A Holistic Framework for IoT-Aware Business Processes

Yusuf Kirikkayis(✉), Florian Gallik, and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Ulm, Germany
{yusuf.kirikkayis,florian-1.gallik,manfred.reichert}@uni-ulm.de

Abstract. The Internet of Things (IoT) enables a variety of smart applications, including smart home, smart factory, and smart health. As Business Process Management (BPM) can also benefit from IoT technologies, the combined use of BPM and IoT has attracted considerable research works. Providing integrated lifecycle support for modeling, executing, and monitoring IoT-aware business processes constitutes a challenge. Existing process modeling and execution languages such as BPMN 2.0 are unable to fully meet the requirements of IoT-aware processes. In this paper, we present an extension of BPMN 2.0 for modeling, executing, and monitoring IoT-aware business processes. We introduce specific artifacts and events that enable IoT awareness during the execution and monitoring of IoT-driven business processes. The resulting framework is illustrated along a real-world scenario.

Keywords: BPMN · IoT · IoT-aware BPM · Execution engine · BPMS

1 Introduction

The interest and relevance of the Internet of Things (IoT) has been increasing continuously during the recent years and IoT has becomes one of the most relevant technologies to realize digital twins of the physical world [1]. The electronic components of IoT devices are becoming smaller, cheaper, and more powerful. As a result, IoT technology is experiencing an upswing [2]. IoT devices are equipped with sensors, actuators, software, protocols, and various network interfaces. This enables IoT devices to capture, collect, and exchange data as well as to physically respond to events [3]. While sensors are used to collect and capture data about the physical world (e.g., humidity, air quality, and temperature), actuators are used to control the latter (e.g., watering systems, light control, air conditioner, and security systems) [4]. While IoT enables the collection and exchange of data about the physical world, BPM enables modeling, implementing, executing, monitoring, and analyzing business processes [5]. Incorporating IoT capabilities into BPM systems, therefore, offers promising perspectives for process automation including automated decision making that takes the state of the physical world into account as well. Moreover, IoT devices can be used to automate various

C. Cabanillas et al. (Eds.): BPM 2022, LNBIP 460, pp. 89–100, 2023.
https://doi.org/10.1007/978-3-031-25383-6_8

types of physical tasks (e.g. opening a window) or digital tasks (e.g. transferring data) [5]. To be able to provide lifecycle support for IoT-aware processes their modeling requires specific elements that allow capturing the physical process context appropriately. Amongst others modeling IoT-aware processes shall foster the understanding of how these processes operate as well as enable the detection and avoidance of problems. Moreover, it should be possible to detect and handle errors and exceptions (e.g., faulty sensors) during process enactment. Existing standards such as BPMN 2.0 do not provide sufficient expressiveness for modeling IoT-aware processes [3].

In this paper, we enhance BPMN 2.0 with IoT-specific artifacts and events, which enable the modeling, execution, and monitoring of IoT-aware processes. The functions and benefits of these artifacts and events are illustrated along a real-world manufacturing process in a smart factory. The remainder of this paper is structured as follows. In Sect. 2, we summarize the problems that emerge when modeling IoT-aware processes with the standard BPMN 2.0 language. Section 3 describes the proposed extension, i.e., specific artifacts and events. In Sect. 4 we present our approach for modeling, executing, and monitoring IoT-aware business processes, which is then applied in the context of a case study in Sect. 5. Section 6 discusses related work. Finally, Sect. 7 summarizes the approach and provides an outlook on future work.

2 Problem Statement

Though BPMN 2.0 does not provide explicit support for capturing and modeling IoT capabilities, it offers various ways to represent IoT aspects such as tasks, events, and resources [6]. However, following such a straightforward approach, no distinction between IoT-related process model elements and regular elements can be made [14]. Consequently, it does not become apparent whether or not a process includes IoT aspects. Instead, the process model needs to be read and understood based on the chosen element (e.g. task) labels.

Figure 1 illustrates a process with IoT aspects modeled in terms of BPMN 2.0. Along this example, we want to demonstrate characteristic problems. The depicted production process involves multiple actuators as well as sensors. The process starts every ten seconds and then checks whether the High-Bay Warehouse (HBW) light barrier is interrupted. If the latter applies, the Vacuum Gripper Robot (VGR) starts moving, otherwise the process terminates. However, the VGR is moved until reaching the pick-up station whose light barrier check is embedded within a loop. After the VGR has reached the pick-up station, a QR code is generated. Subsequently, the status of the HBW is checked. If the status is *OK*, the workpiece is stored in HBW, otherwise it is transported to HBW 2. Finally, the process terminates.

When using BPMN 2.0 for modeling the IoT-related tasks (cf. Fig. 1), it is unclear, which tasks involve IoT devices and which do not. It is further unclear that business rule tasks shall represent sensors and service tasks shall represent actuators. Instead, the process model reader needs to interpret the task labels

correctly to properly understand the process model. Moreover, no visual distinction can be made between an IoT-related service task (cf. Tasks 2, 6, and 7 in Fig. 1) and a service task not involving IoT devices (cf. Task 4), or between an IoT-related business rule task (cf. Tasks 1 and 3) and a normal one (cf. Task 5). Note that this aggravates both the readability and the comprehensibility of the process model.

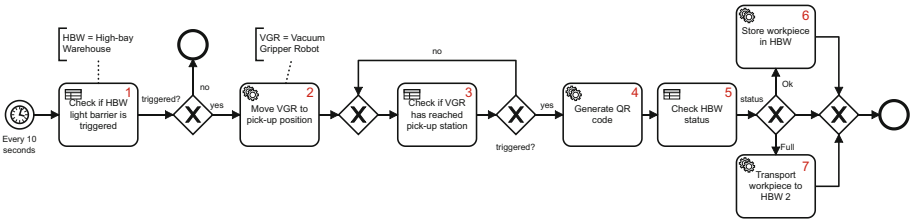


Fig. 1. IoT-aware business process modeled in terms of BPMN 2.0.

3 Solution Proposal

The goal of this paper is to provide a BPMN 2.0 extension that enables the modeling and enactment of IoT-aware processes. Moreover, the behavior of these elements needs to be mapped to a process execution engine, which constitutes the core architecture component of our approach. Taking the problem statement set out in Sect. 2, we extended BPMN with the artifacts and events shown in Fig. 2. In the following, each of these elements is shortly described. Note that all elements are decorated with a WLAN icon and labeled as "IoT". In addition, the letter in the upper left corner indicates the artifact type (i.e., "A" for actuator and "S" for sensor).

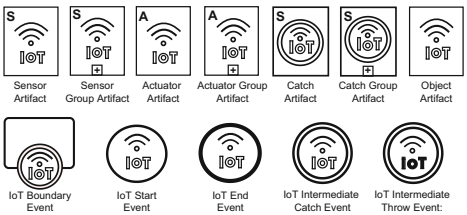


Fig. 2. Extending BPMN 2.0 with IoT-specific elements

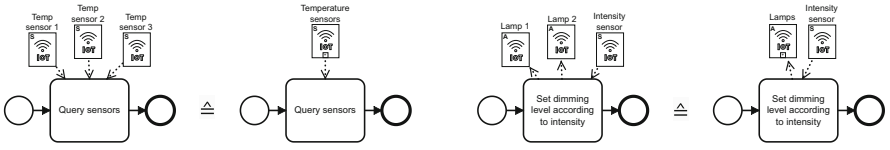
Sensor Artifact: A sensor artifact (Fig. 2) can represent various sensors in a business process model (e.g., measuring temperature, speed, or GPS), and enables the collection of data from the physical environment and process context. When connecting a sensor artifact to a task, the corresponding sensor may

be queried by the task during its execution. All necessary information about the sensor is captured by the sensor artifact. The task connected to it can only be successfully completed after having received a positive response from the sensor. Note that the explicit representation of sensors as artifacts allows linking any number of sensors to a task (Fig. 3a), and a sensor artifact may be arbitrarily combined with other artifacts (Fig. 3b). In such a case, the sensors are concurrently queried during task execution. Note that the representation of the artifact is generic allowing for the representation of arbitrary sensor types. *Moreover, text annotations may be used, for example, to designate artifacts and events.*

Sensor Group Artifact: A sensor group artifact is represented by a collapsed sensor artifact (cf. Fig. 2) and shows the same behavior as a sensor artifact. As depicted in Fig. 3a, individual sensor artifacts may be aggregated to a sensor group artifact in order to increase the abstraction level. More precisely, a sensor group artifact combines multiple sensor artifacts ($n \geqslant 2$, *with n being number of sensor artifacts*).

Actuator Artifact: An actuator artifact (cf. Fig. 3(b)) allows modeling actuators (e.g., electric motor, relay, light, and microphone). This enables the process to react to situations, e.g., by opening a window as soon as a certain temperature threshold is exceeded. An actuator is controlled by the task associated with the corresponding actuator artifact. All necessary information about the actuator is captured by the actuator artifact. The corresponding task is completed successfully once it has received a positive response from the actuator. Note that the representation of the artifact is generic, allowing for the representation of arbitrary actuator types.

Actuator Group Artifact: An actuator group artifact is represented by a collapsed actuator artifact (cf. Fig. 3b) and shows the same behavior as an actuator artifact. More precisely, an actuator group artifact combines multiple actuator artifacts ($n \geqslant 2$, *with n being number of actuator artifacts*). Figure 3b shows an example combining both sensor and actuator artifacts.



(a) Usage of sensor artifact and sensor group artifact

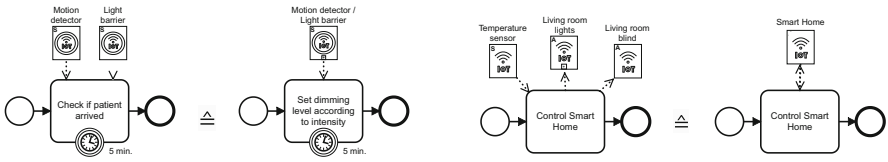
(b) Usage of actuator artifact and actuator group artifact

Fig. 3. Using extended IoT elements in BPMN 2.0.

Catch Artifact: A catch artifact (cf. Fig. 4a) allows checking a condition during task processing in combination with a boundary timer event. Immediately after starting the task, the respective condition is continuously checked. All necessary information about the condition is provided by the catch artifact. The task may be completed successfully only when meeting the specified condition. If the condition is not satisfied within the time period specified by the boundary timer event, the sequence flow attached to the latter is executed (cf. Fig. 4a). If other artifacts are attached to the task, their execution and verification run in parallel.

Catch Group Artifact: A catch group artifact is represented by a collapsed catch artifact. It shows the same behavior as a catch artifact (cf. Fig. 2). As depicted in Fig. 4a, the group artifact aggregates multiple catch artifacts to increase the abstraction level ($n \geqslant 2$, *with n being number of catch artifacts*).

Object Artifact: An object artifact (cf. Fig. 2) enables the modeling of physical objects (e.g., service robot, machine, or smart factory) of the environment, in which the business process is executed. As illustrated in Fig. 4b An object artifact may contain both sensors and actuators. On one hand, this allows hiding unnecessary information from domain experts. On the other, modeling becomes more accurate when using an object artifact.



(a) Usage of IoT intermediate catch and group artifact

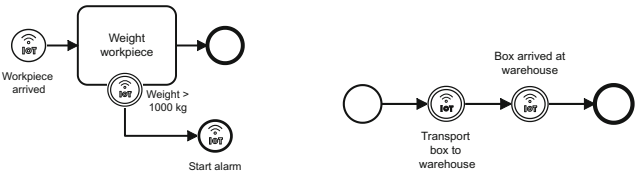
(b) Usage of IoT Object artifact

Fig. 4. Using extended IoT elements in BPMN 2.0.

IoT Boundary Event: An IoT boundary event (cf. Fig. 5a) may be used to define a condition and redirect the sequence flow accordingly if this condition becomes fulfilled during task execution. After starting the task, the condition is continuously checked. This condition checking terminates either upon task completion or when meeting the condition. *Note that all events use sequence flow as connection type.*

IoT Start Event: To enable the start of an IoT-aware process based on IoT sensors, the IoT start event (cf. Fig. 5a) can be used. It trigger a process instance when meeting the specified start condition (e.g., *temperature* $\geqslant 20\,°C$ or *"motion detected"*).

IoT End Event: An IoT end event (cf. Fig. 5a) triggers the execution and/or control of an actuator and terminates the corresponding process instance. Unlike the IoT start event, the end event has a thicker border.



(a) Usage of IoT Boundary, start, and end Event

(b) Usage of IoT intermediate catch and throw event

Fig. 5. Using extended IoT elements in BPMN 2.0.

IoT Intermediate Catch Event: An IoT intermediate catch event (cf. Fig. 5b) is linked to an IoT sensor. It enables the process to check a physical condition along the sequence flow (e.g. *volume > 60 decibels*). More precisely, when reaching an IoT intermediate catch event the sequence flow does not continue until its corresponding condition is met. Note that an artifact (e.g., sensor or actuator artifact) must not be linked to an IoT intermediate catch event.

IoT Intermediate Throw Event: To control an actuator along a sequence flow, the IoT intermediate throw event (cf. Fig. 5b) may be used. Semantically, such events corresponds to a task with a linked actuator artifact. However, only one actuator may be controlled at the same time in the context of an IoT intermediate throw event. The latter is successfully completed upon receipt of a positive response from the actuator. Only then, the sequence flow continues.

4 Business Process Management System for IoT

To execute and monitor IoT-aware processes based on the the described BPMN 2.0 extension, an appropriate software architecture is needed that implements the behavior of the various elements. A coarse-grained view on such an architecture is shown in Fig. 6. It consists of three main components, i.e., BPM system, MQTT broker, and IoT services. An *IoT service* controls or queries IoT devices and offers corresponding functions via its interface (e.g. REST API), while at the same time hiding technical peculiarities of the IoT devices from the calling environment (e.g. the IoT-aware process engine). For example, an IoT service may provide an endpoint to allow querying the room temperature, which can then be fetched with a GET request.

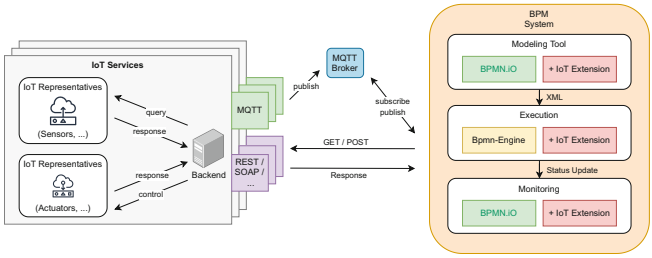


Fig. 6. Architecture of the BPM System.

The *MQTT broker* (cf. Fig. 6) is used to reduce the network load and to react to occurring IoT events. IoT devices publish their data to the broker, which, in turn, distributes these data to all subscribers. Thus, it becomes possible to react to IoT events in real-time during process execution. The *BPM system* consists of three main components, i.e., its *modeling tool*, execution engine, and monitoring system. The modeling tool is based on bpmn.io[1], extended with the elements described in Sect. 3. The latter are therefore available for modeling IoT-aware processes and can be configured for the respective execution context. A complete process model contains all information necessary to execute the IoT-aware process. The corresponding process model information is encoded in an XML file. The corresponding XML format is machine-readable and, thus, executable. The open source javascript workflow engine[2] serves as the basis for executing the IoT-aware processes. We extend the engine with the elements and implement their behavior (e.g., to react to IoT events by subscribing to MQTT topics, to query sensors, or to control actuators using GET/POST requests) Sect. 3. If a new process instance is created the engine reads the XML file and then starts process execution. The *monitoring component*, in turn, uses bpmn.io with the IoT extension as a basis, just like the modeling tool. However, the process model can only be viewed and not edited. The *execution engine* ensures that the state and the data of the process instance become updated in real time, i.e., users can always view the current state of the running IoT-aware process. Corresponding color markings indicate to them how far the execution of the IoT-aware process has progressed and where errors have occurred.

5 Smart Factory Scenario Process

In the following, we illustrate the modeling, execution, and monitoring of an IoT-aware process along a sophisticated smart factory scenario. Using the scenario we want to investigate whether the framework enables the generic integration of business processes with IoT capabilities. Note that we also applied the framework to IoT-aware processes from other domains such as smart home, smart healthcare, and smart logistics.

[1] https://bpmn.io.
[2] https://github.com/paed01/bpmn-engine.

We use physical simulation models developed by *Fischertechnik*®(FT)[3] to emulate the smart factory. These models simulate a complete production line as shown in Fig. 7. The smart factory consists of 5 stations, i.e. high-bay warehouse, vacuum gripper robot, oven, milling machine, and sorting machine.

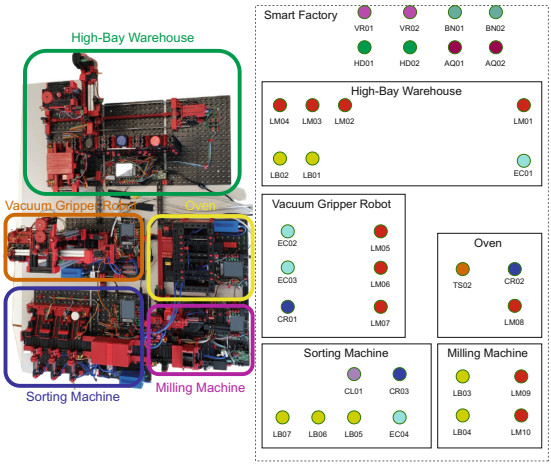


Fig. 7. Overview of sensors in the smart factory.

The smart factory is equipped with six different types of IoT sensors (cf. Fig. 7).

- Limit switch sensors (represented by red circles and labeled as *LM*).
- Light barrier sensors (represented by yellow circles and labeled as *LB*).
- Pressure sensors (represented by blue circles and labeled as *CR*).
- Temperature sensors (represented by orange circles and labeled as *TS*).
- Encoder sensors (represented by cyan circles and labeled as *EC*).
- Color sensors (represented by purple circles and labeled as *CL*).

In addition to the sensors installed in the smart factory, the scenario comprises sensors that measure the environment:

- Vibration sensors (represented by pink circles and labeled as *VR*).
- Brightness sensors (represented by green circles and labeled as *BN*).
- Humidity sensors (represented by neon green circles and labeled as *HD*).
- Air quality sensors (represented by dark red circles and labeled as *AQ*).

In total, the smart factory is equipped with 34 sensors (cf. Fig. 7), i.e. 7 light barrier sensors that detect the interruption of a light beam and display it as an electrical signal, 10 limit switch sensors actuated by the movement of a machine

[3] https://www.fischertechnik.de/en/simulating/industry-4-0.

part or the presence of an object, 3 pressure sensors that measure the overpressure of the suction, 1 temperature sensor that measures the temperature in the oven, 4 encoder sensors that return the current position of the motors, and 1 color sensor to recognize the workpiece color in the sorting machine. In order to be able to assess the workpiece quality, 2 vibration sensors, 2 brightness sensors, 2 humidity sensors, and 2 air quality sensors are additionally used as well.

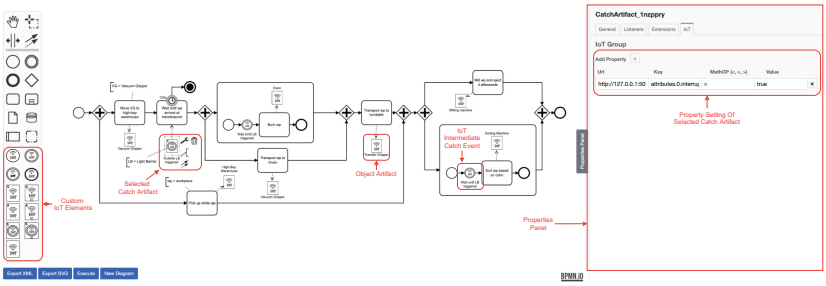


Fig. 8. Modeling and configuring IoT-aware process with the modeling tool.

Figure 8 shows an example of a manufacturing process of the smart factory that we can model, configure, execute, and monitor with our approach. First, a white workpiece (WP) is taken from the high-bay warehouse to the transfer point. In parallel, the vacuum gripper moves to the transfer point and waits there until the workpiece arrives. Waiting is realized with a catch artifact (cf. Sect. 3), which is attached to a task. This artifact checks for the triggering of a light barrier. If the condition is not met within 120 s, the process terminates. As soon as the workpiece has arrived at the transfer point, the vacuum gripper transports it to the oven. The transport is realized by an object artifact (cf. Sect. 3). In parallel, the oven waits until the workpiece arrives. Upon arrival, the workpiece is burned and then transported to the turntable where it is milled. Afterwards the workpiece is moved on a conveyor belt to the sorting machine. Once the light barrier is triggered, the color of the workpiece is detected and the workpiece is sorted according to color. Then, the process terminates.

The smart factory is controlled with the Business Process Management System (BPMS) presented in Sect. 4. First of all, the IoT-aware process needs to be modeled, including the configuration of the involved IoT devices. Figure 8 shows the modeling component of the BPMS that provides all standard BPMN 2.0 elements as well as the newly introduced IoT-specific elements (cf. Sect. 4). The process model shown in Fig. 8 comprises an object artifact, an IoT intermediate catch event, and a sensor catch artifact. In the properties panel (cf. Fig. 8), the IoT condition (cf. Sect. 4) is configured using properties. Figure 8 shows the configuration of the selected sensor catch artifact. For all group artifacts as well as for the object artifact, one may use the *plus* symbol of the *Add Property* label to add another condition. The following properties need to be set for the catch

artifact: (i) an endpoint, (ii) an attribute to be accessed from the response, and (iii) the condition that needs to be statisfied in order to continue process execution. Since a boundary timer event is attached to task *Wait until WP arrived at transfer point*, the condition is checked for correctness only as long as the specified timeout has not been triggered yet. If the timeout is reached, however, the corresponding sequence flow is executed and process execution then terminates. After modeling, the process can be exported as a machine-readable XML file. This XML file contains all the information required to execute the IoT-driven process.

The modeled and configured IoT-aware process is then deployed to the execution engine of the BPM system, and new process instances may be created and started. Figure 9 shows the execution of one such instance. All elements colored in green have been successfully executed; their actual execution time is attached as a yellow colored overlay. The elements colored in orange are currently executed. In case of an error (e.g., network error or timeout during execution), the corresponding elements are colored in red. On the right side an execution log is displayed, which shows relevant events (e.g., current progress and error messages).

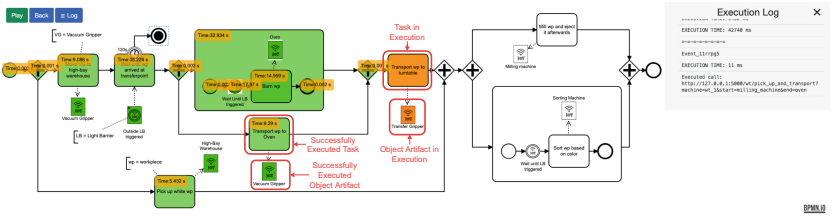


Fig. 9. IoT-aware smart factory process in execution. (Color figure online)

6 Related Work

There exist several works that introduce BPMN notations and extensions for capturing IoT aspects of business processes [12]. [7] enhances BPMN with a sensor task that covers the following aspects: (i) sensor service, (ii) sensor handler, and (iii) sensor device. For representing physical entities (e.g. a bottle of milk) a collapsed pool is used in [8]. In addition, two task types for sensing and actuation are introduced. [9] introduces a wireless sensor network (WSN) task and a WSN pool. The WSN task has an actionType (e.g. sensing (?) or actuation (!)). In [10], an Industry 4.0 process modeling language (I4PML) based on BPMN 2.0 is presented. I4PML comprises the following elements: (i) cloud app, (ii) IoT device, (iii) device data, (iv) actuation task, (v) sensing task, (vi) human computer interface, and (vii) mobility aspect. uBPMN [11], in turn, introduces additional elements for camera, collector, sensor, and microphone. Each of these elements is represented by its own task and event types.

The existing approaches enable modeling IoT aspects in BPMN. However, the approaches are either too specific or cannot fully represent the behavior of IoT devices. For example, none of the approaches enables the modeling of an IoT boundary event. Another scenario that cannot be modeled with existing approaches concerns the verification of an IoT condition when processing a task.

None of the described approaches allows modeling IoT-aware processes at different levels of abstraction as our approach does (cf. Sect. 4). Note that different abstraction levels enable different process views for the various stakeholders (e.g., process expert, IoT expert, or domain expert). Moreover, existing approaches do not support the execution and monitoring of the elements they introduced. Table 1 summarizes the approaches (with ✗ expressing missing support and ✓ indicating support of the respective feature). As can be easily seen, none of the existing approaches encompasses IoT-enabled processes with the necessary treatment.

Table 1. Comparison of related work.

	Sensor	Actuator	Combining sensor and actuator	Start event	End event	React to IoT within a task	Intermediate event	Condition element	Physical entity	IoT data object	Abstraction level	Multiinstance	Execution engine	Score
Cheng et al. [7]	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	1/12
Meyer et al. [8]	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	3/12
Sungur et al. [9]	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	3/12
I4PML [10]	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	4/12
uBPMN [11]	✓	✗	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	5/12
This work	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	11/12

7 Conclusions and Outlook

This paper presented a BPMN 2.0 extension to enable the modeling, execution, and monitoring of IoT-aware business processes. Taking the given problem statement as well as the insights we gained from our literature review, we were able to identified fundamental challenges regarding the modeling, execution, and monitoring of IoT-aware business processes. We extended BPMN 2.0 with IoT artifacts and events that address the identified gaps. The added elements allow collecting, capturing, and exchanging data about the physical world over the Internet with the sensor artifact as well as controlling actuators with the actuator artifact. In addition, IoT conditions may be validated during task processing by the IoT intermediate catch artifacts as well as along the sequence flow with the IoT intermediate events. Furthermore, process start may be triggered by an IoT condition associated with an IoT start event and a process end may execute an actuator with the IoT end event. Finally, the IoT boundary event allows redirecting the sequence flow based on an IoT condition. In addition to the BPMN 2.0 IoT-related extensions, we presented an architecture to execute and monitor the modeled IoT-aware processes. The architecture allows for the execution and monitoring of IoT-aware processes in real time. Moreover, we applied our approach to a smart factory case to demonstrate that the framework is beneficial for modeling, executing, and monitoring IoT-aware business processes.

In future work, we want to make our framework multi-instance capable. In addition, we would like to generate IoT-enhanced logs and exploit them for an advanced process support treatment (e.g. to discover deviation between digital processes and their counterparts in the physical world or to improve process analytics). Moreover, we will conduct additional case studies of different domains to validate the domain independence.

References

1. Chang, C., Srirama, S.N., Buyya, R.: Mobile cloud business process management systems for the internet of things: a survey. ACM Comput. Surv. **49**, 1–42 (2016)
2. Ashton, K.: That 'internet of things' thing. RFID J. **22**, 97–114 (2009)
3. Kirikkayis, Y., Gallik, F., Reichert, M.: Towards a comprehensive BPMN extension for modeling IoT-aware processes in business process models. In: Guizzardi, R., Ralyté, J., Franch, X. (eds.) RCIS 2022. LNBIP, vol. 446, pp. 711–718. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-05760-1_47
4. Valderas, P., Torres, V., Serral, E.: Modelling and executing IoT-enhanced business processes through BPMN and microservices. J. Syst. Softw. **184**, 111139 (2022)
5. Janiesch, C., et al.: The internet of things meets business process management: a manifesto. Syst. Man Cybern. Mag. **6**, 34–44 (2020)
6. Hasić, F., Serral, E.: Executing IoT processes in BPMN 2.0: current support and remaining challenges. In: RCIS (2019)
7. Cheng, Y., et al.: Modeling and deploying iot-aware business process applications in sensor networks (2019)
8. Meyer, S., Ruppe, A., Hilty, L.: The things of the internet of things in BPMN. In: Conference in Advanced Information Systems Engineering Workshops (2015)
9. Sungur, C.T., et al.: Extending BPMN for wireless sensor networks. In: Business Informatics (2013)
10. Petrasch, R., Hentschke, R.: Process modeling for industry 4.0 applications towards an industry 4.0 process modeling language and method. In: Computer Science and Software Engineering (2016)
11. Alaaeddine, et al.: uBPMN: a BPMN extension for modeling ubiquitous business processes. Inf. Softw. Technol. **74**, 55–68 (2016)
12. Torres, V., Serral, E., Valderas, P., Pelechano, V., Grefen, V.: Modeling of IoT devices in business processes: a systematic mapping study. In: CBI (2020)
13. Marrella, A., Mecella, M., Sardina, S.: SmartPM: an adaptive process management system through situation calculus, IndiGolog, and classical planning. In: Principles of Knowledge Representation and Reasoning (2014)
14. Kirikkayis, Y., Gallik, F., Reichert, M.: Modeling, executing and monitoring IoT-driven business rules with BPMN and DMN: current support and challenges. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) EDOC 2022. LNCS, vol. 13585, pp. 111–127. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17604-3_7