterns [111], data patterns [112], resource patterns [113], and exception handling patterns [114]. We consider each dimension separately.

- Control flow patterns: A large collection of structural patterns that may exist in a process model, that has served for numerous evaluation studies of modeling formalisms and workflow systems. While not all are supported by all process engines [115], they set a basic standard to be supported, including parallel and conclusive splits, merges and synchronizations, patterns of multiple instances, state based patterns, looping, and more. We are not aware of an exhaustive study of how these can be supported by CEP engines. Yet, specific patterns that have traditionally been challenging for process engines may be easier to implement with a CEP engine. For example, the non-local semantics of the OR join [111], which should synchronize all (and only) the active branches, can be represented as a complex event. Similarly, the milestone pattern, where tasks are enabled upon occurrence of a given event, is difficult to specify using process model control flow constructs, and easy to specify using event-based rules. Generally speaking, subsets of control flow patterns are supported by specific CEP engines. We discuss these in the context of the process models supported.

- Data patterns: The data patterns [112] address a variety of issues concerning how workflow systems handle data when executing processes and the roles played by data in these processes (e.g., task pre or post conditions are expressed as logical expressions over data). Although not specifically discussed in the literature, all these patterns are well supported by CEP engines.

- Resource patterns: The resource patterns [113] mainly address the distribution of process activities among human operators and the management of individual and group-related work queues. These are defined along the lifecycle of a work item, including its creation, offering, allocation, initiation, and various "detour" operations (e.g., delegation, escalation). Besides a small group of auto-start patterns, the majority of the resource patterns are not supported by most CEP engines. Yet, some examples exist, demonstrating that comparable CEP-based

operations can be defined using designated events denoting human resource behavior. Cicekli and Cicekli [106] support a relatively rich resource management functionality, including the assignment of activities to resources (either human or machine) and managing the state of resources (waiting or busy), as well as worklists with optimization rules for selecting a resource to be assigned for an activity. In summary, the feasibility of supporting at least some resource patterns by a CEP engine has been demonstrated. Although no real advantage of CEP engines for this matter has been indicated, a basic support of these patterns is of importance when seeking to gain other benefits of CEP engines.

- Exception handling patterns: The exception handling patterns [114] deal mainly with how an exception is handled once it is detected. These include actions that should be taken regarding the specific work item that is currently active (e.g., restart, fail), the entire case (e.g., remove case), the full set of running cases (e.g., remove all cases), and compensating actions to be taken. The detection of an exception is however limited to specific predefined exception types (e.g., time exception), but is not addressed otherwise. In contrast, CEP engines can detect any exception as a deviation from an expected pattern of behavior, with data coming from various sources. Yet, the actual actions to be taken should be defined. One example where action upon exception detection is addressed is presented by Morales et al. [110], offering a basic mechanism for restoring the system and recovering from exceptions. Being generic, it does not entail compensating actions of any kind.

Direct execution of process models. Process engines in essence serve for executing process models. Different kinds of models are used by different engines. In particular, the common process models (e.g., BPMN) follow an imperative paradigm, explicitly specifying the possible sequences of activities to be followed. Less commonly used are declarative models, specifying constraints over the possible actions and allowing flexible execution traces. Imperative models are rather intuitive to create and easily understandable by humans [116], while declarative models pose challenges to human understanding [117]. Hybrid process models have also been proposed, combining the clear and easy-to-follow specification of imperative models where flexibility

is not essential, and a flexible declarative model at parts that require this. Process engines usually support a specific model type (imperative, declarative, or hybrid). For employing a CEP engine for process execution, the human understandable process models need to be transformed into low level event specifications. Translating an imperative process model into event specifications has been addressed to a small extent. Li et al. [96] provide a translation of block-structured part of BPEL [118] to event specifications. BPEL itself is an execution language rather than an imperative process model [119], but mappings exist between BPMN and BPEL (despite the differences in their expressive power [120, 121]). Cicekli and Cicekli [106] use an imperative process specification called "control-flow graph", which supports very basic control flow patterns (sequence, choice, concurrency, and looping). They provide event rules for specifying and supporting this expressiveness. Hens et al. [109] use imperative models (BPMN, Yet Another Workflow Language (YAWL) [122]) and fragment them to small chunks, each chunk "wrapped" by events denoting their start/completion that can be processed by a CEP engine. This, however, cannot be considered a model transformation, as the imperative model fragments are still handled by process engines rather than by the CEP engine. Morales et al. [110] use a basic imperative process model mapped to a pub/sub network specification. Their model supports basic and advanced branching and merging control flow patterns. Transforming declarative models to event specifications has been suggested more often. Jergler et al. [108] suggest a CEP-based execution of the GSM model, which is a declarative model for specifying life cycle processes of business artifacts. The GSM model includes Event-Condition-Action (ECA) rules which are guarded by "sentries", which can be implemented as CEP engines. ECA-based execution of a declarative process model is also suggested by Soffer [123]. The specification of declarative process models (e.g., Dynamic Condition Response (DCR) graphs [124]) is often based on event rules. Yet, the execution engine discussed by Hildebrandt and Mukkamala [124] is not a CEP engine. Finally, hybrid models combine the declarative and imperative approaches to gain the benefits of both. For example, YAWL, which is basically imperative, allows using declarative services, as well as worklets. A worklet is a self-contained process snippet to implement a more abstract task in a process [125], triggered by rules which are evaluated at runtime. One could look at implementing the evaluation of such rules by means of CEP: based on the context and subject to change.

While many of the above discussed properties are basic features of process

Table 6: Summary of capabilities of process engines and CEP engines regarding the main issues discussed (Q4)

Issue	Process Engine	CEP Engine	Application to example
Management of process instances	Supported	Formal foundations exist based on event calculus; requires event correlation	Initiating, executing, and monitoring each delivery separately even if they are executed in parallel
Management of process state	Depending on activity start/completion indication	Enables deduction of start / completion based on event data	Manual indication: start/completion of handling transport notification Automatic: start/completion of Drive inferred from location and movement of truck
Control-flow patterns	Supported	Partial support of basic and advanced branching/merging patterns	Activities are enabled according to, e.g., (a) sequence – when Load Container is completed Drive is enabled; (b) in parallel – Activity list is recorded at the truck in parallel to the recording of the activity attributes at the monitoring system
Data patterns	Supported	Supported	Example patterns: (a) task post-condition (data value) – completion of Drive determined based on truck location and movement values; (b) task post-condition (data existence) - completion of Send Activity list determined by existence of activity list at the truck
Resource patterns	Supported	Partial support of offering, allocation, and initiation patterns, including specific kinds of optimization of task/resource assignment	An example pattern: Distribution by offer (multiple resources) – assuming several employees can handle transport notifications. With a CEP engine, the notification event is published to all possible resources. When one starts the task, this event results in removal of the task from the other work lists.
Exception handling patterns	Handling is supported, detection is very limited	Strong detection capabilities, handling is very limited	An exception: mechanical failure of truck during Drive. Detection: based on an event at the truck; handling: by rules (a) for the Drive activity: restart with another truck; (b) for the process instance: cancel, initiate another; (c) compensation: notifying customer about delay
Direct execution of model	Imperative, declarative, hybrid	Declarative models, partial mappings from imperative models exist	Execution of the imperative models in Figures 2 to 4 (by any engine)
Environment considerations & flexibility	Specific flexibility and context awareness solutions (adaptivity of imperative model or use of declarative one)	Built in ability to handle contextual events. Unexpected behavior can be detected without being specified in advance.	Subscription to event streams that provide information about traffic and road conditions in addition to rules for adapting the driving route and timing.
Distributed processes	Possible but require choreography solutions, partial interaction support	Naturally handled. CEP's capability of handling distributed event sources is a basic functionality	Manageing Drive as a sub-process executed by a local CEP engine, which addresses events related to the road and driving, and sends events to another CEP engine executing the main process. This enables immediate adaptation of driving, independent of, e.g., connectivity.

engines, other properties can be considered desirable for certain purposes, typically supported by CEP engines, thus motivating their use.

In particular, these refer to environment considerations, such as the need for distribution, flexibility, and interaction with the environment.

Environment considerations and flexibility. Some processes, specifically context-aware ones, need to interact with the environment during execution. However, process engines are limited to pre-specified interactions. To overcome this limitation, Schlegel et al. [104] as well as Hens et al. [109] propose a CEP-based integration framework to manage the interaction between distributed workflow systems, each managing processes locally. For process engines, context awareness is typically possible only for expected information types and values, and at predefined points in the process. In contrast, CEP engines can conveniently receive various, dynamic, and unpredictable events occurring at run-time from different sources of the environment. And, it can use CEP rules to analyze those events to obtain the high-level events which guide and adapt the process execution. For example, Hermosillo et al. [105] propose a framework which utilizes CEP engines to monitor the execution of business processes and allows business processes to be dynamically adapted to the changing environments. Notably, process engines and CEP engines are complementary in this framework: the former is responsible for the execution of business processes while the latter is employed to monitor process execution and trigger process adaptation according to user-defined CEP rules. Under this framework, a BPEL-extension is given to add flexibility to BPEL processes [126]. Jergler et al. [108] suggest a CEP engine to enact GSM-based processes with the intention to support geographically distributed execution of processes. In summary, CEP engines are more flexible than process engines, because CEP rules can be easily modified, removed, and added at runtime [105], while changing workflow models usually requires more effort [127].

Distributed processes. Process engines support process orchestrations, where while distributed cross-organizational services can be invoked, the whole process is under a single control. To implement fully distributed processes, i.e., process choreographies [128], usually several process engines are needed, and their interactions should be specified before execution [129]. Thus, a full realization of distributed processes by a process engine is not trivial [127]. In contrast, CEP engines that process events from different sources naturally fit distributed processes. For example, Schlegel et al. [104] utilize a CEP engine

to integrate inter-organizational process engines and applications. The merits of this platform are two-fold. First, the involved business processes can be decoupled, and thus there is no need for the corresponding process engines to be connected. Second, the involved business processes can be heterogeneous, that is, business processes implemented in different languages and orchestrated in different engines can be combined together for collaboration. Kong et al. [130] leverage event processing and ECA rules to realize real-time process integration in ubiquitous (distributed) enterprise environments. Similar approaches are described by Schlegel et al. [104], Hens et al. [109].

Two sets of patterns that relate to service oriented architectures, the Service interaction patterns Barros et al. [131] and the Correlation patterns Barros et al. [132] provide a systematic basis for analyzing the relevant capabilities. The correlation patterns take a high-level view, addressing (local) process instances and conversations, namely interactions or processes that span local processes. They list mechanisms for correlating events in an event stream to process instances and to conversations, supporting the execution of both local and distributed processes by CEP engines. The service interaction patterns relate to lower-level mechanisms of message passing among services, including four groups of patterns: (a) single-transmission bilateral interactions are simple and basic messaging operations, supported by both process and CEP engines; (b) single-transmission multilateral patterns involve multiple messaging parties (e.g., one-to-many send), naturally supported by pub/sub protocols, and mostly supported by multiple instance functionalities of process engines; (c) multi-transmission patterns (e.g., multi response to a message), which can be formulated as a complex event (for CEP), but are only supported to a small extent by process engines; (d) routing patterns, which relate to low-level message routing and are not in the scope of this discussion. In summary, the different levels of support for the interaction patterns indicate the superiority of CEP engines for distributed processes.

The above discussion is summarized in Table 6. It appears that for most of the relevant issues a clear advantage of one of the engine types is observed. As a conclusion, we may say that CEP and process engines can complement rather than substitute each other. The opportunity that arises is of using CEP and process engines in combination, where each is responsible for different aspects. This way the strengths of both can be realized.

7. Discussion

Following the discussion of each of the four quadrants separately, in this section we integrate the findings into an overall view, indicating the core challenges as well as opportunities that emerge. We also discuss some limitations of our current analysis and point to further aspects that were beyond the scope of this article.

7.1. Core challenges and opportunities

While each quadrant focuses on a different aspect, there are overarching concerns that recur across quadrants. Table 7 summarizes the main issues identified so far, indicating their relevance to each of the quadrants as well as whether they form a challenge (marked as C) or provide an opportunity (marked as O). In particular, most of the issues combine challenges and opportunities. Since the strengths of CEP and BPM are complementary, many opportunities emerge from the proposed combinations. These, however, can be realized once the indicated challenges are met. The issues included in the table are those that were found relevant to at least two quadrants. The analysis shown in Table 7 enables us to indicate the core issues associated with the combination of BPM and CEP—challenges which need to be addressed at a foundational level rather than specifically on a per-application basis. In addition, the table reveals issues which are relevant in the context of specific quadrants or application areas but not in others. Last, the table highlights opportunities that should motivate additional work in each quadrant or across this area. Below we discuss each of the identified issues.

Unify terminology and bridging abstraction levels is a core challenge for all quadrants, due to the inherent difference in abstraction levels that require mechanisms for overcoming them. It is required for obtaining meaningful process models by process mining (Q1), for incorporating CEP queries in process models (Q2), for deriving event-based rules from process models (Q3), and for interpreting and executing process models by CEP engines (Q4). Hence, this is a main core issue to be tackled across all mechanisms to be developed that integrate event-based and process-oriented systems. We nevertheless note that particularly for Q1 the CEP capability of combining raw events into higher-level complex ones is an opportunity that can help avoiding loaded spaghetti-like mined models.

Accounting for specifics of process instances is also identified as a challenge which is relevant for all the quadrants. While process models

Table 7: Key challenges (C) and opportunities (O) in the quadrant areas xxx

Key Issue	**Q1**	**Q2**	**Q3**	**Q4**
Unify terminology and bridging abstraction levels	O/C	C	C	C
Accounting for specifics of process instances	C	C	C	C
Understandability to humans	O/C	O/C	O	O/C
Flexibility and context awareness		O	O	O
Distributed execution		O	O	O
Monitoring of process progress		O	O	C
Unifying/Integrating engines for process execution		C		O/C
Unifying/Integrating engines for processing event data	O/C		O	
Transformation between process models and event rules		C	O/C	O/C

are specified at a generic level, they are instantiated upon execution. Event streams, which relate to runtime data, are associated to process instances, but this association is not necessarily explicit. Hence, when the starting point is a process model (Q2, Q3, and Q4), instance details need to be augmented at the event processing level. When the starting point is the event stream (Q1), events need to be correlated along the time line and associated with process instances to be abstracted to a process model. Note that as opposed to other issues, we do not indicate opportunities related to this one. Rather, we consider meeting the challenges related to this issue as essential for realizing opportunities related to all other issues.

Understandability to humans is a (non-technical) issue which, again, applies to all quadrants. Here we mainly see the combination of CEP and BPM as an opportunity, with the presence of a process model improving the understandability of event-based concepts and analysis, whose formality may pose cognitive difficulties to humans. Yet, some challenges in this respect still exist: in Q1 the challenge is to produce process models that are at a sufficiently high abstraction level to be meaningful and understandable (to avoid spaghetti models). In Q2 the challenge is to augment the event specifications into the process models without hampering their understandability. In Q4 the challenge is to be able to provide process models which are both executable by CEP and understandable by humans.

Flexibility and context awareness as well as the possibility of **dis-**

tributed execution are identified as opportunities in Q2, Q3, and Q4. While process models have often been criticized for their rigidity and lack of flexibility, and while context awareness and distributed process management have long been identified as challenge areas for BPM, these are inherent capabilities of CEP. Hence, in these respects CEP can enhance and improve the execution of business processes and their monitoring at runtime.

Monitoring of process progress is also identified as a relevant issue for Q2, Q3, and Q4. It is an opportunity that motivates works in Q2 and Q3 (which essentially deal with facilitating such monitoring). For process execution by a CEP engine this forms a challenge, as the execution requires an ongoing monitoring of the process state, and is not possible without it.

Unifying/Integrating engines for process execution has been identified as a challenge relevant for Q2 and Q4. Dealing with the possibility to delegate parts of the process (or all of it) for execution by a CEP engine. Concerning Q2, we see the challenge of finding ways to execute a process model which includes query oriented CEP specifications as well as "traditional" process representations. This should call for an engine or a combination of engines that would be capable of executing both. Regarding Q4, we see challenging questions, such as how to distribute process execution between a process engine and a CEP engine in order to gain the opportunity, which is combining the strengths of both.

Unifying/Integrating engines for processing event data is the "complementary" issue, concerning Q1 and Q3. For Q1, the main question is of how to enable a process mining engine to incorporate CEP for low level event abstraction. Alternatively, one may aim to find ways how a CEP engine can be leveraged to produce process models as output. CEP and process mining solutions coexists in today's software stacks and we see the better integration between them as an opportunity for facing this challenge, rather than developing separate solutions for each. Concerning Q3, while CEP solutions are capable of detecting patterns and monitoring them in a bottom-up manner, Q3 deals with complementing these by patterns derived from a process model in a top-down manner. An engine capable of addressing both directions in an integrated manner is the main opportunity that emerges.

Transformation between process models and event rules forms challenges for Q2, Q3, and Q4. Essentially, the ability of transforming process models to CEP rules is an enabler for the issues addressed in Q3 and Q4. Such transformation is required when a given process model is about to be executed by a CEP engine and when the logic of the model is to be

monitored by a CEP engine. As explained, event patterns can be discovered by CEP engines but are not necessarily the ones that are relevant in business terms (e.g., for monitoring compliance). The transformation is simpler for declarative process models, where the main issue is to overcome the differences in abstraction level and mark ending of completion of activities, and is more challenging for imperative process models, where the control flow needs to be fully transformed. It forms an opportunity for targeted and complete compliance monitoring.

In summary, it is clear that the issues discussed above are fundamental issues to the integration or combined use of event-based and BPM systems. Addressing each one at a foundational level, considering the different kinds of potential application would make a significant advancement of this area. In particular, progress is needed towards the establishment of standards and a common formalism, which should unify the terminology and support transformations between different abstraction levels. It should be generic and applicable for different application contexts and mechanisms.

7.2. Limitations

We note that further issues related to the combination of event-based and process-oriented systems may be considered, beyond the ones discussed here.

Examples include specification and definition of events, security issues in integrated applications, bottom-up identification of event patterns for monitoring, predictive monitoring of business processes, performance and efficiency of computation, and more.

The discussion of related work presented here does not aim to be a complete survey, but rather aims a providing a variety of examples that highlight issues in each of the quadrants. Each of these, however, address specific problems or specific applications. A generalization across the four quadrants allowed us to draw cross-quadrant conclusions and identify the core issues discussed above.

Furthermore, our paper purposefully identifies and discusses challenges and opportunities at a conceptual level. A detailed case study, which was beyond the scope of this paper, could provide further insights and explore selected technical aspects.

8. Conclusion

Event-based and process-oriented systems, as well as CEP and BPM, have traditionally been disjoint research areas, but in the past few years various

attempts used specific combinations for solving specific problems or for gaining specific benefits. This paper takes a broad view of possible combinations of these two areas, and maps them into four quadrants of a matrix, going from events to process models and back along the process life-cycle. We illustrate the challenges as well as the opportunities using a rich and detailed running example of a logistic process.

The main contribution of this paper is twofold. First, the four quadrant map forms a basis for a research agenda in this intersection of two areas. Second, from the generalization of issues—challenges and opportunities—in each quadrants we were able to indicate core issues as a research agenda. These issues bear cross-quadrant relevance; they should be addressed fundamentally, not in the context of a specific application and form a basis for future applications that would build upon this basis.

References

[1] J. Gubbi, R. Buyya, S. Marusic, P. M., Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (7) (2013) 1645–1660.

[2] L. Jiang, D.-Y. Liu, B. Yang, Smart home research, in: International Conference on Machine Learning and Cybernetics (ICMLC'04), IEEE, 659–663, 2004.

[3] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, A. Oliveira, Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation, in: The Future Internet Assembly (FIA'11), Springer, 431–446, 2011.

[4] H. Kargupta, Connected Cars: How Distributed Data Mining Is Changing the Next Generation of Vehicle Telematics Products, in: International Conference on Sensor Systems and Software (S-CUBE'12), Springer, 73–74, 2012.

[5] C. Cabanillas, A. Baumgrass, J. Mendling, P. Rogetzer, B. Bellovoda, Towards the Enhancement of Business Process Monitoring for Complex Logistics Chains, in: International Conference on Business Process Management (BPM'13), Workshops, Springer, 305–317, 2013.

[6] Object Management Group, Business Process Model and Notation (BPMN), Version 2.0, URL http://www.omg.org/spec/BPMN/2.0/, OMG Document Number: formal/2011-01-03, 2011.

[7] A.-W. Scheer, O. Thomas, O. Adam, Process Modeling using Event-Driven Process Chains, in: Process-Aware Information Systems, John Wiley & Sons, Inc., 119–145, 2005.

[8] W. Reisig, Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies, Springer, 2013.

[9] V. S. Kesaraju, F. W. Ciarallo, Integrated simulation combining process-driven and event-driven models, Journal of Simulation 6 (1) (2012) 9–20.

[10] R. von Ammon, T. Ertlmaier, O. Etzion, A. Kofman, T. Paulus, Integrating Complex Events for Collaborating and Dynamically Changing Business Processes, in: International Conference on Service-Oriented Computing (ICSOC/ServiceWave'09), Workshops, Springer, 370–384, 2010.

[11] J. Krumeich, B. Weis, D. Werth, P. Loos, Event-Driven Business Process Management: Where are we now? – A Comprehensive Synthesis and Analysis of Literature, Business Process Management Journal 20 (4) (2014) 615–633.

[12] D. Eyers, A. Gal, H.-A. Jacobsen, M. Weidlich, Seminar on Integrating Process-Oriented and Event-Based Systems, URL http://www.dagstuhl.de/16341, 2016.

[13] M. Weske, Business Process Management: Concepts, Languages, Architectures, Springer, 2nd edn., 2012.

[14] W. M. P. van der Aalst, K. M. van Hee, A. H. M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve, M. T. Wynn, Soundness of workflow nets: classification, decidability, and analysis, Formal Aspects of Computing 23 (3) (2010) 333–363.

[15] F. Leymann, D. Roller, Production Workflow – Concepts and Techniques, Prentice Hall PTR, 2000.

[16] W. van der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, et al., Process mining manifesto, in: International Conference on Business Process Management (BPM'11), Springer, 169–194, 2011.

[17] A. Bolt, M. Sepúlveda, Process Remaining Time Prediction Using Query Catalogs, in: International Conference on Business Process Management (BPM'13), Workshops, Springer, 54–65, 2014.

[18] M. Unuvar, G. T. Lakshmanan, Y. N. Doganata, Leveraging path information to generate predictions for parallel business processes, Knowledge and Information Systems 47 (2) (2016) 433–461.

[19] A. Hinze, K. Sachs, A. Buchmann, Event-based Applications and Enabling Technologies, in: International Conference on Distributed Event-Based Systems (DEBS'09), DEBS '09, ACM, 1–15, 2009.

[20] A. Margara, G. Cugola, G. Tamburrelli, Learning from the past: automated rule generation for complex event processing, in: International Conference on Distributed Event-Based Systems (DEBS'14), ACM, 47–58, 2014.

[21] O.-J. Lee, J. E. Jung, Sequence clustering-based automated rule generation for adaptive complex event processing, Future Generation Computer Systems 66 (2017) 100–109.

[22] J. Boubeta-Puig, G. Ortiz, I. Medina-Bulo, ModeL4CEP: Graphical domain-specific modeling languages for CEP domains and event patterns, Expert Systems with Applications 42 (21) (2015) 8095–8110.

[23] D. Jung, Specifying Single-user and Collaborative Profiles for Alerting Systems, Ph.D. thesis, The University of Waikato, 2009.

[24] A. Hinze, A. Voisard, EVA: An event algebra supporting complex event specification, Information Systems 48 (2015) 1–25.

[25] S. R. Jeffery, G. Alonso, M. J. Franklin, W. Hong, J. Widom, A Pipelined Framework for Online Cleaning of Sensor Data Streams, in: International Conference on Data Engineering (ICDE'06), IEEE, 140–151, 2006.

[26] S. R. Jeffery, M. Garofalakis, M. J. Franklin, Adaptive cleaning for RFID data streams, in: International Conference on Very Large Databases (VLDB'06), VLDB Endowment, 163–174, 2006.

[27] M. Song, W. Van Der Aalst, Supporting process mining by showing events at a glance (2007) 139–145.

[28] C. Floerkemeier, M. Lampe, RFID middleware design: addressing application requirements and RFID constraints, in: Joint Conference on Smart Objects and Ambient Intelligence (sOc-EUSAI'05), ACM, 219–224, 2005.

[29] A. Vijayaraghavan, D. Dornfeld, Automated energy monitoring of machine tools, CIRP Annals – Manufacturing Technology 59 (1) (2010) 21–24.

[30] P. Rosales, K. Oh, K. Kim, J.-Y. Jung, Leveraging business process management through complex event processing for RFID and sensor networks, in: International Conference on Computers & Industrial Engineering (CIE'10), IEEE, 1–6, 2010.

[31] Y. Fei, J. Hu, E. Hua, Z. Luo, RFID Middleware Event Processing Based on CEP, in: International Conference on e-Business Engineering (ICEBE'09), 481–486, 2009.

[32] C. Zang, Y. Fan, Complex Event Processing in Enterprise Information Systems Based on RFID, Enterp. Inf. Syst. 1 (1) (2007) 3–23.

[33] C. Bornhövd, T. Lin, S. Haller, J. Schaper, Integrating Automatic Data Acquisition with Business Processes Experiences with SAP's Auto-ID Infrastructure, in: International Conference on Very Large Databases (VLDB'04), Elsevier BV, 1182–1188, 2004.

[34] L. Dong, D. Wang, H. Sheng, Design of RFID Middleware Based on Complex Event Processing, in: Conference on Cybernetics and Intelligent Systems (CIS'06), 1–6, 2006.

[35] M. L. van Eck, N. Sidorova, W. M. P. van der Aalst, Enabling process mining on sensor data from smart products, in: International Conference on Research Challenges in Information Science (RCIS'16), IEEE, 1–12, 2016.

[36] A. Margara, G. Cugola, G. Tamburrelli, Learning from the past: automated rule generation for complex event processing, in: The 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, Mumbai, India, May 26-29, 2014, 47–58, doi:10.1145/2611286.2611289, URL http://doi.acm.org/10.1145/2611286.2611289, 2014.

[37] N. Tax, N. Sidorova, R. Haakma, W. M. P. van der Aalst, Event Abstraction for Process Mining using Supervised Learning Techniques, URL http://arxiv.org/abs/1606.07283, 2016.

[38] D. Schumm, F. Leymann, A. Streule, Process Viewing Patterns, in: International Enterprise Distributed Object Computing Conference (EDOC'10), IEEE Computer Society Press, 89–98, 2010.

[39] S. Smirnov, M. Weidlich, J. Mendling, Business Process Model Abstraction Based on Behavioral Profiles, in: International Conference Service-Oriented Computing (ICSOC'10), Springer, 1–16, 2010.

[40] A. Koschmider, E. Blanchard, Automatic User Assistance For Business Process Modeling, in: International Conference on Research Challenges in Information Science (RCIS'07), 445–454, 2007.

[41] S. Smirnov, H. A. Reijers, M. Weske, T. Nugteren, Business process model abstraction: a definition, catalog, and survey, Distributed and Parallel Databases 30 (1) (2012) 63–99.

[42] T. Baier, A. Rogge-Solti, J. Mendling, M. Weske, Matching of events and activities: an approach based on behavioral constraint satisfaction, in: Annual Symposium on Applied Computing (SAC'15), 1225–1230, 2015.

[43] T. Baier, C. Di Ciccio, J. Mendling, M. Weske, Matching of Events and Activities - An Approach Using Declarative Modeling Constraints, in: International Conference on Enterprise, Business-Process and Information Systems Modeling (BPMDS'15), 119–134, 2015.

[44] T. Baier, C. Di Ciccio, J. Mendling, M. Weske, Matching events and activities by integrating behavioral aspects and label analysis, Software & Systems Modeling (2017) 1–26.

[45] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, P. J. Toussaint, From Low-Level Events to Activities – A Pattern-Based Approach, in: International Conference on Business Process Management (BPM'16), Springer, 125–141, 2016.

[46] M. Stocker, M. Rönkkö, M. Kolehmainen, Abstractions from Sensor Data with Complex Event Processing and Machine Learning, in: International Congress on Environmental Modelling and Software (EMS'14), iEMSs, 1273–1280, 2014.

[47] T. Sztyler, J. Carmona, J. Völker, H. Stuckenschmidt, Self-tracking Reloaded: Applying Process Mining to Personalized Health Care from Labeled Sensor Data, T. Petri Nets and Other Models of Concurrency 11 (2016) 160–180.

[48] N. Tax, E. Alasgarov, N. Sidorova, R. Haakma, On Generation of Time-based Label Refinements, in: International Workshop on Concurrency, Specification and Programming, 25–36, 2016.

[49] B. Schwegmann, M. Matzner, C. Janiesch, preCEP: Facilitating Predictive Event-Driven Process Analytics, in: International Conference on Design Science at the Intersection of Physical and Virtual Design (DESRIST 2013), Springer, 448–455, 2013.

[50] F. Folino, M. Guarascio, L. Pontieri, Context-Aware Predictions on Business Processes: An Ensemble-Based Solution, in: On the Move to Meaningful Internet Systems (OTM'13), Springer, 215–229, 2013.

[51] F. Folino, M. Guarascio, L. Pontieri, Discovering Context-Aware Models for Predicting Business Process Performances, in: On the Move to Meaningful Internet Systems (OTM'12), Springer, 287–304, 2012.

[52] OASIS, Web Services Business Process Execution Language Version 2.0, URL http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007.

[53] O. Kopp, K. Görlach, D. Karastoyanova, F. Leymann, M. Reiter, D. Schumm, M. Sonntag, S. Strauch, T. Unger, M. Wieland, R. Khalaf, A Classification of BPEL Extensions, Journal of Systems Integration 2 (4) (2011) 3–28.

[54] A. Baumgraß, M. Botezatu, C. Di Ciccio, R. M. Dijkman, P. Grefen, M. Hewelt, J. Mendling, A. Meyer, S. Pourmirza, H. Völzer, Towards a Methodology for the Engineering of Event-Driven Process Applications, in: Conference on Business Process Management (BPM'16), Workshops, Springer, 501–514, 2015.

[55] M. Wieland, D. Martin, O. Kopp, F. Leymann, SOEDA: A Methodology for Specification and Implementation of Applications on a Service-Oriented Event-Driven Architecture, in: International Conference on Business Information Systems (BIS'09), Springer, 193–204, 2009.

[56] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Springer, ISBN 978-3-642-33143-5, 2013.

[57] J. Bae, H. Bae, S.-H. Kang, Y. Kim, Automatic control of workflow processes using ECA rules, IEEE Transactions on Knowledge and Data Engineering 16 (8) (2004) 1010–1023.

[58] R. Lu, S. Sadiq, A Survey of Comparative Business Process Modeling Approaches, in: International Conference on Business Information Systems (BIS'07), vol. 4439 of *LNCS*, Springer, 82–94, 2007.

[59] J. Recker, M. Indulska, M. Rosemen, P. Green, How good is BPMN really? Insights from theory and practise, in: European Conference on Informatoin Systems (ECIS), 2006.

[60] A. Barnawi, A. Awad, A. Elgammal, R. E. Shawi, A. Almalaise, S. Sakr, Runtime self-monitoring approach of business process compliance in cloud environments, Cluster Computing 18 (4) (2015) 1503–1526.

[61] M. Botezatu, H. Völzer, Language and MetaModel for Transport Processes and Snippets, GET Service Project, Report 4.1, URL http://getservice-project.eu/en/project/public-deliverables, 2015.

[62] A. Baumgraß, N. Herzberg, A. Meyer, M. Weske, BPMN extension for business process monitoring, in: Conference on Enterprise Modelling and Information Systems Architectures (EMISA'14), 85–98, 2014.

[63] K. Batoulis, A. Baumgrass, C. Di Ciccio, R. Eid-Sabbagh, M. Hewelt, A. Meyer, M. Pufahl, T. Wong, Automatic aggregation rule generation engine, GET Service Project, Report 6.4.1, URL http://getservice-project.eu/en/project/public-deliverables, 2015.

[64] M. Milanovic, D. Gasevic, Towards a Language for Rule-Enhanced Business Process Modeling, in: International Enterprise Distributed Object Computing Conference (EDOC'09), IEEE, 64–73, 2009.

[65] REWERSE Rule Markup Language, URL http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6, 2006.

[66] S. Mandal, M. Weidlich, M. Weske, Events in Business Process Implementation: Early Subscription and Event Buffering, in: BPM, to appear, 2017.

[67] L. J. R. Stroppi, O. Chiotti, P. D. Villarreal, Extending BPMN 2.0: Method and Tool Support, in: R. M. Dijkman, J. Hofstetter, J. Koehler (Eds.), Business Process Model and Notation - Third International Workshop (BPMN 2011), Springer, 59–73, 2011.

[68] J. Schimmelpfennig, D. Mayer, P. Walter, C. Seel, Involving Business Users in the Design of Complex Event Processing Systems., in: Conference on Database Systems for Business, Technology, and Web (BTW'11), 606–615, 2011.

[69] R. von Ammon, T. Ertlmaier, O. Etzion, A. Kofman, T. Paulus, Integrating Complex Events for Collaborating and Dynamically Changing Business Processes, in: International Conference on Service-Oriented Computing (ICSOC'09), Workshops, Springer Nature, 370–384, 2010.

[70] M. Wieland, O. Kopp, D. Nicklas, F. Leymann, Towards context-aware workflows, in: International Conference on Advanced Information Systems Engineering (CAiSE'07), Workshops and Doctoral Consortium, vol. 2, 25–39, 2007.

[71] M. Bauer, F. Dürr, J. Geiger, M. Grossmann, N. Hönle, J. Joswig, D. Nicklas, T. Schwarz, Information Management and Exchange in the Nexus Platform, Tech. Rep. 2004/04, University of Stuttgart, 2004.

[72] K. Vidackovic, Eine Methode zur Entwicklung dynamischer Geschäftsprozesse auf Basis von Ereignisverarbeitung (A method for the development of dynamic business processes based on event processing), Ph.D. thesis, Universität Stuttgart, 2014.

[73] EsperTech, Homepage for Esper framework, online at http://www.espertech.com/esper/, 2017.

[74] R. Seiger, S. Huber, T. Schlegel, PROtEUS: An Integrated System for Process Execution in Cyber-Physical Systems, in: Enterprise, Business-Process and Information Systems Modeling, Springer, 265–280, 2015.

[75] G. Decker, A. Grosskopf, A. Barros, A Graphical Notation for Modeling Complex Events in Business Processes, in: International Enterprise Distributed Object Computing Conference (EDOC'07), IEEE, 27–36, 2007.

[76] A. Barros, G. Decker, A. Grosskopf, Complex Events in Business Processes, in: Conference on Business Information Systems (BIS'07), Springer, 29–40, 2007.

[77] S. Kunz, T. Fickinger, J. Prescher, K. Spengler, Managing Complex Event Processes with Business Process Modeling Notation, in: International Workshop on Business Process Modeling Notation, Springer, 78–90, 2010.

[78] U. Breitenbücher, P. Hirmer, K. Képes, O. Kopp, F. Leymann, M. Wieland, A situation-aware workflow modelling extension, in: International Conference on Information Integration and Web-based Applications &Services (iiWAS'15), ACM, 1–7, 2015.

[79] K. Häussermann, C. Hubig, P. Levi, F. Leymann, O. Simoneit, M. Wieland, O. Zweigle, Understanding and designing situation-aware mobile and ubiquitous computing systems, in: International Conference on Mobile, Ubiquitous and Pervasive Computing (Ubicomp'10), WASET, 329–339, 2010.

[80] A. C. F. da Silva, P. Hirmer, M. Wieland, B. Mitschang, SitRS XT – Towards Near Real Time Situation Recognition, Journal of Information and Data Management 7 (1) (2016) 4–17.

[81] O. Zweigle, K. Häussermann, U.-P. Käppeler, P. Levi, Supervised learning algorithm for automatic adaption of situation templates using uncertain data, in: International Conference on Interaction Sciences Information Technology, Culture and Human (ICIS'09), ACM, 197–200, 2009.

[82] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Pearson Higher Education, 2004.

[83] D. Zimmer, R. Unland, On the semantics of complex events in active database management systems, in: International Conference on Data Engineering (ICDE'99), IEEE, 392–399, 1999.

[84] K. Görlach, F. Leymann, Dynamic Service Provisioning for the Cloud, in: International Conference on Services Computing (SSC'12), IEEE, 555–561, 2012.

[85] K. Vukojevic-Haupt, F. Haupt, F. Leymann, L. Reinfurt, Bootstrapping Complex Workflow Middleware Systems into the Cloud, in: International Conference on e-Science (e-Science'15), IEEE, 126–135, 2015.

[86] O. Gunalp, C. Escoffier, P. Lalanda, Rondo: A Tool Suite for Continuous Deployment in Dynamic Environments, in: International Conference on Services Computing (SCC'15), IEEE, 720–727, 2015.

[87] V. Muthusamy, H.-A. Jacobsen, T. Chau, A. Chan, P. Coulthard, SLA-driven business process management in SOA, in: Conference of the Center for Advanced Studies on Collaborative Research (CASCON'09), ACM, 86–100, 2009.

[88] E. Mulo, U. Zdun, S. Dustdar, Monitoring web service event trails for business compliance, in: International Conference on Service-Oriented Computing and Applications (SOCA'09), IEEE, 1–8, 2009.

[89] A. Awad, A. Barnawi, A. Elgammal, R. Elshawi, A. Almalaise, S. Sakr, Runtime detection of business process compliance violations, in: Annual Symposium on Applied Computing (SAC'15), ACM, 1203–1210, 2015.

[90] T. Chau, V. Muthusamy, H.-A. Jacobsen, E. Litani, A. Chan, P. Coulthard, Automating SLA modeling, in: Conference of the center

for advanced studies on collaborative research (CASCON'08), ACM, 126–143, 2008.

[91] R. Thullner, S. Rozanyai, J. Schiefer, H. Obweger, M. Suntinger, Proactive business process compliance monitoring with event-based systems, in: Enterprise Distributed Object Computing Conference Workshops (EDOCW), IEEE, 429–437, 2011.

[92] A. Metzger, P. Leitner, D. Ivanović, E. Schmieders, R. Franklin, M. Carro, S. Dustdar, K. Pohl, Comparing and combining predictive business process monitoring techniques, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (2) (2015) 276–290.

[93] F. Bry, M. Eckert, P.-L. Pătrânjan, I. Romanenko, Realizing Business Processes with ECA Rules: Benefits, Challenges, Limits, in: International Workshop on Principles and Practice of Semantic Web Reasoning, Springer, Springer Nature, 48–62, 2006.

[94] M. Sadoghi, M. Jergler, H.-A. Jacobsen, R. Hull, R. Vaculin, Safe Distribution and Parallel Execution of Data-Centric Workflows over the Publish/Subscribe Abstraction, Transactions on Knowledge and Data Engineering 27 (10) (2015) 2824–2838.

[95] V. Muthusamy, H.-A. Jacobsen, BPM in Cloud Architectures: Business Process Management with SLAs and Events, in: International Conference on Business Process Management (BPM'10), Springer Nature, 5–10, 2010.

[96] G. Li, V. Muthusamy, H.-A. Jacobsen, A distributed service-oriented architecture for business process execution, ACM Transactions on the Web 4 (1) (2010) 2:1–2:33.

[97] L. Baresi, G. Meroni, P. Plebani, Using the Guard-Stage-Milestone Notation for Monitoring BPMN-based Processes, in: Enterprise, Business-Process and Information Systems Modeling (BPMDS'16), Springer, 18–33, 2016.

[98] G. Meroni, C. Di Ciccio, J. Mendling, Artifact-driven Process Monitoring: Dynamically Binding Real-world Objects to Running Processes, in: International Conference on Advanced Information Systems Engineering (CAiSE) Forum, 105–112, 2017.

[99] M. Weidlich, H. Ziekow, J. Mendling, O. Günther, M. Weske, N. Desai, Event-Based Monitoring of Process Execution Violations, in: International Conference on Business Process Management (BPM'11), Springer, 182–198, 2011.

[100] C. Cabanillas, C. Di Ciccio, J. Mendling, A. Baumgrass, Predictive Task Monitoring for Business Processes, in: International Conference on Business Process Management (BPM'14, Springer Nature, 424–432, 2014.

[101] C. Di Ciccio, H. van der Aa, C. Cabanillas, J. Mendling, J. Prescher, Detecting flight trajectory anomalies and predicting diversions in freight transportation, Decision Support Systems 88 (2016) 1–17.

[102] O. Kopp, S. Henke, D. Karastoyanova, R. Khalaf, F. Leymann, M. Sonntag, T. Steinmetz, T. Unger, B. Wetzstein, An Event Model for WS-BPEL 2.0, Tech. Rep. 2011/07, University of Stuttgart, 2011.

[103] W. van der Aalst, K. van Hee, Workflow Management: Models, Methods, and Systems, The MIT Press, 2004.

[104] T. Schlegel, K. Vidačković, S. Dusch, R. Seiger, Management of interactive business processes in decentralized service infrastructures through event processing, Journal of King Saud University - Computer and Information Sciences 24 (2) (2012) 137–144.

[105] G. Hermosillo, L. Seinturier, L. Duchien, Using Complex Event Processing for Dynamic Business Process Adaptation, in: International Conference on Services Computing (SSC'10), IEEE, 466–473, 2010.

[106] N. Cicekli, I. Cicekli, Formalizing the specification and execution of workflows using the event calculus, Information Sciences 176 (15) (2006) 2227–2267.

[107] S. Appel, P. Kleber, S. Frischbier, T. Freudenreich, A. Buchmann, Modeling and execution of event stream processing in business processes, Information Systems 46 (2014) 140–156.

[108] M. Jergler, H.-A. Jacobsen, M. Sadoghi, R. Hull, R. Vaculin, Safe distribution and parallel execution of data-centric workflows over the

publish/subscribe abstraction, in: International Conference on Data Engineering (ICDE'16), IEEE, 2824–2838, 2016.

[109] P. Hens, M. Snoeck, G. Poels, M. D. Backer, Process fragmentation, distribution and execution using an event-based interaction scheme, Journal of Systems and Software 89 (2014) 170–192.

[110] A. Morales, T. Robles, R. Alcarria, E. Cedeño, On the Support of Scientific Workflows over Pub/Sub Brokers, Sensors 13 (8) (2013) 10954–10980.

[111] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, A. P. Barros, Workflow Patterns, Distributed and Parallel Databases 14 (1) (2003) 5–51.

[112] N. Russell, A. H. ter Hofstede, D. Edmond, W. M. van der Aalst, Workflow Data Patterns: Identification, Representation and Tool Support, in: International Conference on Conceptual Modeling (ER'05), 353–368, 2005.

[113] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, D. Edmond, Workflow Resource Patterns: Identification, Representation and Tool Support, in: International Conference on Advanced Information Systems Engineering (CAiSE'05), Springer, 216–232, 2005.

[114] N. Russell, W. van der Aalst, A. ter Hofstede, Workflow Exception Patterns, in: International Conference on Advanced Information Systems Engineering (CAiSE'06), Springer, 288–302, 2006.

[115] S. Harrer, C. R. Preißinger, G. Wirtz, BPEL Conformance in Open Source Engines: The Case of Static Analysis, in: International Conference on Service-Oriented Computing and Applications (SOCA'14), IEEE Computer Society, 33–40, 2014.

[116] D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, S. Zugal, Declarative versus Imperative Process Modeling Languages: The Issue of Understandability, in: International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD'09), Springer, 353–366, 2009.

[117] C. Haisjackl, I. Barba, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, B. Weber, Understanding Declare models: strategies, pitfalls, empirical results, Software & Systems Modeling 15 (2) (2014) 325–352.

[118] O. Kopp, D. Martin, D. Wutke, F. Leymann, The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages, Enterprise Modelling and Information Systems 4 (1) (2009) 3–13.

[119] F. Leymann, BPEL vs. BPMN 2.0: Should You Care?, in: International Workshop on Business Process Modeling Notation (BPMN'10), Springer, 8–13, 2010.

[120] M. Weidlich, G. Decker, A. Großkopf, M. Weske, BPEL to BPMN: The Myth of a Straight-Forward Mapping, in: International Conference on Cooperative Information Systems (CoopIS'08), Springer, 265–282, 2008.

[121] J. Vanhatalo, H. Völzer, J. Köhler, The Refined Process Structure Tree, Data Knowledge Engineering 68 (9) (2009) 793–818.

[122] W. M. P. van der Aalst, A. H. M. ter Hofstede, YAWL: yet another workflow language, Information Systems 30 (4) (2005) 245–275.

[123] P. Soffer, A State-Based Intention Driven Declarative Process Model, International Journal of Information System Modeling and Design 4 (2) (2013) 44–64.

[124] T. T. Hildebrandt, R. R. Mukkamala, Declarative Event-Based Workflow as Distributed Dynamic Condition Response Graphs, Electronic Proceedings in Theoretical Computer Science 69 (2011) 59–73.

[125] W. M. van der Aalst, M. Adams, A. H. ter Hofstede, M. Pesic, H. Schonenberg, Flexibility as a Service, in: International Conference on Database Systems for Advanced Applications (DASFAA'09), Springer, 319–333, 2009.

[126] G. Hermosillo, L. Seinturier, L. Duchien, Creating Context-Adaptive Business Processes, in: International Conference on Service-Oriented Computing (ICSOC'10), Springer Nature, 228–242, 2010.

[127] W. Song, H.-A. Jacobsen, Static and Dynamic Process Change, IEEE Transactions on Services Computing .

[128] G. Decker, O. Kopp, A. Barros, An Introduction to Service Choreographies, Information Technology 50 (2) (2008) 122–127.

[129] G. Decker, O. Kopp, F. Leymann, M. Weske, Interacting services: From specification to execution, Data & Knowledge Engineering 68 (10) (2009) 946–972.

[130] J. Kong, J.-Y. Jung, J. Park, Event-driven service coordination for business process integration in ubiquitous enterprises, Computers & Industrial Engineering 57 (1) (2009) 14–26.

[131] A. Barros, M. Dumas, A. H. M. ter Hofstede, Service Interaction Patterns, in: International Conference on Business Process Management (BPM'05), Springer, 302–318, 2005.

[132] A. Barros, G. Decker, M. Dumas, F. Weber, Correlation Patterns in Service-Oriented Architectures, in: International Conference on Fundamental Approaches to Software Engineering (FASE'07), Springer, 245–259, 2007.