Towards an Extended Metamodel of Event-Driven Process Chains to Model Complex Event Patterns

Julian Krumeich(✉), Nijat Mehdiyev, Dirk Werth, and Peter Loos

Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany
{julian.krumeich,nijat.mehdiyev, dirk.werth,peter.loos}@dfki.de

Abstract. This paper proposes an extension of the Event-driven Process Chain (EPC) metamodel in order to provide means to model complex event patterns within process models. There are some first attempts aiming to graphically depict such patterns; however, none of them focus EPC as a widely-used modeling language, especially in a business-related context. Thus, the paper first of all derives and defines typical complex event patterns and analyzes whether they are representable using standard EPC models. On this basis, a metamodel extension is conceived and additional modeling notations proposed. Finally, the notation is applied on two application examples.

Keywords: Complex event processing · Business process management · Event-driven business process management · Event-driven process chain · Metamodel

1 Introduction

Today, enterprises compete in a globalized world characterized by its constantly changing economic conditions. To be successful in this highly-competitive environment, enterprises are forced to react on threats and opportunities in a timely manner. In this regard, it is a mandatory task to continuously monitor and control business processes towards current business situations. With advancements in system integration and new technologies like the Internet of Things (IoT), real-time information availability, especially in manufacturing operations, has reached a new dimension [1]. This allows for in-depth insights into intra-organizational as well as cross-company business processes. Consequently, myriads of internal and external business events become visible forming increasingly big data [2].

To turn this enormous quantity of low level events (such as single sensor signals) into business value (like a discovery of machinery failures or breakdowns), it is crucial to filter and analyze event streams to detect meaningful patterns that indicate important situations with a decisive impact on the control and efficiency of business processes [3]. With Complex Event Processing (CEP) the required technology to detect such complex event patterns in real-time is already available. In this regard, CEP is considered to be an important driver to further advance the domain of business process management

(BPM) [4]. In the last years, this has motivated numerous research efforts coining the term Event-Driven Business Process Management (ED-BPM) [5].

However, considering existing research, it is still an obstacle to express complex event patterns within business process models in order to transparently illustrate their relation in terms of an event-driven business process control, i.e. describing reactions to complex event occurrences in business processes [6]. Thus, domain experts are struggling to communicate event patterns, which are crucial for business operations, with technical CEP experts and vice versa [6]. Thus, latest research dedicatedly call for integrated event modeling methods, i.e. to specify event patterns and their relations within process models [5]. Conventional, business process modeling languages like Business Process Modeling Notation (BPMN), Event-driven Process Chains (EPC) or Unified Modeling Language (UML) Activity Diagrams cannot express complex event patters such as sequential, temporal or spatial relations between single events [7]. Whereas a few first extensions exist for the BPMN, there is a particular lack regarding a dedicated support in EPC. EPC are a typical starting point for non-technical domain experts to model their associated business processes and are widely used in industry and research [8]. Since complex event patterns originate commonly from business scenarios for which non-technical experts are responsible for [6], EPC represent also a promising means for depicting these patterns in relation to business processes.

To address this research gap, the paper at hand first of all derives and examines characteristic event patterns considered in corresponding literature. Furthermore, the feasibility to model them in standard EPC models is evaluated and last but not least an extended EPC metamodel (incl. a modeling notation) proposed that allows for a comprehensive graphical depiction of the derived event pattern. This should eventually support business and domain experts to express event patterns in process models that can be later on transformed into executable rules consumed by CEP engines.

Hence, this paper applies a design-oriented research approach following the design science research guidelines proposed by [9]. In this regard, the EPC metamodel extension proposed in Sect. 4 represents the underlying design science artifact (*Guideline 1*). The relevance for constructing the underlying artifact and the relating research gap is pointed out in the introductory section as well as by analyzing complex event patterns and their support in EPC (cf. Sect. 3) (*Guideline 2*). To comply with *Guideline 3*, the paper exemplifies the application of the artefact by modeling two complex event patterns based on the extended metamodel in Sect. 5. Following the principle of design as an iterative process (*Guideline 6*), the metamodel extension in general builds upon previously proposed model constructs (cf. Sect. 4). *Guideline 5* was accomplished by outlining the applied research methodology in Sect. 1. Last but not least, the submission of this paper aims at fulfilling *Guideline 7*, the dissemination of research results.

2 Theoretical Foundation and Related Work

Complex Event Processing (CEP) has emerged as a novel event processing technology in addition to alternative approaches such as simple event processing and event stream processing. CEP systems enable to determine potential threats and recognize opportunities in real-time. Alongside being researched intensively as a specific research domain, successful industry applications of CEP can be observed in various fields such as manufacturing, logistic and supply chain processes, financial investment, military, traffic tracking, social sensing and so forth [10]. Integrating CEP with other technologies, namely Business Process Management (BPM), is a challenging task requiring domain knowledge in both areas. The primary purpose of such an integration lays in the potential usage of real-time information gained from distributed systems, services and sensor networks for monitoring, controlling and eventually optimizing business processes. In this regard, CEP enables to initiate new process instances, to stop running ones and to influence their behavior based on recognized event correlations stemming from massive streams of (sensor) data [5].

Event patterns represent templates which specify certain event combinations. They can be classified into various categories such as temporal, spatial, spatial-temporal, trend, modal and basic patterns where each category has various subcategories [11]. These patterns can be provided by experts or being automatically derived using machine learning algorithms. The actual detection of event patterns within continuously streaming data is realized through special predetermined queries. They are enabled by query processing techniques that are more evolved and differ from approaches applied to classical database analysis. To technically describe event patterns and rules dedicated Event Pattern Languages (EPL) are used. Yet there is no language standard, which results in diverse competing approaches [5], such as datastream-oriented languages building upon the Structured Query Language (SQL), production rules or rule-based languages applying Event-Condition-Action (ECA) principles originating from the Business Rule Management (BRM) as well as imperative script languages.

In the business process management domain, several process modeling languages have been proposed in the last two decades [12], such as BPMN, UML Activity Diagrams, EPC and many more. Their common goal is to graphically represent business processes in order to bridge the gap between business and IT perspectives. By providing a transparent view on business processes, process models can be used for optimization purposes and to have a blueprint for their technical implementation. As it is for example expressed by the name of the EPC, incorporating events within the flow of business processes is one key characteristic. However, despite of strong modelling abilities in terms of representing business processes, these languages commonly fall short in depicting complex event patterns as they are considered in the field of CEP [7].

This shortcoming again creates barriers between business and IT perspectives in terms of involving business users and process owners in CEP implementations [6]. Thus, as already pointed out in Sect. 1, methods to achieve an integrated event modeling within business process modeling are required. This could be a starting point for CEP experts to transform complex event patters from a conceptual level into precise and technical CEP specifications, which is more helpful than having them written down in a textual form [6]. Thus, representing complex event patterns in process modeling notations, is a crucial step to take to progress ED-BPM, which has already motivated some research activities in recent years. However, in most cases these contributions propose an extension to BPMN for modeling goals, key performance indicators, high-level events or even whole business situations [13–17]. One recently presented concept copes with the integration of dedicated event stream processing units within the BPMN representation of the overall business process [18]. In contrast, some first approaches explicitly seek a graphical solution for the specification of CEP event patterns [14–16]. Other concepts researched possible integration possibilities of business events [19] or even whole BAM artifacts [20] with e.g. process and/or information models [21].

3 Complex Event Patterns and Their Support in EPC

The paper at hand proposes a metamodel extension of EPC, which is a widely-used process modeling notation both in research and industry. Neither the standard EPC modeling notation [7], nor proposed extension [6], are able to depict event patterns that can be considered as complex ones. Even though simple event relations and hierarchies can for example be described with so-called event-diagrams, in which a complex event (such as "order checked") is decomposed into detailed events (such as "customer data checked" and "product data checked") [22], these aggregations should not be confused

Table 1. Complex event patterns and their support in standard EPC models

	Patterns	Definition	EPC support
Logical Patterns	L1: Event Conjunction	Two or more events have to have taken place in order to trigger complex event L1.	+
	L2: Event Disjunction	Alternative events have to have taken place. In a special case, the occurrence of events needs to be mutually exclusive.	+
	L3: Event Cardinality	One or more events have to have taken place a specified number of times. The number can be fixed, variable or defined by a range.	–
	L4: Event Sequence	Two or more events have to have occurred according to a specified sequence. **EPC support only by modelling them into a sequential flow, however this would contradict to EPC conventions (alternation of events and functions)*	–*
	L5: Event Exclusion	This event pattern is triggered, if one or more events have taken place while one or others have been absence. In this regard, both the occurring and the inhibiting events can be represented through a complex composition. ** EPC support only textual via event labeling.*	○*

(*Continued*)

Table 1. (*Continued*)

Temporal Patterns	T1: Event Time Relation	One or more events have to have occurred within or outside a given time window. Windows can be defined as fixed intervals (e.g. Mon 8am – Fr 5 pm) or event intervals (starting event – [ending event OR expiration time offset = 1 day]).	–
	T2: Absolute Time Relation	One or more events have to have occurred before or after an absolute point in time.	–
Spatial Patterns	S1: Fixed Location	One or more events have to have occurred at a fixed location.	–
	S2: Entity Distance Location	One or more events have to have occurred within a range around a certain location.	–
	S3: Event Distance Location	One or more events have to have occurred at or within a range around a certain event happing.	–
Trend	TP: Trend Pattern	Two or more events have to have occurred that satisfy a (non-)increasing, (non-)decreasing or stable pattern.	–
Data	DP: Data Dependency Pattern	One or more events have to have occurred whose data properties match certain data dependencies. * *EPC support only textual via event labeling.*	○*

Key: **+** is fully supported ○ is partly supported **–** is not supported

with what is understand with complex event patterns. Although EPC models provide several modeling elements to link events and activities, including logical operators, there is e.g. no means to express temporal or spatial related correlations [23].

Thus, the first step towards an extended EPC metamodel, which is capable to depict complex event patterns, is to identify characteristic event patterns from literature. These patterns of common event correlation are depicted in Table 1. The patterns are mainly derived from [23] which already synthesized a survey on complex events of possible business scenarios as well as a survey on common features of EPL languages. In addition, patterns considered by [11, 12] are added resulting in twelve patterns that also partly contain sub patterns.

Patterns specifically dealing with technical characteristics of CEP engines, such as subscription and consumption time, are omitted from this combination as they will not be of interest to domain experts (cf. Sect. 5 on limitations).

4 EPC Metamodel Extension and Modeling Notation Proposal

In order to provide a possibility to model the event patterns illustrated in Table 1, the standard EPC metamodel has been extended. As a foundation, the standard EPC metamodel proposed by [24] was chosen, which is represented using a UML Class Diagram (cf. Fig. 1, grey elements). Their metamodel is one of the frequently quoted ones and includes all required rudimentary elements [8]. To reduce complexity, elements detailing the element “Function” are omitted as the conceived extension deals with a dedicated “Event” specification; “Functions” are not the focus of interest. In general, EPC models contain at least one “Function” and two “Events”, which are reusable in other EPC models. In doing so, each function has one successor and one predecessor “Event” element. Thus, in contrast to “Events”, “Functions” are connected with exactly two “Control Flow Connectors”, which again can be connected with “Logical Operators”.

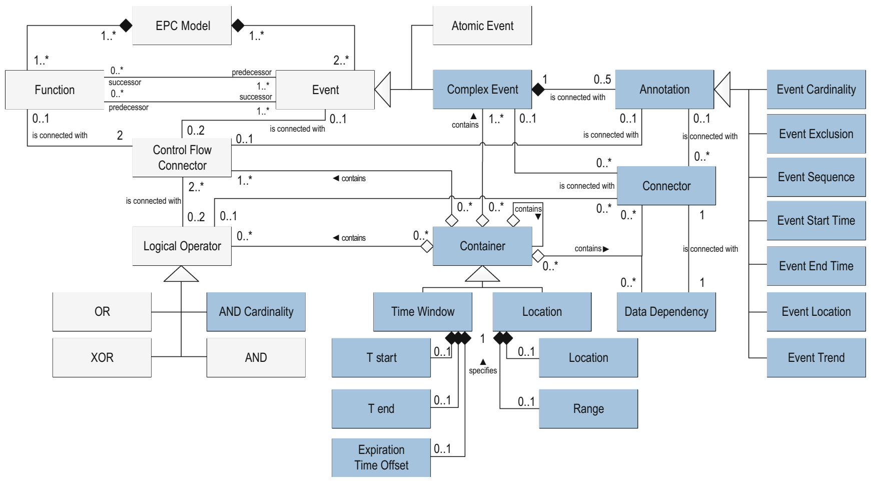
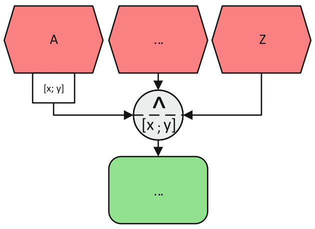
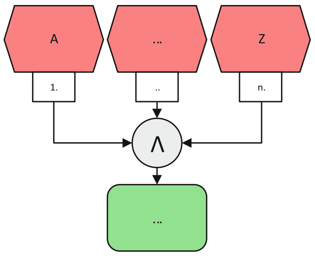
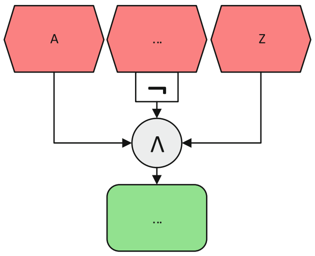
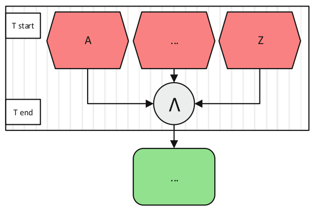
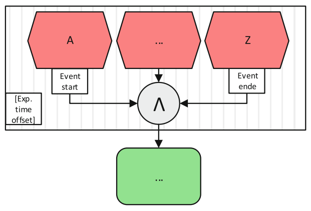


Fig. 1. Extended metamodel of EPC to represent complex events (based on [24])

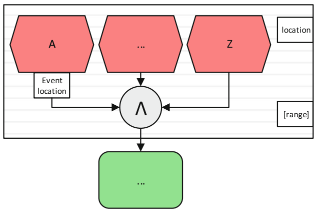
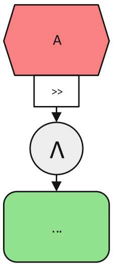
As pointed out in Table 1—with few exceptions (L1 and L2)—none of the derived event patterns can be represented in standard EPC models. Thus, additional types of modeling element are required (cf. Fig. 1, blue elements). First of all, the generic element “Event” is split into atomic events and complex ones. The latter can be connected with several “Annotations” that will be detailed introduced in Table 2. Furthermore, the metamodel comprises an “AND Cardinality” operator to define certain cardinality restrictions to complex event patterns. Moreover, to express additional constraints in relation to event data, dedicated “Data Dependencies” can be mapped via undirected “Connectors” to “Annotations”, “Logical Operators” as well as “Complex Events”. In order to specifically represent temporal as well as spatial relations of

Table 2. Description of metamodel extension and proposal of modeling notation

	Relation to Metamodel	Modeling Notation
L3: Event Card.	To express event cardinalities in EPC models, "Events" are connected with "Event Cardinality" annotations. These "Annotations" are labeled with the number respectively range of cardinality in square brackets. In case, a complex event pattern needs to have occurred a certain amount of time, the "AND Cardinality" operator is to be used.	
L4: Event Sequence	A specific sequential occurrence of "Events" can be expressed by "Event Sequence" annotations whose order is directly connected to the respective events.	
L5: Event Exclusion	To indicate the necessity of absence of specific events, these events can be connected with "Event Exclusion" annotations.	
T1 & T2: Temporal Patterns	To express temporal relations between "Events" they are combined into "Time Windows" acting as a container. To indicate time intervals, the container is specified by a start time "T start" and an associated end time "T end" (T1). To represent event intervals (T2), an "Event" is allocated as a start event, regarding the time window, and another one as the end event. Optionally, an "Expiration Time Offset" can be added. To visually differentiate between "Time Windows" and "Location" containers, the first one is illustrated with vertical lines as a background pattern symbolizing the temporal process flow in accordance to standard modeling conventions. Furthermore, the detailing property area is modeled on the left side of the container.	  A ... Z Event start Event ende [Exp. time offset] ∧ ...

(*Continued*)

Table 2. (*Continued*)

S1 – S3: Spatial Patterns	In accordance with temporal relations, spatial ones are also expressed by mapping the respective events in a "Location" container. To specify the location at which the event pattern has to have occurred, the container is detailed with a "Location" property (cf. pattern S1). In order to indicate a location range, the container is additionally specified by a "Range" property (cf. pattern S2). To satisfy pattern S3, an event can be allocated by an "Event Location" event in contrast to specify a certain location a priori.	 A ... Z location Event location Λ [range] ...
TP1: Trend Pattern	To define relations between events in terms of a trend pattern, the concerning event whose trend value is under consideration is annotated by "Event Trend" symbols: >> ≙ increasing; << ≙ decreasing; ¬ >> ≙ non-increasing; ¬ << ≙ non-decreasing. Generally, specifying trend patterns is only reasonable if combined with a temporal relation.	

complex events, a dedicated "Container" element is established, which comprises several complex events as well as other concepts. Furthermore, a "Container" itself can be a part of another one and is further specified by "Time Windows" or "Locations" as well as their linked properties.

As an inherent characteristic of "Complex Events", the proposed model elements can be jointly combined in multiple, complex ways (cf. Sect. 5 for examples).

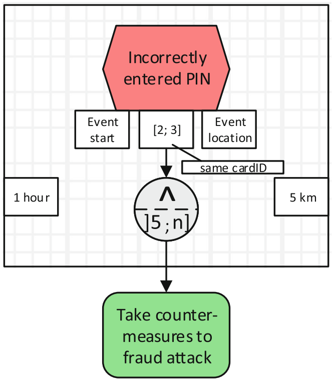
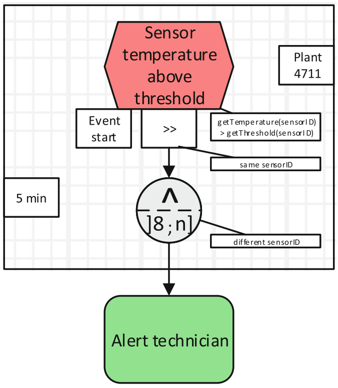
Table 2 explains, how the identified event patterns introduced in Table 1 can be represented using the extended EPC metamodel (excluding L1 and L2 as they are already supported in EPCs) and proposes a corresponding modeling notation.

5 Application Example, Discussion and Limitations

This section illustrates how the previously introduced event patterns can be combined in order to graphically represent complex event patterns that are defined by textual descriptions.

The first example stems from fraud detection, which is a typical application scenario for CEP. As conventional business process engines typically only consider individual events, i.e. "it is checked on a per-message basis if a message matches a registered subscription and only one message is consumed in a *receive* or *onMessage* activity”, CEP engines dedicatedly consider event rules, i.e. patterns of events that have to be matched and which may stem from different process instances across time [12]. In this regard, the fraud pattern defines four different conditions that need to be fulfilled in order to assume a case of fraud (cf. Table 3a). To represent this event pattern using the extended EPC metamodel, the event “Incorrectly entered PIN” is connected with cardinality restrictions—in terms of the same cardID as a “Data Dependency” annotated to the event annotation and as a combination of several of these complex events via an “AND Cardinality” operator—and embedded into a combined “Time Window” and “Location” container. In doing so, spatial and temporal relations are represented in an overlapping container, which is also visually represented by a grid pattern for the background of this container (cf. Table 2 regarding background patterns).

Table 3. Examples of complex event patterns and their representation in extended EPC models

Example	**a) ATM Fraud Detection**	**b) Production Line Malfunction**
Pattern Description	• at least wrongly entered PIN twice • event occurred more than five times • within a range of 5 km • within 1 hour	• sensor temperature above threshold • increasing trend • for more than 8 sensors • within 5 minutes • at plant 4711
Modeling Example	 Incorrectly entered PIN Event start [2; 3] Event location same cardID 1 hour ∧]5 ; n] 5 km Take counter-measures to fraud attack	 Sensor temperature above threshold Plant 4711 Event start >> getTemperature(sensorID) > getThreshold(sensorID) same sensorID 5 min ∧]8 ; n] different sensorID Alert technician

As the second example, continuously occurring events from temperature sensors attached to a production line are examined to detect a specified malfunction pattern (cf. Table 3b). In this regard, the event “Sensor temperature above threshold” is connected with an “Event Trend” annotation that is further specified by a “Data Dependency” expressing a trend in increased temperature values. Whereas this event annotation focuses a specific sensor instance defined by a sensorID, the constraint of having more than eight sensors matching this pattern is expressed by an “AND Cardinality” operator connected with a corresponding “Data Dependency”. Moreover, this complex event is embedded into an overlapping “Time Window” and “Location” container.

At this point, it should be explicitly pointed out that the proposed metamodel extension is not as expressive as existing CEP languages (cf. Sect. 2). Especially technical issues in terms of CEP implementations are omitted. Nevertheless, there is no claim to have a graphical modeling representation on the same level of detail as technical implementations, neither for business process representations and thus nor for complex event pattern representations in process models. Further, the metamodel does not differentiate the specific type of events (e.g. message, timeout or exception as in BPMN); yet, this can still be expressed in textual way using an associated labeling.

6 Conclusion and Future Work

This paper proposed an extension of the EPC metamodel in order to provide a means to model complex event patterns within process models. In literature, there are some first attempts aiming to graphically depict such patterns; however, none of them focus EPC as a widely-used modeling language, especially in a business-related context. To do so, the paper first of all derived and defined typical complex event patterns and analyzed whether they are representable using standard EPC models. On this basis, a metamodel extension is conceived and additional modeling notations proposed. Finally, the notation is applied on two application examples.

In future work, a set-theoretic definition [8] of the extended metamodel will be conceived to achieve a formal definition of complex event patterns within EPC models. Furthermore, in ongoing research activities, the representation of real-world complex event patterns in the context of the German research project iPRODICT is expedited in order to proof the feasibility of the proposed extension beyond the two application scenarios outlined in Sect. 5 (cf. [25] for more details on the research project). In this regard, the involvement of domain and technical experts in the modeling and usage of extended EPC models will also be evaluated regarding its actual usefulness and understandability. Since EPC models get more expressive through the extension, two views could be provided in a tooling context. One expressive view that visualize complex events in full detail and another one of reduced complexity through graphically differentiating between the model elements “Atomic Event” and “Complex Event”. The Complex Event then encapsulates all dedicated extensions such as annotations or containers. In this regard, we are also currently investigating how to integrate the metamodel extension into the ARIS Business Process Analysis Platform.

Another interesting research question will be how to induce complex event patterns out of event stream instances and how to visualize them either using the metamodel proposed in the paper at hand or in others proposed in literature (cf. Sect. 2). Future work will also deal with the transformation of extended EPC models respectively modeled event patterns into templates usable within executable Event Processing Languages (EPL). In this context, we are currently investigating how to build transformation algorithms for the Apama Complex Event Processing Engine. Since the patterns specified in EPC model cannot and should not be as expressive as common EPLs, the respective transformation will result in a first blueprint that has to be detailed and complemented by CEP experts. Yet the main goal of modeling complex events in

EPC is warranted as domain knowledge by business experts can be depicted in process models that can then be reused by CEP experts.

Acknowledgements. This research was funded in part by the German Federal Ministry of Education and Research under grant numbers 01IS12050 (project IDENTIFY) and 01IS14004A (project iPRODICT).

References

1. Bruns, R., Dunkel, J.: Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. Springer, Berlin (2010)
2. Dhar, V., Jarke, M., Laartz, J.: Big data. Bus. Inf. Syst. Eng. **6**, 257–259 (2014)
3. Luckham, D.: Event Processing for Business. John Wiley & Sons, Hoboken (2012)
4. Dixon, J., Jones, T.: Hype Cycle for Business Process Management. https://www.gartner.com/doc/1751119
5. Krumeich, J., Weis, B., Werth, D., Loos, P.: Event-driven business process management: where are we now? Bus. Process Manag. J. **20**, 615–633 (2014)
6. Schimmelpfennig, J., Mayer, D., Walter, P., Seel, C.: Involving business users in the design of complex event processing systems. In: BTW 2011. LNI, vol. 180, pp. 606–615 (2011)
7. Vidackovic, K.: A method for the development of dynamic business processes based on event processing (PhD thesis). Fraunhofer IAO, Stuttgart (2014)
8. Houy, C., Fettke, P., Loos, P.: Zur Evolution der Ereignisgesteuerten Prozesskette. In: Multikonferenz Wirtschaftsinformatik 2014, pp. 1020–1033 (2014)
9. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems reesearch. MIS Q. **28**, 75–105 (2004)
10. Aggarwal, C.: An introduction to sensor data analytics. In: Aggarwal, C. (ed.) Managing and Mining Sensor Data, pp. 1–8. Springer, Berlin (2013)
11. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications, Greenwich (2011)
12. Barros, A., Decker, G., Grosskopf, A.: Complex events in business processes. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 29–40. Springer, Heidelberg (2007)
13. Baumgrass, A., Herzberg, N., Meyer, A., Weske, M.: BPMN extension for business process monitoring. In: EMISA 2014. LNI, vol. 234, pp. 85–98 (2014)
14. Decker, G., Grosskopf, A., Barros, A.: A graphical notation for modeling complex events in business processes. In: 11th IEEE International Enterprise Distributed Object Computing Conference, pp. 27–36. IEEE Press, New York (2007)
15. Estruch, A., Heredia Álvaro, J.A.: Event-driven manufacturing process management approach. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 120–133. Springer, Heidelberg (2012)
16. Kunz, S., Fickinger, T., Prescher, J., Spengler, K.: Managing complex event processes with business process modeling notation. In: Mendling, J., Weidlich, M., Weske, M. (eds.) BPMN 2010. LNBIP, vol. 67, pp. 78–90. Springer, Heidelberg (2010)
17. Koetter, F., Kochanowski, M.: A model-driven approach for event-based business process monitoring. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 378–389. Springer, Heidelberg (2013)
18. Appel, S., Frischbier, S., Freudenreich, T., Buchmann, A.: Event stream processing units in business processes. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 187–202. Springer, Heidelberg (2013)

19. Döhring, M., Karg, L., Godehardt, E., Zimmermann, B.: The convergence of workflows, business rules and complex events. In: 12th International Conference on Enterprise Information Systems, pp. 338–343 (2010)
20. Friedenstab, J.P., Janiesch, C., Matzner, M., Müller, O.: Extending BPMN for business activity monitoring. In: Proceedings of the 45th Hawaii International Conference on System Sciences, pp. 4158–4167. IEEE Computer Society, Washington D.C. (2011)
21. Mulo, E., Zdun, U., Dustdar, S.: Domain-specific language for event-based compliance monitoring in process-driven SOAs. SOCA **7**, 59–73 (2013)
22. Loos, P., Allweyer, T.: Process orientation and object-orientation. In: Publications of the Institute for Information Systems, Paper 144. Saarland University, Saarbrücken (1998)
23. Kim, H., Oussena, S.: A case study on modeling of complex event processing in enterprise architecture. In: 14th International Conference on Enterprise Information Systems, pp. 173–180 (2012)
24. Korherr, B., List, B.: A UML 2 profile for event-driven process chains. In: Tjoa, A.M., Xu, L., Chaudhry, S.S. (eds.) Research and Practical Issues of Enterprise Information Systems 2006. IFIP, vol. 205, pp. 161–172. Springer, Heidelberg (2006)
25. Krumeich, J., Werth, D., Loos, P., Schimmelpfennig, J., Jacobi, S.: Advanced planning and control of manufacturing processes in steel industry through big data analytics: case study and architecture proposal. In: IEEE International Conference on Big Data, pp. 16–24 (2014)