IoTDM4BPMN: An IoT-Enhanced Decision Making Framework for BPMN 2.0

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

***Abstract*—The relevance of the Internet of Things (IoT) for Business Process Management (BPM) support is increasing. IoT devices enable the collection and exchange of data over the Internet, whereby each physical device is uniquely identifiable through its embedded computing system. BPM, in turn, is concerned with analyzing, discovering, modeling, executing, and monitoring (digitized) business processes. By enhancing BPM systems with IoT capabilities, real-world data can be gathered and considered during process execution to enhance process monitoring as well as IoT-driven decision making. In this context, the aggregation of low-level IoT data into high-level process-relevant data constitutes a fundamental step towards IoT-driven decisions in business processes. This paper presents IoT Decision Making for Business Process Model and Notation (IoTDM4BPMN) a web-based framework for modeling, executing, and monitoring IoT-driven decisions in real-time. We give insights into the design and implementation of IoTDM4BPMN and provide a case study as a first validation that applies IoTDM4BPMN to the modeling, executing, and monitoring of a real-world IoT-driven decision process.**

***Index Terms*—BPM, BPMN, IoT, BPM in IoT, IoT Decision, Decision**

I. Introduction

As electronic components are becoming smaller, more powerful, and less expensive, the application areas of the Internet of Things (IoT) are wide-ranging and diverse, including, e.g., smart home environments, medical monitoring systems, and smart factories [2] [21]. IoT is a network of physical objects, i.e., sensors and actuators, which collects data by sensor(s) and action(s) are triggered by actuator(s) [1]. The use of such interconnected physical objects allows transferring an environmental context from the physical to the digital world [3]. While IoT enables the collection and exchange of data about the physical world, Business Process Management (BPM) allows analyzing, modeling, discovering, executing, and monitoring business processes [4]. The incorporation of IoT capabilities to BPM suites creates both business opportunities and customer value, improving process execution, process monitoring, and decision making. Moreover, this combination allows monitoring the progress of manual tasks based on appropriate sensors [5]. By adding IoT devices to a business process, contextual information of its physical environment can be gathered and exploited. This additional data can contribute to the understanding of the process [1]. Furthermore, IoT devices allow for a real-world awareness of digitized business processes. IoT devices can be further used in business processes to automate different types of tasks, which may be digital (e.g. sending data) or physical (e.g., moving a robot) [5]. Overall, IoT enables us to continuously enhance process support with real-time IoT sensor data. Such IoT-aware processes often expose a need for context aggregation and context awareness [7]. To meet this need, the collection and processing of IoT sensor data should be accomplished as follows (i) sense the low-level data (e.g., brightness, temperature, switch state) from the physical world, (ii) aggregate and combine low-level sensor data to high-level information, and (iii) obtain meaningful information to enable IoT-driven decision-making [6].

In general, **low-level data** is generated by sensing the physical world. These low-level data, in turn, are aggregated and combined to **high-level data**, which enhances BPM with physical context data about the real-world, i.e., IoT-driven processes are context- and real-world-aware [7] [3].

Decisions to be made during business process execution require high-level information about the real-world. In this context, it is not sufficient to only retrieve data from traditional repositories, such as databases and data warehouses. In addition, IoT sensor data, provided via in-memory databases or complex event processing, might be useful as well [5].

The combination of IoT with BPM support has gained significant attention in literature. In particular, several notations, approaches, or extensions of the Business Process Model and Notation (BPMN) [8] have been proposed to integrate IoT devices in a process model in terms of resources. Following this straightforward approach, however, IoT data is directly used without aggregating and combining it with other contextual data to obtain high-level information, which impairs its potential capability [3]. Nevertheless, the integration of IoT with BPM is limited due to the lack of a methodological framework to connect an IoT infrastructure to BPM. In contemporary approaches and frameworks, the involvement of IoT devices in decision making does not become apparent. In addition, IoT-driven decision making is indistinguishable from other decisions. Finally, the monitoring of IoT-driven decisions is usually neglected, which leads to difficulties in discovering errors as well as extending, and maintaining the decision logic.

The introduction of the Decision Model and Notation (DMN) [9] [10] standard provides a solution to model decision logic separately from the process logic. In addition, DMN enables the aggregation of low-level information to high-level one. However, in DMN neither monitoring nor error handling are optimized for IoT-driven decisions. For example, if a physical error occurs, it is difficult to identify the source

of the erroneous IoT device as the decision logic is defined exclusively in decision tables.

In this paper, we present IoT Decision Making for BPMN (IoTDM4BPMN for short), a web-based framework for modeling, executing, and monitoring IoT-driven decisions in real-world-aware business processes. The framework uses low-level physical data from IoT devices to evaluate predefined decision rules. In addition, the framework allows logging the sensor data, the decision rules, and the finally made decision. Finally IoTDM4BPMN fosters the understanding of the decision logic through visualization techniques (e.g., coloring).

The remainder of this paper is organized as follows. Section II summarize the problems and issues that emerge when modeling IoT-driven decisions with BPMN and DMN respectively. Related work is discussed in Section III. Section IV defines the requirements for the framework, which is described in detail in Section V. Finally, Section VI summarizes and discusses our results.

II. Problem Statement

A. IoT-driven decisions in BPMN

To model, execute, and monitor IoT-driven decisions in business processes, in principle, standard BPMN elements may be used. As BPMN 2.0 does not explicitly cover IoT capabilities, the involvement of IoT devices in modeling, executing, and monitoring is not apparent. On one hand, IoT devices can be represented by services, scripts, and Business rule tasks in BPMN. On the other, data objects, events, or resources may be used in combination with annotations to express IoT involvement [22]. However, when using standard BPMN elements to represent IoT devices, no distinction between regular BPMN tasks and IoT-related tasks can be made.

To model IoT-driven decisions, gateways may be used in BPMN. As IoT-driven decisions often involve multiple IoT devices, however, decision rule complexity increases with the number of IoT devices involved. Note that this affects both the readability and comprehensibility of IoT-driven decisions in business processes. In addition, with a high number of decisions and a complex nested rule logic, the process model might become challenging to read. As another problem the higher the number of involved IoT devices is, the more complex the extension and maintenance of the decision logic becomes. Finally, modeling IoT-driven decisions in BPMN impairs the scalability and flexibility of the process model.

To illustrate the problems and issues relevant in this context, a simplified process from the healthcare domain is used.

Example 1*: Consider a system that monitors the health status of a patient with Chronic Obstructive Pulmonary Disease (COPD). COPD describes a disease in which the lungs, airflow, and breathing of the patient are obstructed. At any point in time, the patient may experience unpleasant complications such as fast heart rate, hyperactive muscle use, fast breathing, and a cold skin. In this context it has been shown that IoT-driven monitoring of sensor-equipped patients can help increasing their quality of life. In order to detect COPD, all necessary sensors are queried. Based on the values provided by them, either no treatment, treatment with an oxygen mask, or treatment with an inhaler is administered.*

The IoT-driven process from *Example 1* is modeled in terms of BPMN 2.0 in Figure 1. The following issues emerge when modeling IoT-driven decisions with BPMN:

1) The involvement of IoT does not become clear.
2) The complexity of the decision increases with the number of IoT devices involved.
3) IoT data is used without linking it to other contextual process data.
4) With an increasing number of decisions and a complex nested rule logic, the process model becomes less incomprehensible.
5) The scalability and flexibility of the resulting process model becomes impaired
6) Any later extension or change of the specified decision logic becomes impaired.

B. IoT-driven decisions in DMN

An IoT-driven decision can be modeled in terms of DMN as well [9]. The decision logic, input data for making the decision, and the decision table provided by DMN may be used for this purpose. As DMN separates decision from process logic, monitoring becomes challenging. When modeling decisions with DMN, the respective decision rule is represented exclusively via a Business rule task in BPMN. If an error occurs during the execution of an IoT-driven decision in BPMN, such as sensor failure, non-reachability of an IoT device, or an erroneous sensor value, the Business rule task will throw an error. To figure out which sensors are erroneous, one must consider the DMN decision process in addition to the BPMN process model (Figure 2). Then, all sensors and IoT-driven decision tables must be checked. In addition, IoT devices cannot be distinguished from other input data or devices in DMN. This turns monitoring and troubleshooting into a challenging task (Figure 2). As another problem, during the execution of an IoT-driven decision it is not possible to trace back how the DMN-based decisions are actually made. Instead, the Business rule task only provides the final decision (Figure 2).

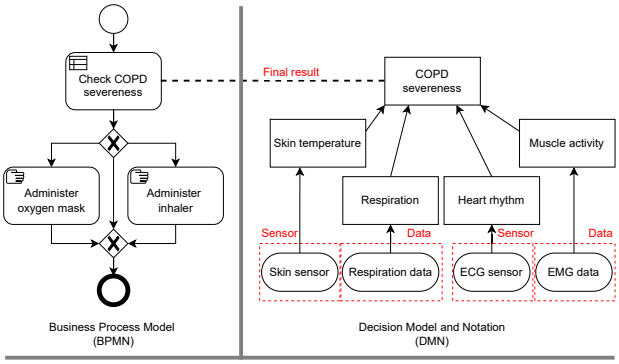

Fig. 2. Relationship between BPMN and DMN (Adapted from [10])

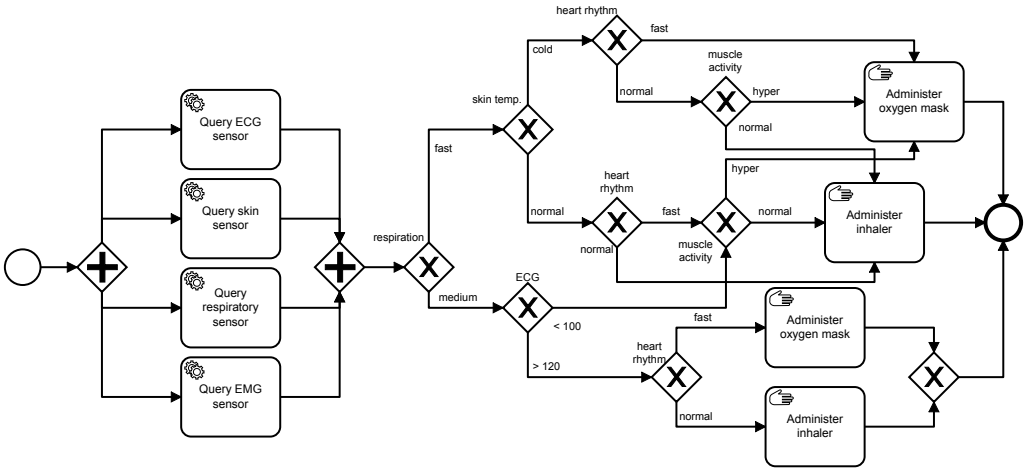

Fig. 1. IoT-Driven decision process modeled with BPMN (adapted from [9] [15])

III. Related Work

There exist several approaches [16] [17] [18] that integrate IoT devices into BPMN models in terms of specific artifacts, IoT tasks, and physical entities. Most approaches focus on automating process execution or improving performance efficiency. Regarding the modeling of IoT-aware processes, various concepts for representing IoT devices at the modeling level have been proposed. For example, many existing approaches integrate IoT entities as a resource in business process models. However, this straightforward approach does not utilize IoT data for context reasoning and decision making [18].

In [3], a framework is presented to bridge the gap between IoT infrastructure and context-aware Business Process Management (BPM) by integrating IoT data into context ontologies. This integration intends to improve business process decision making. A BPM ecosystem with four components is proposed contextual process models, contextual models, decision models, and contextual process execution. However, the framework does not allow for the monitoring of IoT-driven decisions in real-time. As decisions are considered separately from the process flow, in addition their comprehensibility becomes challenging.

In [19], IoT data is used for data analysis to improve decision making. For this purpose, a model for logistics management based on RFID technology is proposed. This model enables the detection of inconsistencies based on IoT-driven decisions. However, the developed model is specific to logistics management and cannot be used in other domains. Moreover, the decisions are not modeled separately, but are hard-coded in the model. Finally, the modeling, execution, and monitoring of the decisions are not based on BPMN.

[20] introduces an Industry 4.0 process modeling language (I4PML) that extends BPMN 2.0 with the following elements: cloud app, IoT device, device data, actuation task, sensing task, human computer interface, and mobility aspect. I4PML does not include any solution for modeling, executing, and monitoring IoT-driven decisions.

None of the discussed works provides a complete approach for modeling, executing, and monitoring IoT-driven decisions. Most works extend BPMN with IoT-related elements to foster a better understanding of IoT-driven processes. When integrating IoT entities in terms of resources, however, IoT data is used directly without linking and aggregating it with other contextual data. No engine for processing IoT-driven decisions in the context of BPMN models exists to the best of our knowledge. Another missing aspect concerns the monitoring of IoT-driven decisions. During the processing of decision rules it is crucial to be able to monitor the current decision rule processing state, possible errors, intermediate results, and the structure of the decisions.

IV. Requirements

Based on various application scenarios and our literature review, we can define requirements that shall address the described problems in modeling, execution, and monitoring IoT-driven decisions in BPMN 2.0.

- **RQ1:** The modeling, executing, and monitoring of IoT-driven decisions shall be supported.
- **RQ2:** The involvement of IoT-related data in process decisions shall become apparent in the modeling, execution, and monitoring of IoT-driven decisions.
- **RQ3:** The approach shall be detached from the IoT infrastructure and support the most common protocols.
- **RQ4:** The modeling of IoT-driven decisions shall be feasible in BPMN.
- **RQ5:** IoT-driven decisions shall be executed and monitored in real time.
- **RQ6:** Sensor polling should be enabled in real time based on suitable protocols.
- **RQ7:** Non-availability, runtime errors, or timeouts of sensors must be discovered by the decision engine and be explicitly displayed during decision rule monitoring.
- **RQ8:** Any error occurring during decision rule processing shall be traceable, e.g., in terms of an event log.

- **RQ9:** Low-level sensor data shall be aggregatable to high-level information or events.
- **RQ10:** The definition of decision rules shall be enabled in a visual and intuitive form.
- **RQ11**: The defined decision rules shall be processed by the decision engine.
- **RQ12:** The results of a processed decision rule shall be used in the process.
- **RQ13:** When monitoring the execution of a decision rule, it should be possible to distinguish between erroneous sensors, sensors in execution, and sensors that have already been executed.
- **RQ14:** It shall be possible to visually display the retrieved sensor values as well as the decisions made in the monitoring component.

V. System Framework

This section introduces the main functions and architecture of the IoTDM4BPMN, i.e., our framework for modeling, executing, and monitoring IoT-driven decisions in BPMN models.

A. Main Functions

The IoTDM4BPMN framework offers four main functions to improve IoT-driven decisions during real-time:

- **Modeling** To be able to explicitly model IoT-driven decisions, BPMN 2.0 is extended with the following IoT elements; *IoT sensor artifact, IoT decision container, IoT decision table, and IoT decision task*. These elements shall make IoT involvement more explicit and, thus, fosters the distinctiveness between IoT- and Non-IoT-driven decisions.
- **Execution** IoT-driven decisions can be executed in real time by the IoTDM4BPMN decision engine.
- **Monitoring** The IoT-driven decisions can be monitored in real-time. In this context, intermediate and final results as well as the sensors involved may be displayed. Different color patterns are used to indicate the status of processing a decision rule, i.e., under execution, successfully completed, or failed.
- **Logging** To be able to trace IoT sensor failures back to the respective sources, various parameters are logged including decision logic, involved IoT devices, timestamp, process id, name of the IoT decision container, and IoT decision table.

B. IoT-related BPMN elements

To explicitly cover IoT involvement in decision modeling, we extend BPMN 2.0 with the four elements shown in Figure 3:

- **Sensor artifact:** A sensor artifact represents different sensor types relevant in the context of IoT-driven decision modeling, e.g., electrocardiography sensor, temperature sensor, GPS sensor, switch, or camera. Using this artifact, the involvement of sensors in IoT-driven decision modeling becomes apparent. Furthermore, at runtime this artifact can be queried in real-time to evaluate the condition of the respective decision rule. Note that all necessary information about the sensor (e.g., address, type, or name) is captured in this artifact during modeling [23].
- **IoT decision table and container:** For defining IoT-driven decisions, the IoT decision container may be used in combination with the IoT decision table and the sensor artifacts. The IoT decision table is part of the IoT decision container. It uses the sensor artifacts as input elements to define decision rules. Each IoT decision container may consist of *n* IoT decision containers and likewise *n* sensor artifacts.
- **IoT decision task:** The IoT decision container is connected to the IoT decision task using an association. The IoT decision task may incorporate both final (root IoT decision container) and partial decisions (child IoT decision container) into the process model (Figure 3). In addition, the IoT decision task can extend or collapse the decision logic. Finally, an IoT-driven decision is distinguishable from other decisions such as a DMN decision by the new icons.

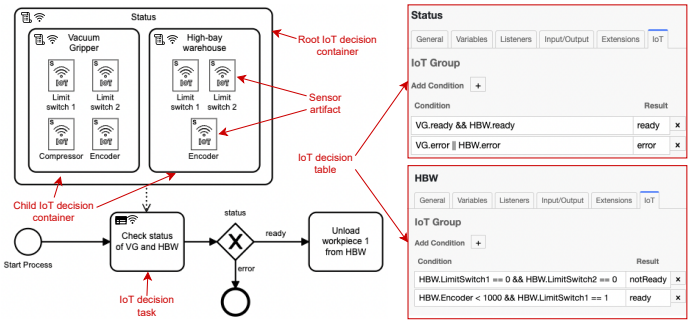

Fig. 3. Example IoT-driven decision process in BPMN in IoTDM4BPMN

Figure 3 shows an IoT-driven decision process and illustrates the above mentioned elements. The *Status* root IoT decision container includes the *Vacuum Gripper* and *High-bay warehouse* child decision container. These, in turn, contain sensor artifacts representing various physical IoT devices. Through the IoT decision table, decision rules can be defined for each IoT decision container based on the physical IoT devices (sensor artifacts). The IoT decision table allows for any mathematical comparison. For example, decision rule $HBW.LimitSwitch1 == 0 \&\& HBW.LimitSwitch2 == 0$ is defined in the *HBW* IoT decision table. When this decision rule becomes satisfied, the *HBW* IoT decision container returns result $notReady$. This result, in turn, is used by the root IoT decision container *Status* to check the following decision rule: $VG.error || HBW.error$. The result of the root IoT decision container can be subsequently used in the process by the IoT decision task. Extending BPMN with the sensor artifact, IoT decision container, and IoT decision task fulfills **RQ2**.

C. Architecture

IoTDM4BPMN is implemented as a web-based software application. This makes the framework platform-independent

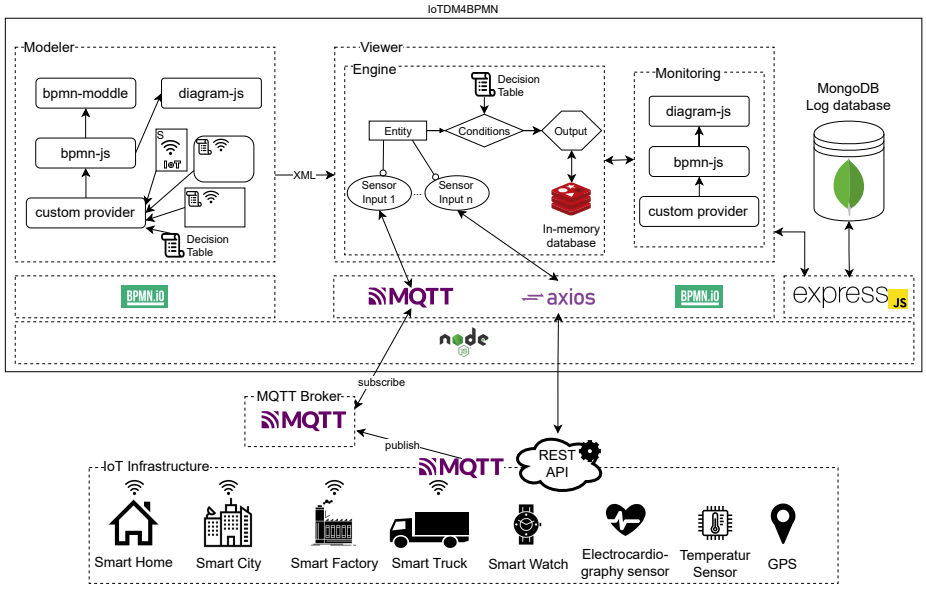

Fig. 4. IoTDM4BPMN Architecture

and accessible to the public. IoTDM4BPMN is hosted by a *Node.js*[1] web server. For modeling and monitoring IoT-driven decisions, *Bpmn.io*[2] is used, which is available as a Node.js package. The storage of the values of the queried sensors, the intermediate results of the decisions, and the final decisions are provided by the in-memory database *redis*[3]. For the parallel query of the sensors the thread pool design pattern is applied. The *workpool*[4] package supports this pattern and enables the execution of parallel tasks by offloading tasks from the main event loop to workers. The logs generated for each IoT-driven decision process by IoTDM4BPMN are stored in a MongoDB database via a REST interface. The sensors can be queried via a REST interface as well as via MQTT. Figure 4 shows the components of IoTDM4BPMN and their interactions. The individual components of the framework are described in detail below. The defined architecture fulfills **RQ1, RQ3, and RQ5**.

D. Modeler

For modeling IoT-driven decisions, the open-source *BPMN.io* library is used. This library allows extending and embedding the standard BPMN 2.0 libraries such as *bpmn-js*, *diagram-js*, and *custom provider*. *BPMN.io* is built on top of the web modeler and the rendering toolkit *bpmn-js*. The library can be used both as a modeler and as a viewer. The modeler can be used to create BPMN 2.0 diagrams within an application. To enable this, *bpmn-js* builds on two other libraries (cf. Figure 4): *digram-js* and *bpmn-moddle*; *diagram-js* extends *bpmn-js* with the renderer and the modeler component. Custom elements may be created through a *custom provider*. The customized elements of *bpmn-js* are registered with *diagram-js*, which renders them in the modeler. The *bpmn-moddle* library provides the BPMN meta model. When importing a BPMN diagram, a JavaScript object tree is parsed, which can be edited during modeling. Furthermore, the library validates BPMN 2.0 diagrams, provides suitable modeling rules, and allows exporting the revised JavaScript object tree into an XML document. The IoT decision elements introduced in Section V-B are created in *BPMN.io* as *Custom Elements*. The *Custom Renderer* is used to define the shape, color, and size of the *Custom Elements*. The *Custom Rule Provider* is then used to specify the behavior rules of the *Custom Elements*. For example, it is defined that each IoT decision container may contain *n* IoT decision containers and likewise *n* sensor artifacts. The IoT decision table is populated by the *Custom Properties Panel*. Each IoT decision container has its own IoT decision table. Clicking on an IoT decision container displays the *Custom Properties Panel* with the IoT decision table. The extension modules *Custom Elements*, *Custom Renderer*, *Custom Properties Panel*, and *Custom Rules* are passed to the Modeler by the *Custom Provider* as *additonalModules*. After modeling IoT-driven decisions in the IoTDM4BPMN Modeler, the *bpmn-moddle* library generates an XML file that is passed to the Viewer. The developed Modeler fulfills **RQ4, RQ9, and RQ10**. Figure 5 shows the Modeler. As each IoT decision container may contain *n* IoT decision containers and likewise *n* sensor artifacts, a tree structure is obtained. The conversion of this nesting in XML into a JavaScript tree can be achieved with Algorithm 1. The tree must be converted into a JavaScript tree, which can be read and executed by the IoTDM4BPMN decision engine. Furthermore, the tree contains information such as name, address, and IoT decision table, which were all defined during modeling. Finally, the dependency and execution order of the IoT decision containers can be derived from the tree. For this purpose, the root IoT

[1]https://bpmn.io/
[2]https://bpmn.io
[3]https://redis.io/
[4]https://www.npmjs.com/package/workerpool

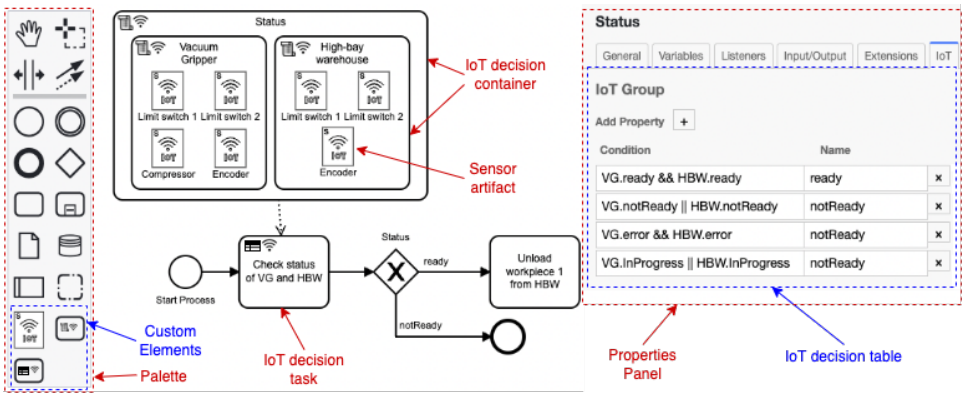

Fig. 5. IoTDM4BPMN Modeler

decision container is passed to the *mainNode* object. Then, it is checked whether *mainNode* has child elements. Afterwards, each *childNode* is iterated and it is checked whether it has type *iotDecisionContainer*. In this case, the *childNote* is pushed into the *mainNode* object and the algorithm is recursively invoked with the new root IoT decision container.

Algorithm 1 Algorithm for creating recursion tree

```
function createTree (mainNode)
 1: if mainNode has childNode then
 2:     for each childNode do
 3:         if childNode type is iotDecisionContainer then
 4:             mainNode.descendants.push(createTree(childNode))
 5:         end if
 6:     end for
 7: end if
 8: return mainNode
end function
```

The tree object built by Algorithm 1 represents the root IoT decision container as the *tree root*, the child IoT decision containers as the *inner nodes*, and the sensor artifacts as the *leaves*. While the left part (1) of Figure 6 shows the general structure of the tree, the right part (2) shows the decision tree of the IoT-driven decision process shown in Figure 5.

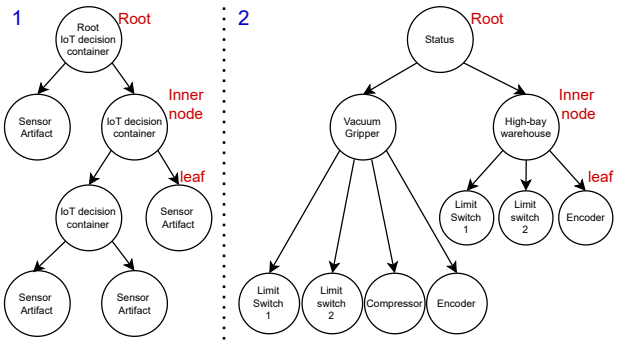

Fig. 6. Decision tree structure generated by Algorithm 1

E. Viewer

The IoTDM4BPMN viewer consists of the decision engine and the monitoring component. The XML file generated by the Modeler contains the structure of both the process model and the decision logic. The process is executed using an open-source BPMN 2.0 JavaScript workflow engine[5], which is extended by the elements introduced in Section V-B. The IoTDM4BPMN decision engine converts the XML file created by the Modeler into a JavaScript object tree. If an IoT decision container is detected during process execution, the evaluation of the modeled decision rule will be triggered. As the monitoring is directly connected to the IoTDM4BPMN decision engine, the individual steps of the decision evaluation can be displayed and monitored in real-time. The associated tree is built following a bottom-up approach. Thereby, the results of the *inner nodes* are passed from the bottom to the top until the root IoT decision container delivers a final decision. Finally, the evaluation is performed using a recursive algorithm, which is described in the following.

Consider Algorithm 2, the function *treeResult* receives as input parameter the root IoT decision container. If the latter contains sensor artifacts, these are stored in the *sensorInputs* array (cf. Algorithm 2 Lines 4-10). Then, it is checked whether the root IoT decision container comprises other IoT decision containers as descendants. In this case, the function recursively calls itself for each individual descendant (cf. Algorithm 2 Lines 11-14). Once all descendants have returned their results, the query of the sensors, starting at the lowest level of the tree, is executed according to the thread pool design pattern. The sensors can be queried via HTTP using axios or via MQTT. Their polling in real-time fulfills **RQ6**. During query processing, the sensors are colored orange (cf. Lines 15-18). If a sensor reports an error during the query processing, an error is thrown and the sensor is colored red (cf. Lines 20-23). The detection of an error and the corresponding coloring fulfills **RQ7**. After successfully querying the sensors of the deepest IoT decision container (cf. Line 19), the previously filled IoT decision table is evaluated with the *extractedDecisionSeatteldPromise* function. For this purpose, the decisions of the corresponding IoT decision container are detected from the generated tree via the ID. These decisions are then passed

[5]https://github.com/paed01/bpmn-engine

Algorithm 2 Algorithm for the execution of the decision logic

```
function treeResult (treeRoot)
 1: childrenPromises ← []
 2: workerArr ← []
 3: sensorInputs ← []
 4: if treeRoot.child is sensorArtifact then
 5:     for each bpmnViewer.elem do
 6:         if bpmnViewer.elem.id is treeRoot.child.id then
 7:             sensorInputs.push(treeRoot.child)
 8:         end if
 9:     end for
10: end if
11: if treeRoot.descendants.length > 0 then
12:     for each treeRoot.descendants do
13:         childrenPromises.push(treeResult(treeRoot.descendants)
14:     end for
15:     return Promise.allSettled(childrenPromises){
16:     if childrenPromises.rejected is 0 then
17:         extreactedDecision(iotInputs, treeRoot)
18:         highlightElement(treeRoot, orange)
19:         return extractedDecisionSeatteldPromise()}
20:     else
21:         highlightElement(treeRoot, red)
22:         return Proimise(new Error)
23:     end if
24: end if
25: extractedDecisionSeatteldPromise() {
26: return Promise.allSettled(workArr){
27: if workArr.rejected is 0 then
28:     decisionResult ← evaluateDecision(treeRoot.ID)
29:     highlightElement(treeRoot, green)
30:     addOverlay(treeRoot.vakue, decisionResult)}}
31: end if
32: return extractedDecisionSeatteldPromise()
end function
```

to the *evaluateDecision* function. The evaluation of decision rules from the IoT decision table satisfies **RQ11**. If the latter does not throw an error, the IoT decision container will be colored green, otherwise it will be colored red (cf. Lines 26-30). Coloring according to the status fulfills **RQ13**. Function *treeResult* is called recursively until all decision containers of the described procedure will have been traversed. The query and evaluation of the sensor artifacts and the IoT decision containers are performed in parallel using the pool design pattern. The results provided by the individual sensors, decision rules, and results of the individual IoT decision containers, and final decision are stored in a *redis in-memory database*, i.e. the results can be further used by the business process.

The IoTDM4BPMN decision engine is linked to the monitoring system. In order to visualize that sensors or IoT decision containers are active, they are colored orange (Figure 7) (**RQ13**). If the query and evaluation are successful, they will be colored green (Figure 7), otherwise red (**RQ13**). Furthermore, two labels with designation *Decision* and *Result* are attached to the upper corners of each IoT decision container. Hovering over the *Decision* label displays a table with all decisions and results of the IoT decision container. In turn, hovering over the *Results* label displays a table with all sensors and their results. The monitoring of the results and sensor values fulfills **RQ14**. After executing the decision rule, the result can be used in the process via the IoT decision task and thus fulfills **RQ12**. All information generated by the viewer (decision engine and monitoring) is stored in a MongoDB database and therefore fulfills **RQ8**. This database stores information such as timestamps, results of queried sensors, decision rules (IoT decision containers), intermediate decisions of each IoT decision container, final decision of the root IoT decision container, and generated tree.

Figure 7 shows an executed decision process in the monitoring view. The modeled decision logic can be either shown or hidden during the execution of the business process by clicking on the WLAN or decision icon of the IoT decision task (1). According to their status, the sensor artifacts, IoT decision container, and IoT decision task are colored. The requested sensor values as well as intermediate and final decisions of the individual containers can be displayed by hovering on labels *Decision* (2) or *Results* (3). Through the coloring and the labels (2, 3) the erroneous elements are illustrated, which facilitates error handling. In parallel to the coloring of the elements, additional information such as the HTTP request and response, MQTT request results, and evaluation of the decision rules are logged and monitored by the IoTDM4BPMN decision engine (4). In case of an error, therefore, specific information can be read off (Figure 7). Finally, the generated log is stored in the database for each decision process.

The decision process from Figure 7 refers to different sensors (e.g., limit switch, light barrier, encoder position, and compressor pressure) in order to determine the state of the high-bay warehouse and the vacuum gripper. Based on the final result of the root IoT decision container with label *Status*, either Workpiece 1 is unloaded from the high-bay warehouse or the process ends. After unloading Workpiece 1 its quality is determined based on the environmental condition that refers to a temperature sensor, humidity sensor, and brightness sensor. In case of damage, the Workpiece is transported to the post-processing station.

A video of the execution of the process from Figure 7 by the IoTDM4BPMN decision engine can be viewed on YouTube[6]. It shows the decision engine, the monitoring, and the behavior of the Fischertechnik[7], a small scale physical smart factory model, in real-time.

VI. Conclusions

With IoTDM4BPMN this paper presented a framework for web-based IoT Decision Making in BPMN. The IoTDM4BPMN framework enables modeling, executing, and monitoring of IoT-driven decisions in BPMN. Starting with problem investigation and a literature review, we were able to show that there are no approaches for modeling, executing, and monitoring IoT-driven decisions with the IoT involvement becoming apparent. Based on various application scenarios and our literature review, we defined requirements that shall

[6]https://youtu.be/Eou_HT8vmA4

[7]https://www.fischertechnik.de/

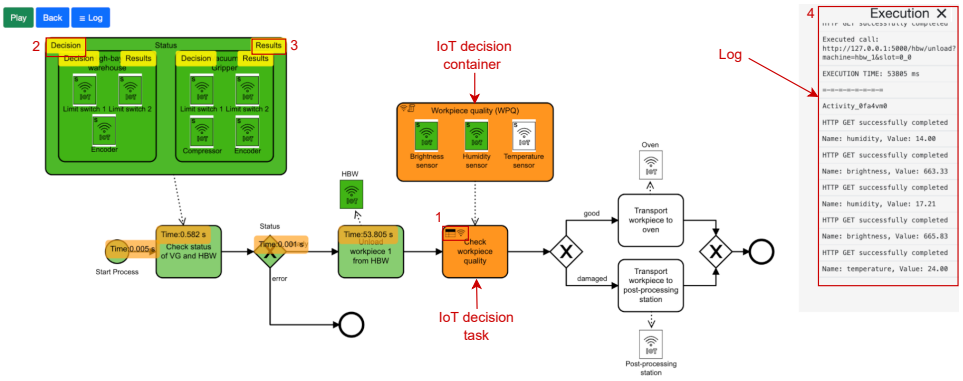

Fig. 7. IoTDM4BPMN Viewer - Decision engine and monitoring example

address the described problems in modeling, execution, and monitoring IoT-driven decisions in BPMN 2.0. For this purpose, the BPMN standard has been extended with IoT decision task, IoT decision container, sensor artifact, and IoT decision table. While the IoT decision task, the IoT decision container, and the sensor artifact make the IoT involvement apparent, the IoT decision table enables the definition of decision rules referring to physical IoT devices. The modeled IoT-driven decision processes can be passed to the IoTDM4BPMN decision engine, whose thread pool design enables execution of tasks such as querying sensors or executing the decisions in Multithreading. IoTDM4BPMN enables querying over both HTTP and MQTT. The values of the queried sensors as well as the decisions are stored in a redis in-memory database for performance reasons and can be reused during process execution. The monitoring system of IoTDM4BPMN is linked to the decision engine and can therefore display the behavior in real time. During runtime, erroneous, successful, and sensors in execution can be distinguished from each other by different color representations. Information such as sensor values, partial decisions, final decisions, execution duration, IoT decision table, and the generated trees are stored in a MongoDB database in order to be able to identify the cause in the event of an error.

In future work we will perform various experiments and studies with different users such as BPMN modelers or domain experts to investigate the completeness of IoTDM4BPMN.

Acknowledgments

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under project number 449721677.

References

[1] S. Cherrier and V. Deshpande, “From BPM to IoT,” in International Conference on Business Process, Semptember 2017.
[2] F. Hasić, E. Serral and M. Snoeck, “Comparing BPMN to BPMN + DMN for IoT process modelling: a case-based inquiry,” in 35th ACM/SIGAPP Symposium on Applied Computing, January 2020.
[3] R. Song, J. Vanthienen, W. Cui, Y. Wang and L. Huang, “Context-Aware BPM Using IoT-Integrated Context Ontologies and IoT-Enhanced Decision Models,” in IEEE Conference on Commerce and Enterprise Computing, July 2019.
[4] M. Dumas, M. La Rosa, J. Mendling and H. Reijers, Fundamentals of Business Process Management, 2nd ed., Springer-Verlag.
[5] C. Janisch *et al.*, “The Internet-of-Things Meets Business Process Management: Mutual Benefits and Challenges”, September 2017.
[6] R. Krishnamurthi *et al.*, “An Overview of IoT Sensor Data Processing, Fusion, and Analysis Techniques”, in Sensors, August 2020.
[7] A. Koschmider, F. Mannhardt and T. Heuser, “On the Contextualization of Event-Activity Mappings”, in Business Process Management Workshops, September 2018.
[8] OMG, “Business Process Model and Notation (BPMN) 2.0”, 2011.
[9] OMG, “Decision Model and Notation (DMN) 1.2”, 2018.
[10] J. Taylor, A. Fish, J. Vanthienen and P. Vincent, “Emerging standards in decision modeling”, 2013.
[11] M. von Rosing, S. White, F. Cummins and H. Man, “Business Process Model and Notation - BPMN”, The Complete Business Process Handbook, 2015.
[12] G. Aagesen and J. Krogstie, “BPMN 2.0 for Modeling Business Processes”, Handbook on Business Process Management, April 2014, pp. 219–250.
[13] M. Geiger and G. Wirtz, “BPMN 2.0 Serialization - Standard Compliance Issues and Evaluation of Modeling Tools”, EMISA, 2013, pp. 177–190.
[14] M. Weske, Business Process Management Concepts, Languages, Architectures, 2nd ed., Springer-Verlag, 2012.
[15] K. Kluza, P. Wisniewski, K. Jobczyk and A. Ligeza, “Comparison of Selected Modeling Notations for Process, Decision and System Modeling”, in Federated Conference on Computer Science and Information Systems (FedCSIS), September 2017.
[16] K. Suri, W. Gaaloul, A. Cuccuru, S. Gerard, “Semantic Framework for Internet of Things-Aware Business Process Development”, in 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2017.
[17] Cheng et al. “Modeling and Deploying IoT-Aware Business Process Applications in Sensor Networks”, in Sensors, 2019.
[18] S. Meyer, A. Ruppe, L. Hilty, “The Things of the Internet of Things in BPMN”, in Conference in Advanced Information Systems Engineering Workshops, 2015.
[19] R. Oliveira et al. “An intelligent model for logistics management based on geofencing algorithms and RFID technology,” in Expert Systems with Applications, vol 42, no.15-16, pp. 6082-6097, 2015.
[20] R. Petrasch and R. Hentschke, “Process modeling for industry 4.0 applications: Towards an industry 4.0 process modeling language and method,” in International Joint Conference on Computer Science and Software Engineering (JCSSE), 2016.
[21] Gruhn et al.: BRIBOT: Towards a Service-Based Methodology for Bridging Business Processes and IoT Big Data, (2021).
[22] F. Hasic and E. S. Asensio: “Executing IoT Processes in BPMN 2.0: Current Support and Remaining Challenges”, in Research Challenges in Information Science (RCIS), 2019.
[23] Y. Kirikkayis, F. Gallik and M. Reichert: “Towards a Comprehensive BPMN Extension for Modeling IoT-Aware Processes in Business Process Models”, in Research Challenges in Information Science (RCIS), (Accepted for Publication), 2022.