Position Papers of the Federated Conference on Computer Science and Information Systems pp. 17–22
DOI: 10.15439/2020F209
ISSN 2300-5963 ACSIS, Vol. 22

Increasing the Reusability of IoT-aware Business Processes

Robert Wehlitz, Florian Jauer, Ingo Rößner
Institute for Applied Informatics (InfAI),
Goerdelerring 9, 04109 Leipzig, Germany
Email: {wehlitz, jauer, roessner}@infai.org

Bogdan Franczyk
Leipzig University, Grimmaische Str. 12,
04109 Leipzig, Germany
Wrocław University of Economics, ul. Komandorska
118-120, 53-345 Wrocław, Poland
Email: franczyk@wifa.uni-leipzig.de

***Abstract*—The Internet of Things (IoT) is based on connected devices which are often heterogeneous in terms of supported communication protocols, interfaces and message formats. IoT-aware business processes, which are executed by process engines, are often bound to specific device types. This decreases their reusability when they are ought to be deployed in multiple IoT scenarios where the ability of supporting different device types is an important requirement. In this paper, we introduce a novel approach on how to overcome the heterogeneity of IoT devices, thus increasing the reusability of IoT-aware business processes. The contribution of this work to information systems research is twofold: First, we present a device abstraction model as the basis to define business process tasks across heterogeneous device types without the need of dealing with their technical implementations. Secondly, we propose a system architecture which supports the modeling, deployment, execution and reuse of IoT-aware business processes.**

I. Introduction

THE vision of the Internet of Things (IoT) is based on the ubiquitous utilization of electronic devices which are equipped with sensors or actuators and are connected to the Internet. Market analysts estimate that the number of these IoT devices will increase to around 38.6 billion worldwide by 2025 [1]. All of them are sources of data, which provide businesses and customers with possibilities to gain valuable insights into commercial value creation and the everyday life of people. For instance, in the smart home domain, businesses are enabled to collect and analyze more detailed data about customers and their behavior in order to individualize and improve products and services [2]. Customers, in turn, may benefit from smart home solutions enabling them to analyze and reduce their energy consumption, secure their homes, remote control appliances for more convenience or live a longer self-determined life at home in old age [3].

The integration of heterogeneous IoT devices with services, applications and business processes represents a major challenge in this context. Heterogeneity hereby means that IoT devices often support different communication protocols, interfaces to access device functionality and message formats for providing sensor data and receiving device commands [4]. In our previous work [5], we presented an architectural concept on how to cope with heterogeneity issues related to IoT-aware business processes by leveraging IoT middleware and device management functionality in conjunction with system components for business process automation. However, this approach has its limitations when reusing executable IoT-aware business processes for different IoT device types, which provide similar functionality and should therefore be interchangeable from a user perspective. In this paper, we introduce a device abstraction model which enables to define business process tasks across heterogeneous IoT device types. Furthermore, we adapt the architecture presented in [5] to support the modeling, deployment, execution and reuse of IoT-aware business processes.

The remainder of this paper is structured as follows: First, we outline the background for our research, introduce an application scenario in order to illustrate why the reusability of IoT-aware business processes can be important, and provide an overview of related work (Sect. 2). In Sect. 3, we present our device abstraction model and describe its classes as well as their relationships with each other. We then introduce our system architecture proposal and point out how an increase of the reusability of executable IoT-aware business processes is achieved (Sect. 4). Finally, in Sect. 5, the paper concludes with a short summary of the findings and an outlook on future work.

II. Background

Within the scope of this paper, we define IoT-aware business processes as sequences of tasks, events and decisions which integrate IoT devices as process resources in order to achieve a certain process goal. In this regard, we focus on processes that are described using a machine-readable format, e.g. Business Process Model and Notation (BPMN) 2.0, and can be executed by process engines for automation purposes. Against this background, business process tasks are assignable to IoT devices and can be completed by them using their sensing or actuation capabilities. IoT-aware business processes may also be consumers of events, which are detected based on processing sensor data streams and may

This work is partly funded by the European Regional Development Fund (ERDF) and the Free State of Saxony (Sächsische Aufbaubank – SAB).

have an impact on their control flow. Additionally, decisions, e.g. threshold value analysis, within a running process instance can be made automatically by evaluating sensor values against predefined rule sets.

To illustrate the reusability problem regarding executable IoT-aware business processes and to show how our approach works, we use the example of smart home as a user-centric IoT domain. In this example, smart home services, such as automatic light or heating control, are defined using a process modeling language and are executed by a process engine. IoT-aware business process models, thus, define and represent the internal logic of smart home services.

Let us assume a smart home scenario in which two rooms are equipped with smart lamps. To simplify matters, *room A* has several *smart lamps* of type *A* from *manufacturer A* and *room B* has several *smart lamps* of type *B* from *manufacturer B*. All smart lamps provide the same functionalities, which are: Turning the light on or off, change the light color and dim the light. These functionalities can be remote controlled via a Wi-Fi connection. However, *smart lamps* of type *A* require an additional gateway device which receives commands over the Message Queuing Telemetry Transport (MQTT) communication protocol whereas each *smart lamp* of type *B* provides a REST-API in order to be controllable via HTTP requests. Additionally, the accepted message format of each interface differs strongly between *smart lamps* of type *A* and *B*. For instance, the gateway for *smart lamps* of type *A* expects the value "1" to be published to a specific internal MQTT topic structure for turning on a lamp. The *smart lamps* of type *B*, for obtaining the same effect, expect a JavaScript Object Notation (JSON) payload including the property "light" with the value "on" to be posted to a specific REST endpoint.

In order to hide this complexity from developers of smart home services, we use the method of abstraction. Therefore, we distinguish between three different device abstraction levels.

At level zero, the device instance level, a specific device of a certain user is described and no further abstraction is made, e.g. *smart lamp* of type *A* with ID A132 of *user U*.

At level one, the device type level, device instances of the same type are abstracted to a device type, e.g. *smart lamp* of type *A* from *manufacturer A*.

At level two, the device class level, heterogeneous device types, as described in our example above, are mapped by a device class, e.g. *smart lamps* of type *A* from *manufacturer A* and *smart lamps* of type *B* from *manufacturer B* are elements of the device class *smart lamp*.

When an IoT-aware business process in our smart home scenario is to include smart lamps, for instance, to control the room light automatically, a decision has to be made at which abstraction level the devices should be integrated. This has a strong impact on the reusability of the whole process. Modeling business process tasks for IoT devices at the device instance level requires the same process to be modeled twice: For *room A* and *B*. In addition, the process model has to be adapted and redeployed every time this setting changes even slightly. This is the case if, for instance, a smart lamp is broken at some time and has to be replaced by a new one. To increase the reusability of the defined process, smart lamps should be integrated at least at the device type level. Nonetheless, the process needs to be modeled for each room due to the different types of smart lamps, but a replacement of single device instances would not affect the model scope. However, the highest degree of reusability is achieved by modeling process tasks for IoT devices at the device class level. In this case, the process needs to be modeled only once to be deployable for both, *room A* and *B*. Moreover, an additional integration of *smart lamps* of type *C* from *manufacturer C* at a later time would be covered by the model scope as well, if those lamps also support the defined functionalities of the device class *smart lamp*.

Research in the field of IoT-aware business processes is still at its beginning. A state-of-the-art report carried out by [6] shows that numerous publications which focus on the modeling of IoT services (e.g. [7]) and IoT-aware business processes exist (e.g. [8]). However, technical aspects regarding the utilization of heterogeneous IoT devices within executable business processes have not yet been sufficiently investigated. Although some approaches cover implementation and execution aspects, they are mostly limited to the integration of wireless sensor networks [4], which do not provide actuation capabilities, or are based on translations from BPMN to program code (e.g. [9]). Therefore, the concrete implementation of abstract process models in order to deploy and execute them in IoT scenarios remains a major challenge, which is still open [10]. In our previous work [5], we already tackled this challenge by designing a system architecture, which combines an IoT middleware, device management and components for business process automation. However, this approach, like many other related works, has the limitation that IoT devices can only be integrated as business process resources at the device type level, which has a negative impact on the reusability of IoT-aware business processes (c.f. the example above). For this reason, we have improved our concept and extended it by a device abstraction model, which covers syntactical and semantic aspects of IoT devices. With regard to the latter, we are aware that many semantic middleware solutions and ontologies, e.g. SensorML and Semantic Sensor Network (SSN), for IoT systems exist. But in order to enable IoT device integration in business processes at the device class level, we had to design our own model which is partly based on classes and properties of the oneM2M Base Ontology (www.onem2m.org). This ontology, compared to others, proved to be the most suitable starting point for mapping concrete service implementations with generic device functions.

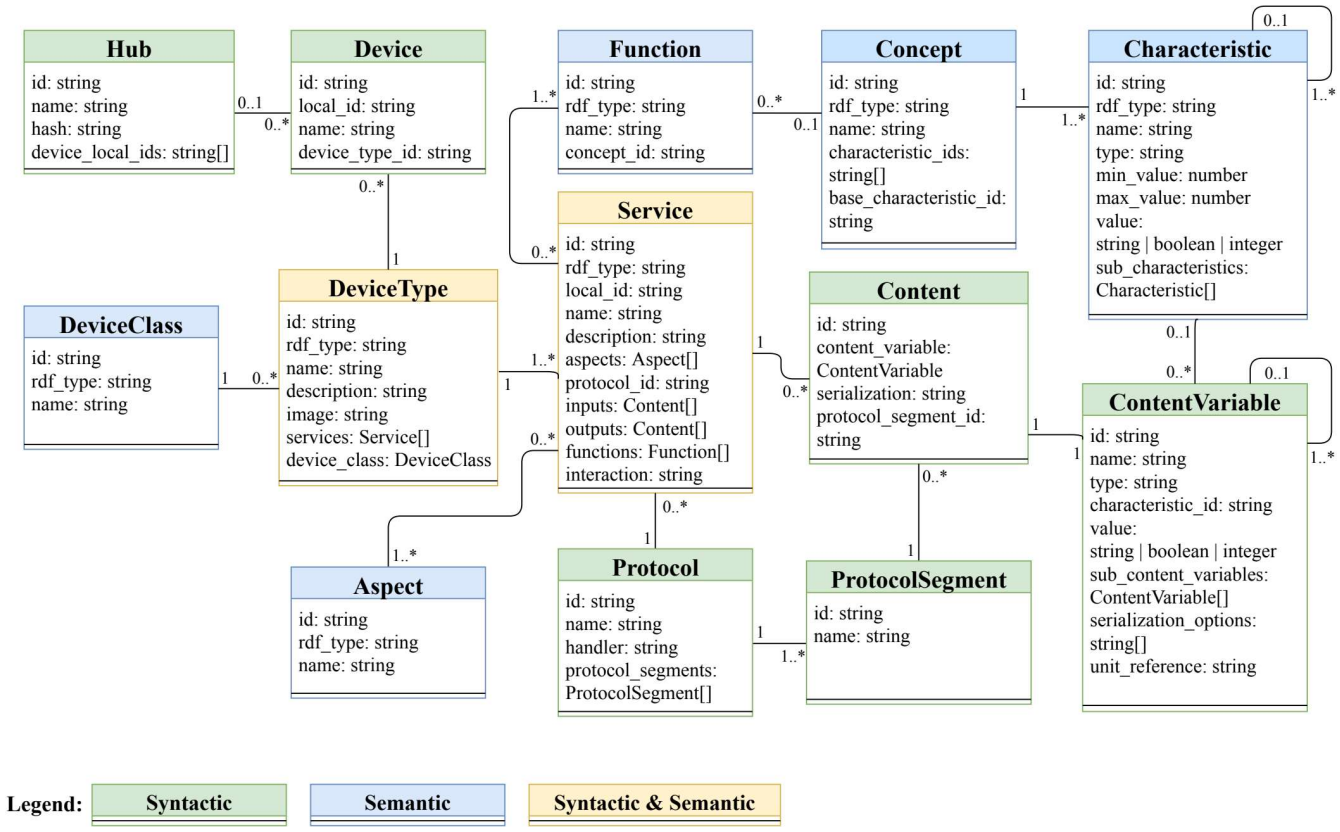

Fig. 1 Class diagram of the device abstraction model

III. Device Abstraction Model

After we outlined the background for our research and gave an overview of related work in the previous section, we present our device abstraction model in Fig. 1.

Our device abstraction model consists of 13 classes which are used to map properties of IoT devices to an internal data structure. In this context, we distinguish between syntactical and semantic aspects of the model. Syntactical aspects address technical details on, for instance, required message formats and are highlighted green in the diagram. Semantic aspects, highlighted in blue, address context information about, for instance, what is measured by a sensor, respectively, what is moved or controlled by an actuator. Classes which are highlighted in yellow cover both, syntactical and semantic aspects.

Diving into the model, a *DeviceClass* (e.g. *smart lamp*) represents a set of *DeviceTypes* (e.g. *smart lamps* of type *A* from *manufacturer A* and *smart lamps* of type *B* from *manufacturer B*) which provide similar functionalities (e.g. change the light color). A *DeviceType*, in turn, represents a set of similar physical *Devices* (e.g. *smart lamps* of type *A* from *manufacturer A* with *local_ids* from A1 to A9999) which may be connected to a *Hub*, also known as gateway, for data transmission. A *DeviceType* has one or more individually implemented *Services* (e.g. *setColorService*) which expose one or more generic *Functions* (e.g. *setColorFunction*) and support event- or/and request-based *interactions*. A *Service* has to be linked to an *Aspect* that is a real-world phenomenon (e.g. room lighting) it relates to. A *Service* also may have several inputs (e.g. color value) and outputs (e.g. response code). Both require a *Content* (i.e. a payload) which is serialized (e.g. using XML or JSON) for data exchange, placed within a *ProtocolSegment* (e.g. header or body) and transmitted over a *Protocol* (e.g. HTTP or MQTT). A *Content* consists of one *ContentVariable* which describes the internal structure of a *Content* (e.g. { “name”: “color”, “type”: “string” }) and can be nested in order to map more complex structures. In the case of an XML serialization, the *serialization_options* field of a *ContentVariable* can be used to define whether it is an XML attribute or the content of an XML tag. *ContentVariables* can be linked to a *Characteristic* which can be nested as well. *Characteristics* (e.g. color definitions, such as RGB, HSB or Hex) define different ways of how a *Concept* (e.g. color), which is a property of an *Aspect* (e.g. room lighting), can be represented. Therefore, it is also possible to declare a base characteristic for each *Concept*. Finally, a *Concept* can be linked to *Functions* in order to indicate which property of an *Aspect* is measured or changed by a *Function* (e.g. *setColorFunction* changes the *Concept* color of *Aspect* room lighting).

The presented device abstraction model enables its users to map the most important syntactical properties of IoT devices and to enrich them with useful context information. Furthermore, the main advantage of the model is that concrete service implementations are abstracted and, thus, de-

coupled from generic IoT device functionalities, which, in turn, allows defining and using abstract device classes. Hence, heterogeneous IoT devices, which support different communication protocols, interfaces and message formats, can be used in different deployments of the same IoT-aware business process.

IV. System Architecture

In the previous section, we presented our device abstraction model and described its classes as well as their relationships with each other using the example of smart lamps. In the following, we introduce our architectural concept for an IoT-aware Business Process Management (BPM) system (see Fig. 2), which instantiates the device abstraction model to support the modeling, deployment, execution and reuse of IoT-aware business processes. The architecture comprises 13 different components (shown in white) whereby twelve are supposed to run in a cloud environment and one is deployed on IoT devices (shown in grey) in local networks.

The process designer component enables the graphical modeling of executable IoT-aware business processes with a suitable notation and metamodel, such as BPMN 2.0. It generates machine-readable definition files of process models according to drawn process diagrams. To define a business process task for IoT devices, the user selects the task element and assigns a generic *Function* (c.f. Sect. 3) to it. The process designer retrieves *Functions* and other semantic metadata about IoT devices, i.e. *DeviceClass*, *Aspect* and *Concept,* from the semantic repository and enriches the machine-readable definition files with them. This allows composing and reusing business process models at design time independently of the implementation details of device types

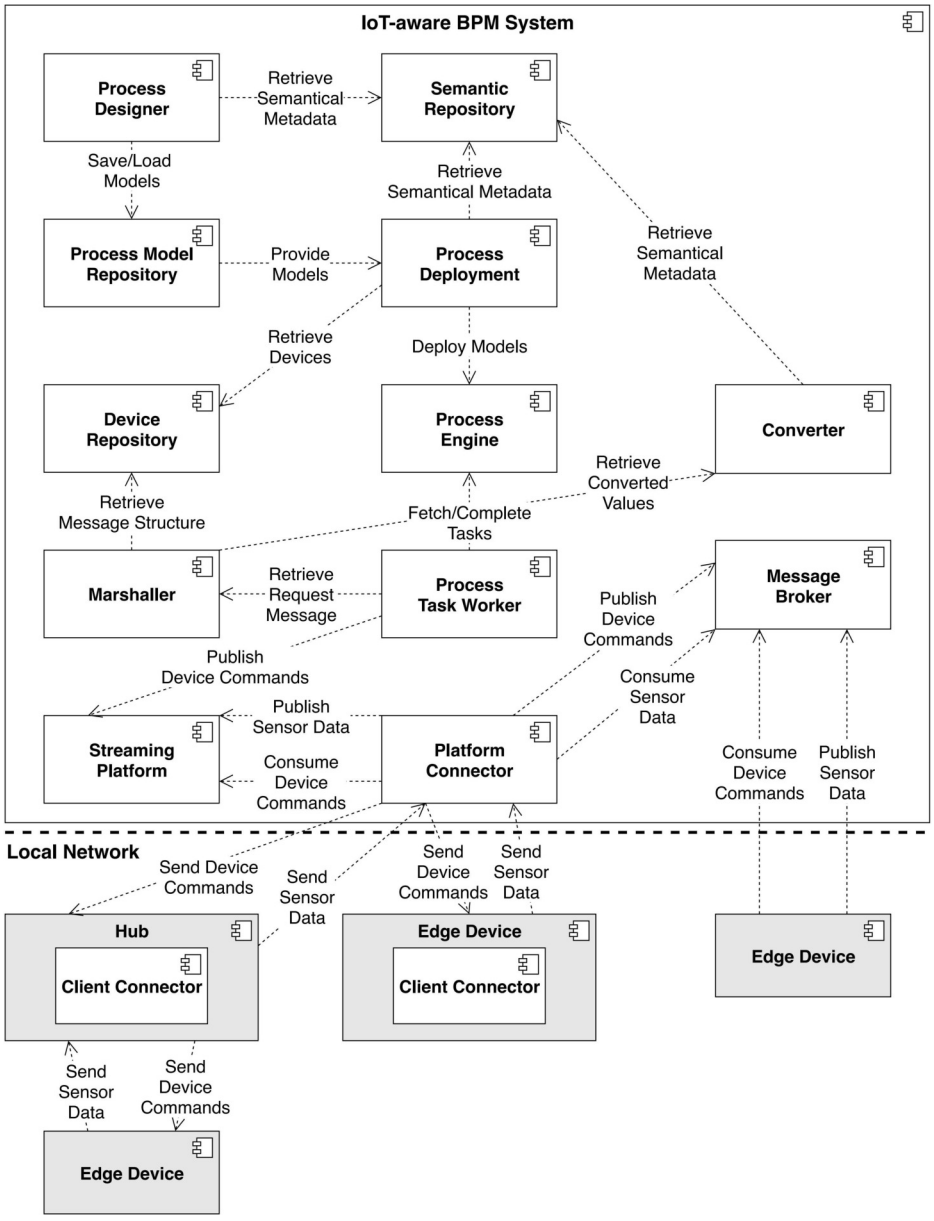

Fig. 2 Component diagram of the IoT-aware BPM system

and instances. Furthermore, it enables to deploy the same business process model multiple times for device instances of different device types.

Created model definitions are stored in the process model repository and provided to the process deployment component for implementation purposes. The process deployment component parses definition files and identifies for each business process task which concrete user devices, that are registered in the device repository, are able to complete the task. In order to achieve this, the semantic repository is queried to match metadata about *DeviceClasses*, *Functions*, *Aspects* and *Concepts* with metadata about *DeviceTypes* and *Services*. The cross-type result set, which contains device instances that are able to execute the task, is presented to the user who then can make the task assignment and define input values for *Services* (e.g. the color for *setColorService*), if required. Thereafter, the business process model is enriched with metadata about device types and instances (i.e. IDs of *Device*, *DeviceType* and *Service*) and deployed to the process engine, which ensures that process instances are executed according to the underlying model.

The business process task execution is done by the process task worker component. It fetches pending execution jobs from the process engine and informs it as soon as a job has been completed allowing the engine to proceed with the next process step. The process task worker also requests device type-specific message formats (i.e. *Content* with *ContentVariables*) from the marshaller component and publishes them together with device instance-related data (i.e. ID of the *Device*) as device commands to the streaming platform. The marshaller component can pass input values required by a *Service* to the converter component, which responds with values that were converted from one *Characteristic* to another (e.g. conversion of RGB values into HSB). This is necessary if the format or unit of input values given by an application is different from those expected by the target device. The streaming platform provides an interface by which the platform connector, that is responsible for handling the communication between the IoT-aware BPM system and edge devices, consumes pending device commands and forward them to client connectors. Client connectors run on edge devices themselves or on hub devices. They are responsible for the registration and discovery of IoT devices and may also pass sensing data from local networks on to the IoT-aware BPM system. Furthermore, they forward device commands to edge devices. The platform connector publishes incoming sensor data to the streaming platform where they can be consumed by other platform services or applications. In the case of IoT devices which use publish/subscribe-based communication protocols, such as MQTT, a suitable message broker can serve as an intermediary component to forward device commands and sensor data between the platform connector and edge devices.

V. Conclusion and Outlook

In this paper, we highlighted the challenge of coping with heterogeneous IoT devices in a business process context. For this, we used the example of smart home as a user-centric IoT domain in which executable IoT-aware business processes define and represent the internal logic of smart home services. We introduced three different levels of device abstraction (device instance, device type and device class) for overcoming heterogeneity issues and discussed how each of them affect the reusability of IoT-aware business processes. We then presented a device abstraction model for mapping the properties of IoT devices to a data structure which covers both, semantic and syntactical aspects. Moreover, it enables the decoupling of generic device functionalities from concrete service implementations which, in turn, increases the reusability of executable IoT-aware business processes. Afterwards, we introduced an architecture for an IoT-aware BPM system, which instantiates the device abstraction model in order to support the modeling, deployment, execution and reuse of IoT-aware business processes.

In the future, we want to evaluate our approach by piloting a software prototype of the IoT-aware BPM system in conjunction with a larger amount of real-world IoT devices. In this context, other application domains than smart home, such as smart manufacturing and Industry 4.0, might be of interest. Against this background, we expect to gain more insights into the applicability, scalability and flexibility of the device abstraction model and the system architecture. Furthermore, we want to extend the architecture in order to achieve interchangeability of IoT devices, not only at the process deployment stage but even at process runtime. This would be beneficial if, for instance, an IoT device has a malfunction and has to be replaced instantly by another suitable one during a running process instance.

References

[1] Strategy Analytics, “Number of internet of things (IoT) connected devices worldwide in 2018, 2025 and 2030 (in billions)”, Statista Inc., https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology, last accessed 2020/07/17.

[2] S. Khoshafian, D. Schuerman, “Process of Everything”, in *iBPMS – Intelligent BPM Systems – Impact and Opportunity* L. Fischer, Ed. Future Strategies Inc., Lighthouse Point, 2013, pp. 67–82.

[3] Zion Market Research, https://www.zionmarketresearch.com/news/smart-home-market, last accessed 2020/07/17.

[4] C. Chang, S. N. Srirama, R. Buyya, “Mobile Cloud Business Process Management System for the Internet of Things”, *ACM Computing Surveys*, vol. 49, no. 4, pp. 1–42, 2016. doi:10.1145/3012000

[5] R. Wehlitz, I. Rößner, B. Franczyk, “Integrating Smart Devices as Business Process Resources – Concept and Software Prototype”, in *Service-Oriented Computing – ICSOC 2017 Workshops*, L. Braubach, J. M. Murillo, N. Kaviani, M. Lama, L. Burgueno, N. Moha, M. Oriol, Eds. Heidelberg: Springer-Verlag, LNCS, vol. 10797, pp. 252–257, 2018. doi:10.1007/978-3-319-91764-1_20

[6] N. Brouns, S. Tata, H. Ludwig, E. S. Asensio, P. Grefen, “Modeling IoT-aware Business Processes – A State of the Art Report”, IBM Research Division, San Jose, 2018.

[7] G. Fortino, W. Russo, C. Savaglio, M. Viroli, M. Zhou, “Modeling Opportunistic IoT Services in Open IoT Ecosystems”, in *WOA*, pp. 90–95, 2017.

[8] S. Meyer, A. Ruppen, L. Hilty, “The Things of the Internet of Things in BPMN”, in Advanced Information Systems Engineering Workshops, A. Persson, J. Stirna, Eds. Springer International Publishing, pp. 285–297, 2015. doi:10.1007/978-3-319-19243-7_27

[9] F. Martins, D. Domingos, “Modelling IoT behaviour within BPMN Business Processes”, in Procedia Computer Science, Elsevier B.V., pp. 1014–1022, 2017. doi:10.1016/j.procs.2017.11.131

[10] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, A. Gal, U. Kannengiesser, F. Mannhardt, J. Mendling, A. Oberweis, M. Reichert, S. Rinderle-Ma, W.-Z. Song, J. Su, V. Torres, M. Weidlich, M. Weske, L. Zhang, “The Internet-of-Things Meets Business Process Management: Mutual Benefits and Challenges”, arXiv eprint, 2017. arXiv:1709.03628