Data-driven Generation of Services for IoT-based Online Activity Detection

Ronny Seiger, Marco Franceschetti, and Barbara Weber

Institute of Computer Science, University of St.Gallen, St.Gallen, Switzerland
firstname.lastname@unisg.ch

Abstract. Business process management (BPM) technologies are increasingly adopted in the Internet of Things (IoT) to analyze processes executed in the physical world. Process mining is a mature discipline for analyzing business process executions from digital traces recorded by information systems. In typical IoT environments there is no central information system available to create homogeneous execution traces. Instead, many distributed devices including sensors and actuators produce low-level IoT data related to their operations, interactions and surroundings. We leverage this data to monitor the execution of activities and to create events suitable for process mining. We propose a framework to generate activity detection services from IoT data and a software architecture to execute these services. Our proof-of-concept implementation is based on an extensible complex event processing platform enabling the online detection of activities from IoT data. We use a running example from smart manufacturing to showcase the framework.

Keywords: Internet of Things · Business Process Management · Activity Detection Service · Complex Event Processing · Process Mining

1 Introduction

The Internet of Things (IoT) emerged as new paradigm to foster the interaction of software systems with the physical world through connected devices and objects [3]. Thereby, sensors act as new data sources providing real-time information about the execution of activities, interactions and states of devices and objects and their surroundings. In a Business Process Management (BPM) context, a Process-aware Information System (PAIS) is used to execute and monitor process and activity executions, creating digital traces in *event logs* that can be used for *process mining* [21]. In IoT, there is no central PAIS available to create event logs suitable for process mining [12,4]. Instead, we are faced with a heterogeneous set of IoT devices that are controlled by different software applications providing data on different levels of abstraction [4,5,3]. In this work we present a data-driven framework for generating activity detection services from IoT data. From an *Activity Signature*, which represents the sensor readings associated with an activity, we generate services that are able to detect the occurrence of this activity using Complex Event Processing (CEP). These services are deployed to

a CEP platform to generate process and activity-related events from IoT data. These high-level events serve as basis for process mining. We demonstrate and discuss the framework as a proof-of-concept from service generation to deployment and detection using examples from smart manufacturing. The approach promises to be generalizable to less automated IoT domains for analysis of manual activities, which are tracked through IoT devices (e.g., in smart healthcare). The contribution of this paper is 1) a framework for transforming IoT sensor readings into CEP-based activity detection services; and 2) an extensible software architecture enabling the use and composition of these services at runtime.

Section 2 recalls fundamentals and requirements. Sect. 3 presents the framework and software architecture. Sect. 4 discusses our approach. Sect. 5 elaborates on related work. Sect. 6 concludes the paper and presents future work.

2 Fundamentals and Requirements

Activities and Events: An *activity* is considered to be an atomic unit of work in a business process, which is executed by a (non-)human process performer [21]. *Events* are used to capture the occurrence of something of relevance that has happened in the process execution (e.g., the start of an activity). We refer to these events as *process-level events*. In contrast, events are also used to capture the state of IoT components and their surroundings *sensed* at a certain point in time [3]. We denote these as *low-level IoT events*. Moving between these levels requires event aggregation, event transformation, and event correlation [20,7].

IoT Setup: In this work, we use a smart factory model as a typical IoT environment [18]. The model simulates the execution of production activities in discrete manufacturing processes. It features different IoT components (production machines) that act as performers of process activities. Each IoT component is equipped with sensors $i_1, .., i_n$, motors $m_1, .., m_m$, output devices $o_1, .., o_o$, and machine-specific sensors (e.g., representing positions) [18]. Real-time access to the time-stamped values of these sensors and actuators (low-level IoT events) is enabled via a message broker [19]. A PAIS to record event logs is not available.

Requirements: We follow design science [16] in developing the activity detection framework. The following non-exhaustive list of non-functional requirements was derived based on the authors' experience in the field of BPM and IoT and on relevant characteristics of IoT/Cyber-physical Systems (CPS).

- **R1** **Non-invasiveness**: The approach should work non-invasively with existing IoT systems assuming they provide at least one interface to emit low-level events from their IoT components. It shall not be necessary to introduce a heavy-weight software component (e.g, a PAIS) to the IoT environment.
- **R2** **No-code services**: Activity detection services shall be added to IoT environments at runtime to be used by other components. These services shall require no implementation effort for service providers, enabling domain experts to create and deploy services for new types of activities (*extensibility*).

R3 Runtime capabilities: Components in IoT and CPS are interacting with the physical world when executing activities. Here activity detection is crucial to provide (near) real-time feedback [14]. Activity detection must work at runtime, possibly already considering partial activity occurrences.

R4 Robustness: IoT components interact with the physical world. They are subject to various (context) factors [14], which influence the activity executions regarding execution times and involved sensors/actuators. The activity detection shall be robust to a certain degree against these variations.

3 Generation of Activity Detection Services

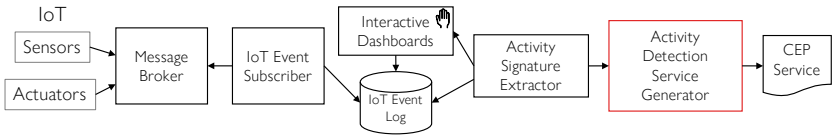

Fig. 1. Framework for generation of activity detection services from IoT data

Fig. 1 presents the framework for generating activity detection services from IoT data. Low-level IoT events are published via a Message Broker. They are retrieved by an IoT Event Subscriber and persisted in an IoT Event Log. We rely on the domain expert applying an interactive method to identify and label activity executions observed in the IoT data from the IoT Event Log [18].

Activity Signatures: In [18] we introduce *Activity Signatures* to represent the IoT data associated with the execution of an activity. We assume the domain expert to identify **one** representative instance for each type of activity in the IoT data–denoted as *Activity Prototype*–from which the signature is extracted. Table 1 contains the low-level IoT events for all sampled timestamps $t_0..t_{12}$ associated with the prototype of the *Burn* activity executed by *Oven*. The domain expert identified timestamps t_0 as start of this activity and t_{12} as its end.

Table 1. Activity signature for activity prototype of *Burn*

Sensor/Actuator	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
i1_pos_switch	0	0	1	1	1	1	1	1	1	0	0	0	0
i2_pos_switch	0	0	0	0	0	0	0	0	0	0	0	1	0
i5_light_barrier	1	1	1	1	1	1	1	1	1	1	1	1	1
m1_speed	0	-512	0	0	0	0	0	0	0	512	512	0	0
o7_valve	0	512	512	512	0	0	0	0	0	512	512	512	0
o8_compressor	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2. Changes in the activity signature for Burn

	Timestamps (t_i, t_{i+1})	Changes
1	(t_0, t_1)	m1_speed: 0 → -512; o7_valve: 0 → 512
2	(t_1, t_2)	i1_pos_switch: 0 → 1; m1_speed: -512 → 0
3	(t_3, t_4)	o7_valve: 512 → 0
4	(t_8, t_9)	i1_pos_switch: 1 → 0; m1_speed: 0 → 512; o7_valve: 0 → 512
5	(t_{10}, t_{11})	i2_pos_switch: 0 → 1; m1_speed: 512 → 0
6	(t_{11}, t_{12})	i2_pos_switch: 1 → 0; o7_valve: 512 → 0

Listing 1. Query for change detection in the event sequences for timestamps (t_0,t_1)

```
@info(name="Detect-LowLevel-Pattern-1")                                    1
from every e1 = OV_1Stream, e2 = OV_1Stream[(e1.m1_speed==0 and            2
    e2.m1_speed==-512) and (e1.o7_valve==0 and e2.o7_valve==512)]
select "LowLevelPat-1" as name, "burn" as activity, "(t0,t1)" as time      3
insert into DetectedLowLevelPatterns;                                      4
```

Service Generation: The signature of an activity prototype including the activity-related data (start and end time, label) and low-level IoT events are retrieved from the interactive dashboards and from the IoT event log by an *Activity Signature Extractor*. This component forwards the signature into the core component of the framework, the *Activity Detection Service Generator*. We have decided to use CEP as the basic technology here as it is designed for processing high amounts of events in multiple streams in online settings (cf. **R3**) [8,6]. Typical CEP platforms feature an event processing language (EPL), which simplifies the development of CEP-based applications–*CEP apps* (cf. **R2**). We generate a CEP app for activity detection based on a given activity signature as follows.

1) Change Detection: For each pair of consecutive timestamps (t_i, t_{i+1}) the changes among the values of all sensors and actuators are derived. Table 2 shows the result for the signature of the activity prototype for *Burn*.

2) Change Translation: For each pair of consecutive timestamps, the change in the event attributes is translated into the syntax of the EPL for the *Event Sequence Pattern*, which considers changes in the attributes of consecutive events within one or multiple event streams [8]. Listing 1 shows the exemplary translation for (t_0, t_1) (Table 2, row 1) into a query of *Streaming SQL* used within *Siddhi* [6]. The attributes of two events ($e1$, $e2$) on the stream related to the Oven are analyzed for the occurrence of the derived changes (line 2). We denote changes that relate to the low-level IoT events as *low-level patterns*. When one of these patterns is detected, a new event is created (line 3) and emitted on a stream for the occurrence of low-level patterns (line 4).

3) Change Sequence Detection: The low-level patterns refer to changes in an activity signature between two points in time (i.e., *one* row in Table 2). An activity is represented by the sequence of these patterns (i.e., *all* rows in Table 2). We use a *hierarchy* of events as CEP feature: the detection of the first low-level

pattern (cf. Table 2, row 1) leads to a high-level event e_{H1}. The detection of the second low-level pattern (cf. Table 2, row 2) leads to a high-level event e_{H2} only if e_{H1} occurred before. This is continued by emitting e_{H3} only if e_{H2} occurred before, etc. The sequence of e_{H1} to e_{H6} represents the entire activity with e_{H1} and e_{H6} denoting its *start* and *end*. This allows us to work with *partial* activity detection (cf. **R3**), e.g., we can specify that 50 % of the high-level events or only e_{H1} are sufficient to identify an activity–trading off completeness and latency.

4) Activity Detection Service: The CEP app serves as event subscriber to receive low-level IoT events and as event publisher to emit process-level events to other subscribers following the publish-subscribe pattern. Additionally, we add service-based interfaces for request-response communication, transforming the CEP apps into CEP-based activity detection services. One service corresponds to one type of activity to be detected. We query the current status of tracking the activity detection as one service method. Additionally, we keep a log of detected activity executions and expose it via another method of the service API.

Service Deployment: Fig. 2 shows the software architecture for the online detection of process activities from IoT data streams. We use the CEP platform *Siddhi* as technological basis to run the activity detection services. Siddhi is itself a service providing an API for deploying and running CEP apps at runtime [6]. By using this API we are able to fully automate the generation of the activity detection services and their deployment to the CEP platform (cf. **R2**). The activity detection services may also be composed to create *Process Detection Services*. Assuming that either the control flow (i.e., sequence of activities) or the *Process Signature* (i.e., composition of activity signatures [18]) is known, a new CEP-based service can be generated to detect process executions in a similar way.

Online Activity Detection: Upon deployment to the CEP platform, the activity detection services can be activated using the platform's API. When activated, a service establishes the connections to the external event sources (here: the IoT components) and the interfaces to be used by external event and service consumers. Low-level IoT events are processed based on the CEP queries in the services. The detected process-level events are published via the message-based interfaces to be used, e.g., for online process discovery or conformance checking.

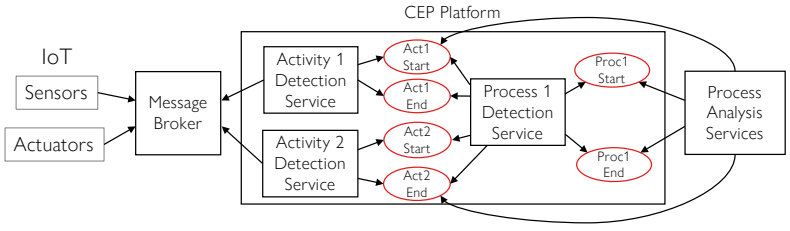

Fig. 2. Service-based software architecture for activity detection at runtime

4 Discussion

We implemented the framework as a proof-of-concept prototype. We generated 6 services to detect activities executed in our smart factory and recorded the associated low-level IoT data (cf. `https://doi.org/10.5281/zenodo.8087219`). We discuss how the framework fulfills requirements **R1**–**R4** and its limitations:

R1 (Non-invasiveness): The integration with existing IoT environments is achieved using *Siddhi* as a light-weight service that integrates seamlessly with other services using standard protocols [6]. Nevertheless, activity detection highly depends on the quality of the low-level data that the IoT components provide. Here, additional sensors and semantic knowledge about IoT components can help to increase data quality through more advanced CEP queries [18].

R2 (No-code services): The activity detection services are automatically generated and deployed to the CEP platform. The domain expert only needs to identify an activity prototype [18]. In contrast to machine learning models, the CEP apps show the patterns used in activity detection–fostering *explainability*. Extensibility is ensured as new services can be added via the CEP platform API.

R3 (Runtime capabilities): CEP platforms process high volumes of event streams at runtime [6,8]. The CEP-based activity detection services allow processing IoT event streams in near real-time and emit process-level events with CAIRO properties [9] for further process analysis. Relying on sequences of patterns in the low-level IoT events, we can also detect partial activity executions.

R4 (Robustness): External factors and different execution parameters might lead to variations in the low-level IoT events and the corresponding changes for the same activities [14]. Detecting the sequence of changes is robust against varying execution *durations* as time is not factored in. Activities of the same type showing different *change sequences* might lead to incorrect activity detections. One approach to address variations is to limit the detection to only the *low-level start* and *end patterns*. The domain expert might also resort to an underfitting approach, selecting the activity prototype that is most common for all instances.

5 Related Work

A general discussion of opportunities, challenges and an architecture for CEP as a service are presented in [11]. Prior works studied the integration of IoT with BPM, with the BPM-meets-IoT manifesto [12] at the forefront. Related works addressing process activity detection and monitoring are [13,17]. The framework proposed in this paper involves event extraction, abstraction, and correlation: Diba et al. in [7] give a general overview on existing approaches to these tasks. Related to the processing of IoT data is the work in [5], which proposes an architecture combining CEP with stream processing to realize a system able

to process heterogeneous IoT data in real time. The proposed architecture, however, does not integrate automatically generated services: we aim at bridging this gap and alleviating analysts' workload. A similar intent is found in [10], with a visual framework for programming CEP apps for processing IoT data; an approach for developing IoT-based monitoring systems is also presented in [2]. While the goal of these works is to support developers by abstracting low-level design aspects, they still involve manual work in a visual programming environment. In contrast, we provide an approach for a fully automated generation of CEP apps as services. In [1], the authors propose a method to automatically generate CEP queries associated with the lifecycle transitions of control flow elements for process monitoring. The method is based on the annotation of a process model with *Monitoring Points*, and therefore assumes formalized, structured knowledge of a process. We relax this assumption and allow unstructured process knowledge, requiring only domain knowledge for recognizing an activity prototype. The approach in [15] allows automatically learning and generating CEP rules for activity detection from historical traces. However, it requires a corpus of training data for the learning phase, which might not always be available. Our approach is applicable also in cases where only one trace is available.

6 Conclusion and Future Work

Novel IoT data sources foster process analysis and decision making at runtime, even in the absence of a PAIS. However, IoT data can be too heterogeneous and too fine-grained to be suitable for process mining [4]. We propose a framework to generate services capable of detecting activity executions from IoT data at runtime. The framework relies on the domain expert to identify an activity execution in the IoT data, from which we generate and deploy an activity detection service based on CEP. The CEP platform provides means for near real-time event processing and extension mechanisms for service composition at runtime.

In future work we will apply semantic knowledge about IoT components and activities to integrate additional CEP patterns for increased robustness of the approach. We will apply the framework in larger scale manufacturing and healthcare settings to prove its feasibility and generalizability in other IoT domains.

Acknowledgments: This work has received funding from the Swiss National Science Foundation under Grant No. IZSTZ0_208497 (*ProAmbitIon* project).

References

1. Backmann, M., Baumgrass, A., Herzberg, N., Meyer, A., Weske, M.: Model-driven event query generation for business process monitoring. In: ICSOC 2013 Workshops. Lecture Notes in Computer Science, vol. 8377, pp. 406–418. Springer (2013)
2. Barricelli, B.R., Valtolina, S.: A visual language and interactive system for end-user development of internet of things ecosystems. Journal of Visual Languages & Computing **40**, 1–19 (2017)

3. Bauer, M., Bui, N., De Loof, J., Magerkurth, C., Nettsträter, A., Stefa, J., Walewski, J.W.: Iot reference model. Enabling things to talk: designing IoT solutions with the IoT architectural reference model pp. 113–162 (2013)
4. Beerepoot, I., Di Ciccio, C., Reijers, H.A., Rinderle-Ma, S., Bandara, W., et al.: The biggest business process management problems to solve before we die. Computers in Industry **146**, 103837 (2023)
5. Corral-Plaza, D., Medina-Bulo, I., Ortiz, G., Boubeta-Puig, J.: A stream processing architecture for heterogeneous data sources in the internet of things. Computer Standards & Interfaces **70**, 103426 (2020)
6. Dayarathna, M., Perera, S.: Recent advancements in event processing. ACM Comput. Surv. **51**(2) (feb 2018)
7. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **10**(3), e1346 (2020)
8. Etzion, O., Niblett, P.: Event processing in action. Manning Publications (2010)
9. Franceschetti, M., Seiger, R., Weber, B.: An event-centric metamodel for iot-driven process monitoring and conformance checking. In: Business Process Management Workshops. Springer International Publishing (2023)
10. Gökalp, M.O., Koçyiğit, A., Eren, P.E.: A visual programming framework for distributed internet of things centric complex event processing. Computers & Electrical Engineering **74**, 581–604 (2019)
11. Higashino, W.A., Capretz, M.A., Bittencourt, L.F.: Cepaas: Complex event processing as a service. In: Int. Congress on Big Data. pp. 169–176. IEEE (2017)
12. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Fortino, G., Gal, A., Kannengiesser, U., Leotta, F., et al.: The internet of things meets business process management: a manifesto. IEEE Systems, Man, and Cybernetics Magazine **6**(4), 34–44 (2020)
13. Janssen, D., Mannhardt, F., Koschmider, A., van Zelst, S.J.: Process model discovery from sensor event data. In: ICPM 2020. pp. 69–81. Springer (2020)
14. Lee, E.A.: Cyber physical systems: Design challenges. In: 2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC). pp. 363–369. IEEE (2008)
15. Mousheimish, R., Taher, Y., Zeitouni, K.: Autocep: automatic learning of predictive rules for complex event processing. In: ICSOC 2016. pp. 586–593. Springer
16. Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of management information systems **24**(3), 45–77 (2007)
17. Rebmann, A., Emrich, A., Fettke, P.: Enabling the discovery of manual processes using a multi-modal activity recognition approach. In: BPM 2019 International Workshops, Revised Selected Papers 17. pp. 130–141. Springer (2019)
18. Seiger, R., Franceschetti, M., Weber, B.: An interactive method for detection of process activity executions from iot data. Future Internet **15**(2) (2023)
19. Seiger, R., Malburg, L., Weber, B., Bergmann, R.: Integrating process management and event processing in smart factories: A systems architecture and use cases. Journal of Manufacturing Systems **63**, 575–592 (2022)
20. Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., Di Ciccio, C., Koldehofe, B., Kopp, O., Jacobsen, A., Sürmeli, J., Song, W.: From event streams to process models and back: Challenges and opportunities. Inf. Sys. **81**, 181–200 (2019)
21. Van Der Aalst, W., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., et al.: Process mining manifesto. In: BPM 2011 International Workshops, Part I 9. pp. 169–194. Springer (2012)