Bus Inf Syst Eng 65(1):25–48 (2023)
https://doi.org/10.1007/s12599-022-00770-y

RESEARCH PAPER

Towards an Interdisciplinary Development of IoT-Enhanced Business Processes

Pedro Valderas · Victoria Torres · Estefanía Serral

Received: 11 December 2021 / Accepted: 28 June 2022 / Published online: 31 August 2022

Abstract IoT-enhanced Business Processes make use of sensors and actuators to carry out the process tasks and achieve a specific goal. One of the most important difficulties in the development of IoT-enhanced BPs is the interdisciplinarity that is demanded by this type of project. Defining an interdisciplinary tool-supported development approach that facilitates the collaboration of different professionals, with a special focus on three main facets: business process requirements, interoperability between IoT devices and BPs, and low-level data processing. The study followed a Design Science Research methodology for information systems that consists of a 6-step process: (1) problem identification and motivation; (2) define the objectives for a solution; (3) design and development; (4) demonstration; (5) evaluation; and (6) communication. The paper presents an interdisciplinary development process to support the creation of IoT-enhanced BPs by applying the Separation of Concerns principle. A collaborative development environment is built to provide each professional with the tools required to accomplish her/his development responsibilities. The approach is successfully validated through a case-study evaluation. The evaluation allows to conclude that the proposed development process and the supporting development environment are effective to face the interdisciplinary nature of IoT-enhanced BPs.

Keywords IoT · Microservices · Business process · Interdisciplinarity

1 Introduction

Business Processes enhanced with the Internet of Things, namely IoT-enhanced BPs, make use of IoT devices to carry out the tasks required to achieve a process' goal (Torres et al. 2020). These devices rely on the so-called *context*, i.e., relevant data from the physical world (Abowd et al. 1999; Dey 2001), to perform either sensing or actuating tasks over it. Sensors are IoT devices that are used to collect and transfer data about the process execution context (e.g., temperature sensor, camera, heart rate sensor, etc.). They provide Business Processes (BPs) with real-time data to take more informed decisions based on context (Janiesch et al. 2020). Actuators are IoT devices that can control context conditions (e.g., air conditioner, heating, watering systems, security systems, etc.). They can be used as digitized physical resources that join processes as artificial actors to automate and improve the execution of some of their tasks (Beverungen et al. 2021).

In this work, we present a tool-supported approach to help interdisciplinary development teams to develop IoT-enhanced BPs. This approach builds on top of a microservice architecture to provide a collaborative environment that supports the integration of different developers in the same project (Fowler 2015) by applying the Separation of Concerns (SoC) principle (Hürsch and Lopes 1995).

Accepted after one revision by Pnina Soffer.

P. Valderas (✉) · V. Torres
VRAIN Institute, Universitat Politècnica de València, Camí de vera S/N, 46022 Valencia, Spain
e-mail: pvalderas@vrain.upv.es

V. Torres
e-mail: vtorres@vrain.upv.es

E. Serral
LIRIS, Louvain, KU, Belgium
e-mail: estefania.serralasensio@kuleuven.be

To do so, a Design Science Research (DSR) (Hevner et al. 2004; Peffers et al. 2007) is applied. Following this research methodology, Sect. 1.1 identifies and motivates the problem while Sect. 1.2 defines the objectives of the solution and characterizes the artifact we have developed to support these objectives. Afterwards, Sect. 1.3 describes the rest of the stages proposed by the DSR methodology, and how they have been applied to design, develop, and evaluate the proposed artifact.

1.1 Problem Identification and Motivation

Let us illustrate the problem we face in this paper with an example from the logistics domain, specifically the transport of perishable products whose safety and quality highly depend on controlling temperature and humidity from the origin (harvest fields) to consumption (Bowman et al. 2009). Imagine the part of the process implemented at a smart distribution center, where the received products from warehouses are distributed to supermarkets. Following the quality control proposal presented in Valero and Ruiz-Altisent (2000), perishable products are checked and stored prior to their distribution. Figure 1 presents the flow of such a process defined with the Business Process Model and Notation (BPMN 2010). Note how this model includes several service tasks (units of work implemented by a web service or an automated application) that correspond with the actions that IoT actuators will automatically do.

The process starts when a container with a pallet of products arrives at the smart distribution center. The first thing to do is to identify the type of product being received (e.g., product name, product variety, harvest date, etc.). This is done automatically by reading the labels of the container's pallet (e.g., a QR code). The conditions in which the products have been transported, i.e., the container's temperature and humidity, are also automatically sensed. Next, a worker checks the quality of the products of the pallet (e.g., level of firmness, color, and possible damages) and introduces them in the system. Based on this quality evaluation and the conditions sensed previously, the products are considered in good quality or not for distribution. If not, the rejection of the pallet is registered and it is discarded by moving it to a garbage. On the contrary, if the quality of the products is good for consumption, the pallet is registered in the distribution center and placed into a transportation line to be stored in a refrigerator acclimatized accordingly to avoid product spoilage (e.g., oranges must be kept between 2 and 12 Celsius degrees and at 90% relative humidity).

Besides this first product control, a second one is performed over a sample in the laboratory. This analysis will determine whether molds, yeast, and certain bacteria has grown in the received products. If so, an alarm is activated, and the pallet is discarded by transporting it to the garbage. If no bacteria are detected, the shipment task of the received products can start. If the quality of the products is not excellent (e.g., they are good for distribution but firmness or color are not the optimum), the price of the products is reduced and the pallet is prioritized to avoid their spoilage. Finally, all shipped pallets are registered in the system once a truck for transporting them is available.

The diagram shown in Fig. 1 involves tasks that must be manually done by humans (e.g., select sample to analyze), tasks that humans must do interacting with a system (e.g.,

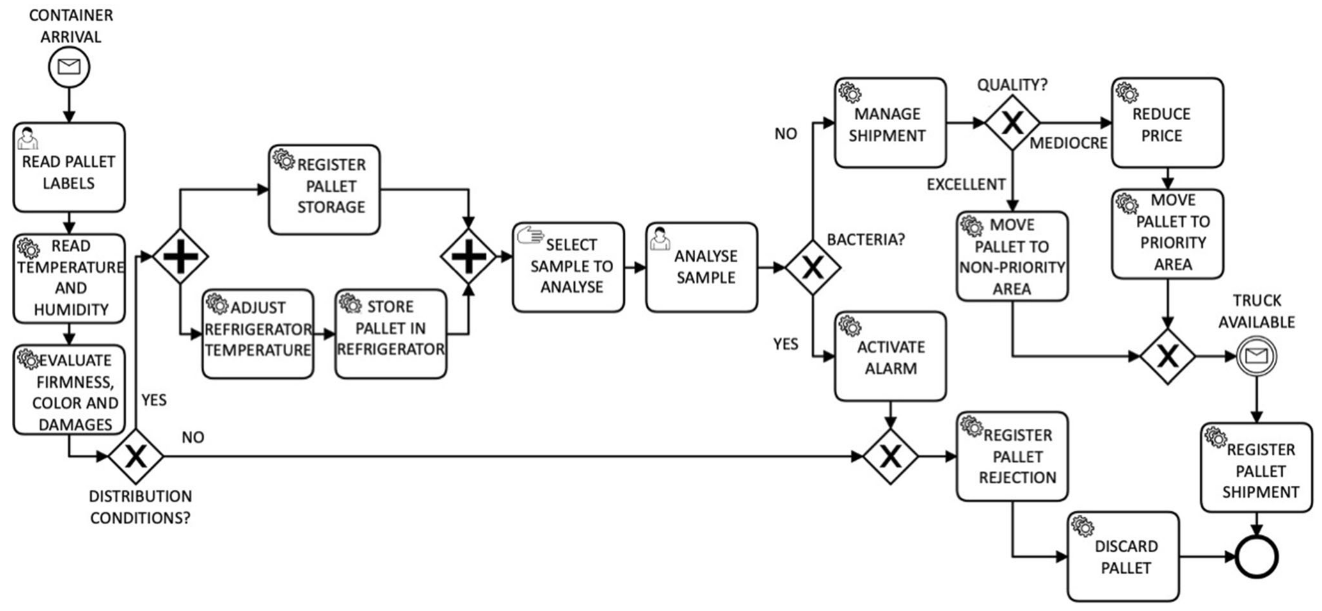


Fig. 1 BPMN representation of an IoT-enhanced BPs

evaluate the level of firmness, color and damage or analyze sample) and tasks that can automatically be done by software applications or IoT devices (e.g., sense temperature, and humidity values, move the pallet to a refrigerator camera, or activate an alarm).

Process descriptions such as the one presented above are generally created by business engineers who are experts in describing the requirements that IoT-enhanced BPs must satisfy. To do so, high-level modeling languages such as BPMN, BPEL, Petri Nets, or EPC are typically used.

If we want these processes to be executable, not only the process will need to be deployed in a process engine, but also the IoT devices must be set up and configured. However, business engineers may not have the knowledge and skills to manage the technology required to interact with the IoT devices that participate in the process (e.g., robots in charge of moving pallets to the refrigerator or shipment area, QR readers used to read the labels of the received pallets, and sensors used to capture the temperature and humidity of containers). Note that each of these IoT devices may require managing different technology (e.g., MQTT, Zigbee, Bluetooth or CoAP) in order to interact with them.

As such, to support the execution of IoT-enhanced BPs experts on IoT technologies must be involved in order to implement mechanisms that facilitate the interaction between IoT devices and the BP executed by a process engine. These mechanisms must allow 1) the process to consume the data produced by the IoT sensors and 2) the interaction between the process tasks automated by IoT devices with the actual IoT actuators. These mechanisms are usually based on either web services that can be invoked by the process engine or programming artifacts implemented in the technology supported by each engine. In addition, most IoT devices use a proprietary approach to represent the data that is provided, making it difficult their interaction with a process engine. This poses an interoperability challenge: a process engine needs to "understand" the data produced by IoT devices independently from the format used to provide this data.

Also, note that IoT sensors usually generate low-level data that are not interesting from the process point of view. Instead, some processing of these data may be required to generate higher-level information that is of interest to the process. For instance, a Bluetooth beacon may detect that an object is close to it, but this low-level data should be integrated with other data such as the type of the object (e.g., a Container) and the status of the object (e.g., its content has not been processed yet) in order to generate the high-level event "Container Arrival" that starts the IoT-enhanced BP above presented. This issue is related to the area of Complex Event Processing (CEP) (Etzion and Niblett 2010): IoT devices generate raw data that need to be processed and transformed into high-level events in order to be injected into the process. The processing of this data should be done by experts on CEP techniques that allow them to process different sources of low-level context data (e.g., data generated by IoT devices, by the process itself and by other data sources such as system logs) to generate high-level events.

Considering the above motivation, we can conclude that the development of IoT-enhanced BPs requires experts from different disciplines to collaborate. Thus, the problem that this work tries to address can be stated by the following research question: How can we support the collaboration of BP experts, IoT experts and CEP experts to improve the development of IoT-enhanced BPs?

1.2 Objectives of the Solution and Artifact Characterization

As happens with any software project that integrates IoT technologies (Khan et al. 2017), the management of IoT-enhanced BPs turns difficult due to the high complexity, heterogeneity, cost explosion, and blurred boundaries that exist between the physical and the virtual world, but also because of the interdisciplinarity nature that we find within and outside organizations. In this work, we focus on the interdisciplinarity that is demanded by this type of projects, which require software development solutions with a holistic vision that helps to integrate different disciplines (Chapline and Sullivan 2010).

In particular, the main goal of this work is to answer the research question presented above by providing an interdisciplinary solution for managing the presented three facets: definition of executable BPs; interoperability between IoT devices and the BP; and the processing of low-level context data. Note that other facets of an IoT-enhanced BP, such as security, privacy, usability, etc. are out of the scope of this work. For more information on other interdisciplinary aspects, we refer to, for instance, Choo et al. (2021).

The proposed interdisciplinary solution to implement IoT-enhanced BPs should facilitate the collaborative participation of different professionals in the development process. To do so, each professional must be supported in such a way they can face their development responsibilities independently from the others. To achieve this, the development responsibilities should be coordinated according to a specific development process and professionals should be provided with tools that facilitate the pragmatic execution of this process in a collaborative way.

In order to provide a solution that satisfies the above-introduced objectives, we present a tool-supported development process to help interdisciplinary teams to develop IoT-enhanced BPs. This process considers the three facets of an IoT-enhanced BP presented above, proposes techniques and tools to develop them in an independent but collaborative way, assigns the development responsibilities to the corresponding professional profiles, and specifies the sequence in which development activity must be done. Considering the artifact classifications presented in Hevner et al. (2004) and Peffers et al. (2012) this development process can be classified as a method since it defines a set of actionable instructions that are conceptual and not algorithmic.

1.3 Research Methodology and Paper Structure

A methodology in line with the precepts of Design Science Research (DSR) (Hevner et al. 2004; Peffers et al. 2007) is used for this research project. DSR aims at developing practical solutions that can be used by professionals in their field. More concretely, solutions – or design artifacts – can take the form of constructs, models, methods, or instantiations (Hevner et al. 2004; Peffers et al. 2012). In the previous section, we introduced the artifact developed in this work (the tool-supported interdisciplinary development process) and characterized it as a method.

We applied the DSR methodology to develop this artifact and performed the six activities proposed by Peffers et al. (2007) by following a problem-centered approach. These six activities are: (1) Problem identification and motivation; (2) Define the objectives for a solution; (3) Design and development; (4) Demonstration; (5) Evaluation; and (6) Communication.

Thus, in the first activity we identified the specific research problem and motivated it, which was presented in Sect. 1.1. This motivation is complemented by the study of the state-of-the-art that is presented in Sect. 2, which compares the improvements introduced by our solution with pre-existing ones. Next, in the second activity we defined the objectives of the solution, which have been presented in Sect. 1.2.

Then, in the third activity we designed and developed the artifact required to support the proposed objectives. To this end we have developed a tool-supported interdisciplinary development process to support the construction of IoT-enhanced BPs, which is presented in Sects. 3, 4 and 5. First, we studied how the Separation of Concerns (SoC) principle can help us to achieve our purpose and took some technical decisions to do so. These technical decisions were justified and founded on the existing literature. In addition, their application to the motivation example was used as a method to ensure they were adequate to achieve the defined objectives. This is explained in Sect. 3. Once the technical issues regarding the application of the SoC principle were resolved, we proposed the sequence of steps that were required to define the three considered facets of an IoT-enhanced BPs. To do so, we analyzed the professional profiles that can oversee the responsibility of developing each facet, studied the dependencies among facets, and specified the sequences of steps that professionals should follow to successfully create an IoT-enhanced BP. This is explained in Sect. 4. The study of the technical decisions to apply the SoC principle as well as the creation of the interdisciplinary development process were done by following an action research development (Avison et al. 1999) in such a way we iteratively studied the problem to solve, applied some actions, and analyzed if the obtained results satisfy our purposes. Once this was done, we implemented a tool environment that supports professionals in the application of the proposed development process. It is presented in Sect. 5. To implement these tools, we sough open-source solutions that could support our needs and extended them as needed. This implementation activity was done by following an iterative and incremental process (Larman and Basili 2003) in such a way the tools were progressively developed and tested with examples, refining previous implementations when some errors were detected.

In the fourth activity (Demonstration), the developed artifact must be used to solve one or more instances of the considered problem. This was done using the tool environment to develop some examples and demonstrate the feasibility of the interdisciplinary development process. This is illustrated with the implementation of the motivation example shown when presenting the tool environment in Sect. 5.

In the fifth activity we must observe and measure how well the artifact supports a solution to the problem. To do so, we arranged a controlled subject-based experiment (Hevner et al. 2004; Peffers et al. 2012) in order to evaluate the effectiveness of the developed artifact to allow users to solve an instance of the problem considered in this work. The experiment was conducted by applying the guidelines proposed by Runeson and Höst (2009). It is presented in Sect. 6

Finally, in the sixth activity we are communicating our results to the research community through this paper, whose last section presents some conclusions and provides insights into directions for future work.

Besides, to conclude the description of how the DSR has been applied to this research work, the seven guidelines proposed by Hevner et al. (2004) are used to characterize the obtained results (see Table 1).

2 Related Work

There are many works in the literature that face the modeling of IoT-enhanced BPs (Torres et al. 2020). However, none of them faces explicitly the modeling of such systems from an interdisciplinary point of view. Nevertheless, some of these works apply the SoC principle to define their modeling proposal, which we think is key to addressing the interdisciplinary nature of IoT-enhanced BPs. For example, all these works propose different models that by being combined provide a complete description of the system. While in some cases the approaches make use of ontologies to represent a specific part of the system, others just rely on different types of models that are designed expressly for a specific purpose.

On the one hand, within the category of works that make use of ontologies we find the works of: Albreshne and Pasquier (2015) that propose combining the GPL4SRE (the modeling proposal based on BPEL4WS to describe BPs) with the ontology Ont4SRE to describe "smart objects"; Dörndorfer and Seel (2018) that propose combining an extended version of BPMN (Context4BPMN) and the SenSoMod model to specify sensors, context and how these relate to each other; Gao et al. (2011) that propose linking BPMN models with the Functional Model to import a sensor ontology and its instance data; Sasirekha and Swamynathan (2016) that propose combining BPEL models with an ontology that defines IoT entities and that is integrated with the SSN ontology; Serral et al. (2015) that integrate CPN with OWL ontologies to describe context; Suri et al. (2017) that propose providing a semantic description of the BPMN models by means of an ontology that integrates concepts from the BP and the IoT domains. This integrated ontology is built from IoTBPO (an ontology defined from the IoT resources (Sensor, actuator, and tag) included in the defined BPMN extension), BPMO (Dimitrov et al. 2007), and IoT-Lite (Bermudez-Edo et al. 2017) ontologies. On the other hand, within the category of works that combine different types of models we find the works of: Baresi et al. (2016) that propose the use of an extended Guard-Stage-Milestone (GSM) model to model and monitor the BP part that refers to goods that are moving from different organizations. These goods are turned into smart objects since these are equipped with software running, sensing data and communication capabilities; Tu et al. (2018) that propose to combine three models specified at three different layers to represent separately the domain (IoTCM), the process (IoTPM), and objects (IoTOM); Yousfi et al. (2019) that propose combining their BPMN extended proposal (uBPMN) with a Decision Model where ubiquitous decisions, i.e., decisions taken based on an important amount of data (e.g., location, traffic status, gas level, etc.) are defined to improve the BP.

As a summary, Table 2 presents the most important characteristics of the above-introduced approaches regarding the application of the SoC principle to develop IoT-enhanced d BPs. Note that none of these works explicitly consider the three facets supported by our approach. They consider IoT as a global facet and propose new models to complement business process descriptions in order to capture IoT aspects. In addition, they pay little attention to the collaboration of different professionals, which is needed to face the interdisciplinary nature of IoT-

Table 1 Consideration of design science research guidelines (Hevner et al. 2004)

Guideline	Characterization of the results presented in this paper
1. Design as an artifact	The artifact developed in this research work is a tool-supported development process to help interdisciplinary teams develop IoT-enhanced BPs. This artifact is characterized as a method
2. Problem relevance	The relevance of considering the interdisciplinary nature of IoT-enhanced BPs is justified in the motivation introduced in Sect. 1.2 and proven by the literature review presented in Sect. 2
3. Design evaluation	The tool-supported interdisciplinary development process has been evaluated with a controlled experiment with subjects, in which a complete IoT-enhanced BP has been developed. This experiment has proven the effectiveness of the approach to solving the considered problem. Consecutive research should enhance the empirical foundation of the evaluation procedure with respect to the number of cases and the diversity of the scenarios to be evaluated
4. Research contribution	This research project contributes to improving the development of IoT-enhanced BPs by considering the interdisciplinary nature of the development of these processes. In particular, we apply existing knowledge from the areas of BP Modeling, IoT Development and Ontology Management in a new and innovative way
5. Research rigor	The interdisciplinary development process has been defined and justified basing on existing solutions published in the literature. Decisions and statements have been justified by the published literature. Rigorous methods have been applied to the construction (Avison et al. 1999; Larman and Basili 2003) and the evaluation (Runeson and Höst 2009) of the designed artifact
6. Design as a search process	The project is designed as a search process because the results of the evaluation are used to enhance the tool-supported interdisciplinary development process and improve its applicability
7. Communication of research	This paper presents the proposed tool-supported interdisciplinary development process with a balance between some technology-oriented details to required understand its internal implementation and the knowledge required to effectively apply the artifact to solve the identified problem in a novel way

Table 2 Works that apply the SoC principle to face the modeling of IoT-enhanced BPs

Paper	BP modeling	IoT concepts modeling		Collaboration of different professionals	Execution Support
		Ontology	Other models	–	–
Albreshne and Pasquier (2015)	GPL4SRE	Ont4SRE		–	–
Dörndorfer and Seel (2018)	Context4BPMN	SenSoMod		–	
Gao et al. (2011)	BPMN + Functional model	Sensor ontology		–	–
Sasirekha and Swamynathan (2016)	BPEL	Extended SSN		–	Web service middleware
Serral et al. (2015)	CPN	OWL ontology		–	CPN Tool extension
Suri et al. (2017)	BPMN	IoTBPO + BPMO + IoT-Lite		–	–
Baresi et al. (2016)	BPMN		Guard-Stage-Milestone (GSM)	–	–
Tu et al. (2018)	IoTPM		domain (IoTCM), objects (IoTOM);	–	Transf. to Petri Nets
Yousfi et al. (2019)	uBPMN		Decision Model	–	–

enhanced BPs (Breiner et al. 2016). Finally, an important drawback of many of these works is that it is not clear how they support the execution of their solutions in a real environment.

Besides these approaches, and from a more general view of the IoT area, we can find several works that propose holistic approaches to face the development of IoT systems. However, they do not focus on IoT-BP as we do. For instance, Khan et al. (2017) provide a holistic view based on the Six Sigma methodology for IoT projects that covers phases from sensor data collection to the generation of business value. This work faces the challenge of providing a holistic approach from a very general and theoretical perspective without giving details about how it can be put into practice. We can also find a few approaches that support multi-paradigm, co-modeling or collaborative modeling of smart systems or cyber-physical systems (Fortino et al. 2021). These approaches typically focus on supporting the combination of multiple models of computation such as continuous-time, discrete-event, and synchronous data flow. Finally, the approach presented in Corradini et al. (2021), considers multiple actors for the modeling and development of IoT applications as executable Node-RED flows. The approach involves a Modeling Expert (ME), an expert on modeling languages and tools, an IoT Expert (IoTE), an expert on the management of IoT devices, and an IoT application developer. This approach, however, is focused on data-flow programming and therefore is limited to data manipulation activities.

3 Separations of Concerns in the Development of IoT-enhanced BP

According to the motivation presented in Sect. 1.1, this work pays special attention to facilitating an interdisciplinary development of an IoT-enhanced BP considering three main facets: (1) the definition of BP requirements; (2) the processing of low-level context data; and (3) the interoperability between IoT devices and the BP.

In this sense, the Separation of Concerns (SoC) principle is a fundamental pillar in our approach as it allows considering each of the above-introduced facets as different concerns, which can be developed through independent and decoupling activities. This facilitates the collaborative participation of different professionals in the development process. Next subsections explain the technical decisions taken to support the development of each concern and how the developed software artifacts required to support each of them are integrated to create an IoT-enhanced BP.

3.1 Technical Decisions

The development of the above-introduced concerns is driven by the following technical decisions:

BPMN to capture BP requirements. According to Weske (2012), a BP is defined as “a set of activities performed in coordination in an organizational and technical environment”. Analyzing this definition, a BP defines *what* activities must be performed, *how* these should be performed, and by *whom*. As commented above, BPs are typically defined by high-level modeling languages such as BPMN, BPEL, Petri Nets, or EPC. In this work, we propose the use of BPMN since it is a well-known and accepted standard by academia and industry. It provides an intuitive and easy way to represent the semantics of complex processes including the three aspects introduced above related to *what*, *how* and *whom*. This notation is not only used by process designers who are experts in the notation to define processes, but also by other process stakeholders such as customers, marketing professionals, or finance employees that just need to analyze them (Harmon and Wolf 2011; Leopold et al. 2015; Nysetvold and Krogstie 2005). In addition, BPMN is the most used and preferred modeling language to face the integration of BPs and IoT (Torres et al. 2020). Finally, another advantage of choosing BPMN is the myriad of commercial engines (e.g., Camunda,[1] Activti,[2] Bonita,[3] jBM,[4] Bizagi,[5] etc.) that can be used to execute these models in a real environment.

Ontology-based technology to process low-level context data and to generate high-level events. In order to process low-level context data, we followed the main ideas proposed by Taylor and Leidinger (2011) which demonstrated that ontology languages are a valuable tool to represent context data and to generate complex events. These ideas are reinforced by an exhaustive comparison of techniques to model context done by Perera et al. (2014), which demonstrates that ontologies were one of the best choices. More recent works such as Cao et al. (2019) also demonstrate that ontologies can be used to face the processing of complex events. In this work, context data is stored in OWL according to the SOSA (Sensor, Observation, Sample, and Actuator) ontology,[6] which is focused on the semantical description of IoT systems. In addition, ontologies are supported by query languages such as SPARQL that can be used by developers to analyze low-level data in order to infer the high-level events that are relevant for the process. Finally, in order to work with ontologies, there are mature open-source technologies such as the tool Protégé or the Jena engine.

Microservices as a reference architecture to face the interoperability of IoT devices. IoT devices can be supported by a myriad of technologies. In order to facilitate their interoperability with the BP at runtime, we propose a microservice architecture. Microservices (Lewis and Fowler 2014) propose an architectural style where systems are decomposed into small independent building blocks (the microservices), each of them focused on a single business capability. Microservices communicate to each other with lightweight mechanisms, and they can be deployed and evolved independently and by different development teams and technologies, which leads to more agile developments and technological independence between them (Fowler 2015). For instance, a system can be made up of both microservices implemented in Java and deployed in a Linux system, and microservices fully developed with. Net technologies. To provide the services that end-users demand microservices must be composed and must share data. To do so, lightweight mechanisms that do not introduce a high level of coupling among microservices must be used. Usually, microservices use either REST APIs or message brokers based on queues. Following this architecture, we propose a solution in which each IoT device is controlled by a microservice, which can be implemented by the technology demanded by the IoT device. However, they must publish a REST API that allows a decoupled execution of the operations supported by the IoT device. These microservices play the role of IoT Device Managers and constitute the façade that the BP must interact with in order to operate with IoT devices in a technology-independent manner.

3.2 Integration of the Developed Software Artifacts

According to the above-introduced decisions, an IoT-enhanced BP must be developed by creating the following software artifacts: (1) a BPMN model that describes the executable BPrequirements; (2) SPARQL queries that infer high-level events from the low-level context data stored in a SOSA-based context ontology; and (3) IoT Device Manager microservices that are in charge of managing the IoT devices. At runtime, the integration of these software artifacts is achieved through a microservice architecture specifically proposed to execute IoT-enhanced BPs (Valderas et al. 2022).[7] In addition to the IoT Device Manager microservices, this architecture proposes: (1) a *BP Controller* microservice that executes the BPMN model that describes the executable BP requirements; (2) a *Context Manager* microservice that is in charge of both applying the SPARQL queries to process the low-level data published by the IoT Device Managers microservices, and inferring the high-level events that are injected into de BP Controller; (3) a *Service Registry* that gives support to the service discovery, containing the network locations of microservice instances; and (4) an *Asynchronous Event Bus* that is used as communication channel between the *Context*

[1] https://camunda.com/.

[2] https://www.activiti.org/.

[3] https://es.bonitasoft.com/.

[4] https://www.jbpm.org/.

[5] https://www.bizagi.com/.

[6] https://www.w3.org/TR/vocab-ssn/.

[7] An example of an IoT-Enhanced BP implemented over this architecture can be found in https://github.com/pvalderas/iot-enhanced-business-process-example.

Manager and the IoT Device Manager microservices. Figure 2 shows the interaction required between these software artifacts to support the execution of an IoT-enhanced BP. This interaction is achieved through the following five steps:

1. The IoT Device Manager microservices periodically publish low-level context data into the asynchronous event bus. This data can be produced in any format (e.g., JSON, XML, CSV, etc.). In addition, these microservices are registered in the *Service Registry*.
2. The *Context Manager* microservice gets the low-level context data from the event bus, transforms it into a corresponding semantic description based on the SOSA concepts, and stores this description in the context ontology.
3. The *Context Manager* microservice uses a SPARQL engine to launch the SPARQL queries over the context ontology and infer high-level events.
4. The inferred high-level events are injected into a BPMN engine deployed into the *BP Controller* microservice, which is executing the BPMN model.
5. During this execution, the BPMN engine may need to interact with microservices in order to ask for the execution of an operation of an IoT device. The invocation data required to use the REST API of each microservice is obtained from the Service Registry.

To better understand this integration, let us consider again the motivating example presented in Sect. 1.1. According to the model presented in Fig. 1, two high-level events that occur in the physical world need to be considered: "Container Arrival" and "Truck Available". These events must be inferred from the low-level context data generated by IoT devices. The "Container Arrival" is the event that triggers the execution of the BP. Once the BP is running, there are some tasks that must be performed by IoT devices (e.g., move pallet to a refrigerator camera, activate an alarm). According to the five-step process presented previously we next detail what occurs at each step of the process.

Step 1. Generation of low-level context. Let us consider that there is an IoT Device Manager microservice called *Container Detector*, which detects the location of a container by using several Bluetooth beacons that detect the location of an object through their activation, and some QR readers that identify the object each time it activates a beacon. Let us also consider that they generate low-level context data, in raw format, which is defined as a JSON structure like the following:

```
{
    "activatedBeacon": "beacon25345",
    "objectID": "container28292",
    "timeStamp": "15/06/2021 19:24:10"
}
```

Step 2. Registration of low-level context in the ontology. This generated low-level context data is processed by the Context Manager microservice which registers it into the context ontology by using the concepts provided by SOSA. This is done by using the OWL language. To better understand it, Fig. 3 shows a graphical representation of this specification. White ellipses represent classes of the SOSA ontology while dark gray ellipses

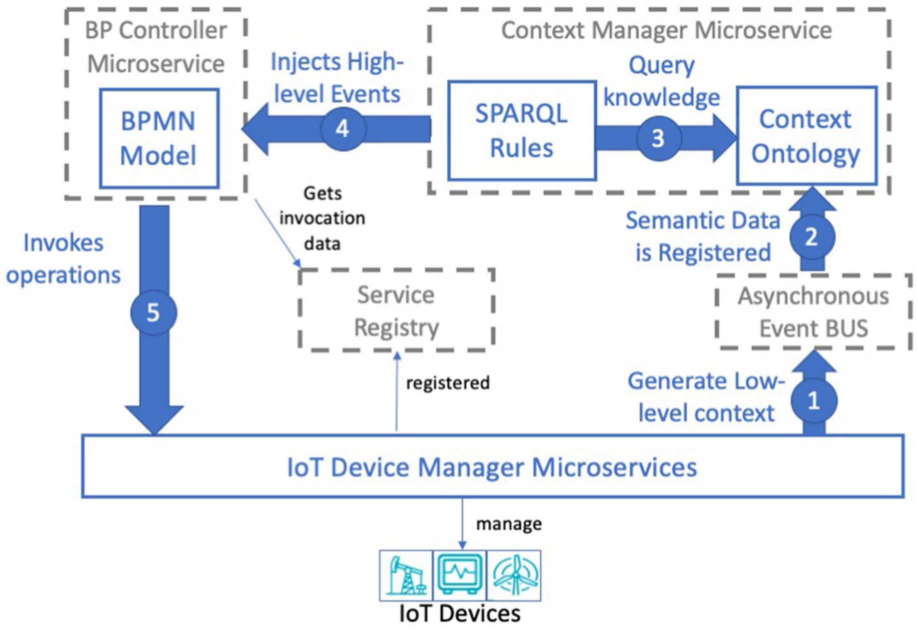

BP Controller Microservice
BPMN Model
Injects High-level Events
4
Context Manager Microservice
SPARQL Rules
Query knowledge
3
Context Ontology
Gets invocation data
Semantic Data is Registered
2
Invokes operations
5
Service Registry
Asynchronous Event BUS
registered
Generate Low-level context
1
IoT Device Manager Microservices
manage
IoT Devices

Fig. 2 Interaction among BPMN, ontology, and IoT devices

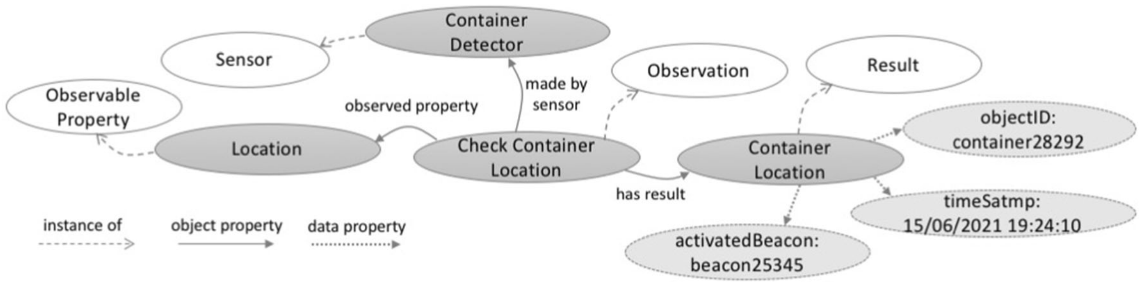


Fig. 3 Ontology-based representation of low-level context data

represent their instantiation (individuals). The "instance of" relationship that associates an individual with a class is depicted with dashed arrows. Solid arrows have a name and represent a relationship between individuals (object properties). Data properties are depicted by means of doted light gray ellipses. Note that this ontological representation includes the definition of data that are not included in the above-presented JSON data (e.g., the name of individuals such as *Check Container Location* or *Container Detector*). Section 5 will explain further how IoT developers can define this data by means of Java annotations.

Step 3 and 4. Inferring and injecting high-level events. The data registered into the context ontology must be processed in order to infer the high-level events required by the BPMN model. To do so, specific SPARQL queries are launched. For instance, let us consider the high-level event "Container Arrival". This high-level event must be triggered when a Container that has not been processed is detected in the reception area. In order to infer this event, the following SPARQL query can be executed on the context ontology. This query returns *true* or *false* depending on whether or not the defined statements are satisfied. If it returns *true* then the high-level event "Container Arrival" can be injected into the engine of the BP Controller that executes the BPMN model presented in Fig. 1.

Step 5. Execution of tasks by IoT devices. Finally, during the execution of the BPMN model that represents the IoT-enhanced BP, the corresponding engine needs to perform tasks that imply the execution of an operation of an IoT device, such as move the pallet to a refrigerator camera and activate an alarm. To do these tasks, the BP Controller gets from the Service Registry the invocation data of the corresponding IoT Device Manager microservice. Then, the BPMN engine asks the corresponding microservice for the execution of the operation.

Note that the implementation of the supporting microservices such as the BP Controller, the Context Manager and the Service Registry is out of the scope of this paper. They were presented in a previous work (Valderas et al. 2022) in which we precisely define the elements of the architecture and performed a complete evaluation that allowed us to conclude that IoT-enhanced BPs were successfully executed upon this architecture.

4 Interdisciplinary Development of IoT-enhanced BPs

The application of the SoC principle in the development of IoT-enhanced BPs allows us to propose a separation of responsibilities in order to support interdisciplinary teams.

```
PREFIX sosa: <http://www.w3.org/ns/sosa/>
ASK {
      ?obs a sosa:Observation .
      ?obs sosa:madeBySensor :ContainerDetector .
      ?obs sosa:hasResult ?result .
      ?result :activatedBeacon ?beaconId .
      ?result :objectID ?objectId .
      ?beacon :id ?beaconId .
      ?beacon :location 'receptionarea' .
      ?container a :Container .
      ?container :id ?objectId .
      ?container :processed 'false' .

}
```

Specifically, these responsibilities are: (1) the creation of IoT Device Manager microservices; (2) the definition of the BP model that is executed by the BP controller; and (3) the specification of the SPARQL rules that are executed by the Context Manager. While Sect. 4.1 explains how each of these responsibilities support one of the three facets this work focuses on, Sect. 4.2 introduces the development process based on the distribution of development responsibilities that we proposed in this work.

4.1 Supporting the Separation of Concerns of an IoT-enhanced BP

In this section, we explain how the development responsibilities required to create an IoT-enhanced BP support the three facets considered in this work. We also propose the professional profiles to be in charge of each responsibility.

4.1.1 Facet: Interoperability between IoT devices and the BP.

The interoperability between IoT devices and the BP is supported through the implementation of the microservices that are in charge of managing each IoT device. Note that these microservices publish a REST API that can be used by the BP Controller to request the execution of an IoT device operation without the need of knowing the supporting technology of the IoT device.

Development responsibility: Development of IoT-Device Manager microservices.

Professional profile: IoT developer, which is the person in charge of creating the platforms, software, hardware, and systems that allow IoT devices to function.

4.1.2 Facet: Definition of BP requirements

The business requirements that must be supported by an executable IoT-enhanced BP are captured at a high-level of abstraction using BPMN. This model is executed by the BPMN engine deployed into the BP Controller.

Development responsibility: Definition of the BPMN model.

Professional profile: Business engineer, which is the person with the knowledge and understanding of how the BP of an organization must work at macro level and are in charge of describing it in a high-level business model.

4.1.3 Facet: Processing of low-level context data

The low-level context data generated by the microservices that manage the IoT devices are published in the event Bus. The Context Manager gets this data and stores it in a context ontology implemented in OWL. Next, SPARQL queries, which play the role of Complex Event Processing rules, are launched to process the low-level context data and infer high-level events.

Development responsibility: Specification of the SPARQL queries.

Professional profile: CEP expert, which is the person with the expertise on concepts and techniques for processing real-time events and extracting the information that is relevant for the process. Specifically, in our approach these CEP techniques are supported by ontology-based technologies.

4.2 Development Process

Once the development responsibilities are identified and assigned to the corresponding professional profiles, a development process must be defined. Broadly speaking, a development process is defined from a set of sequential steps as well as the software artifacts created in each step. The steps and the software artifacts managed in each of them can be directly derived from the development responsibilities presented above. To define the sequence of steps we propose two possible approaches (see Fig. 4):

1 A top-down development approach: in this case, the interaction with IoT devices is managed according to the requirements defined by business engineers. The first step is for business engineers to define the BPMN model that satisfies the requirements of the IoT-enhanced BP. Afterwards, IoT developers must create the IoT Device Managers microservices required to support the execution of the BPMN model. Finally, CEP experts must create the SPARQL rules that generate the high-level events required by the BPMN model from the low-level context data generated by IoT devices. This type of development is useful to support situations in which a new IoT system must be set up in order to support a specific IoT-enhanced BP.

2 A bottom-up development approach: in this case, the IoT-enhanced BP is defined considering the IoT devices that exist in a system. The first step is for IoT developers to create and deploy the IoT Device Manager microservices. Afterwards, business engineers must create the BPMN model considering the operations published by each IoT Device Manager. Finally, CEP experts must create the SPARQL rules that generate the high-level events required by the BPMN model from the low-level context data generated by IoT devices. This type of development is useful to support situations in which a new IoT-enhanced BP must be defined considering only the operations

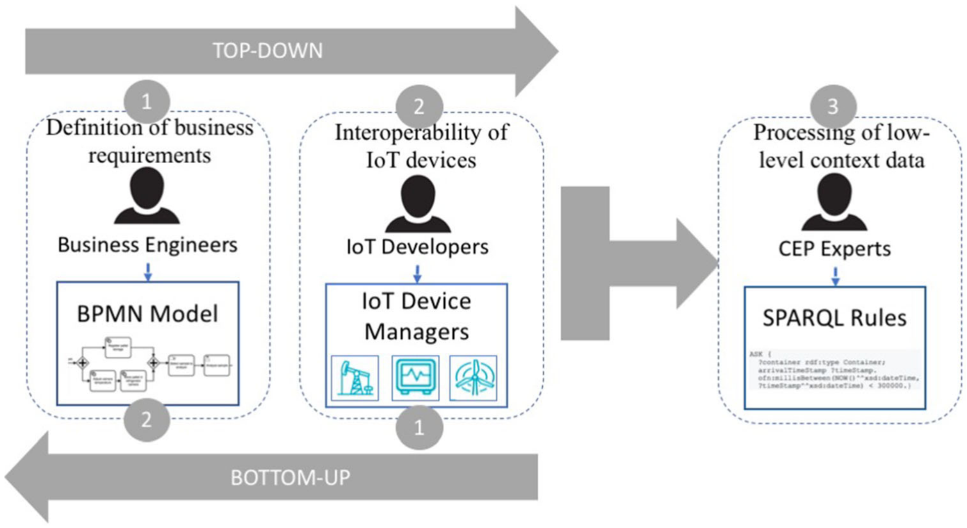


Fig. 4 Interdisciplinary Development Process for IoT-enhanced BPs

available in an IoT system whose devices are already deployed.

Note that these two approaches for developing an IoT-enhanced BP are not orthogonal. A hybrid or iterative version is possible to support an evolution of an existing IoT system driven by a BP. Consider, for instance, that business developers start to define an IoT-enhanced BP taking into account the IoT devices that are available in an existing system (bottom-up approach). However, some tasks defined in the BPMN model may require the interaction with IoT devices that are not supported by the current system. Thus, these new devices must be set up and supported by the corresponding IoT Device Manager microservice by IoT developers (top-down approach). In any case, note how CEP experts always perform their development activities after the IoT devices that are supported by the corresponding microservices fit the requirements defined in the BPMN model. This is because CEP experts need to know both: the low-level data that can be processed from IoT devices, and the high-level events that need to be injected into de BP Controller at runtime.

5 A Collaborative Development Environment. Tool Support and Demonstration

In this section, we present a collaborative development environment to support each professional (IoT developers, business engineers, and CEP experts) in the fulfillment of their development responsibilities. This environment is also used to demonstrate the feasibility of the proposed interdisciplinary development process to create an IoT-enhanced BP in a collaborative way. Figure 5 graphically shows this environment, where red squares represent the tools provided to each professional, and red arrows represent the interaction among tools and with the microservice architecture above presented. Note that:

1. To facilitate the development of IoT Device Manager microservices, IoT developers are provided with Java libraries that introduce some annotations by using the reflection and injection capabilities of the Java environment. Among other issues, this library helps IoT developers to register in the Service Registry not only the invocation data of the REST API but also a semantic description of the IoT Device and its operations.
2. To support a bottom-up approach, business engineers need to know the IoT devices that are deployed in the system, as well as their offered operations, to include them in the BPMN model. To this end, the BP Modeler obtains this information from the semantic description of each IoT device stored in the Service Registry.
3. To support a top-down approach, the BP Modeler generates Java templates that can be used by IoT developers in order to implement the IoT Device Manager microservices.
4. CEP Experts need to know the low-level data generated by each IoT Device as well as the high-level events defined in the BPMN. To support this requirement, the CEP Editor obtains the low-level context data from the semantic description of each IoT device that is available in the Service Registry. In addition, this tool connects to the BP Modeler in order to obtain the high-level events defined in the BPMN model.

Next subsections explain the tool support provided to each professional in detail.

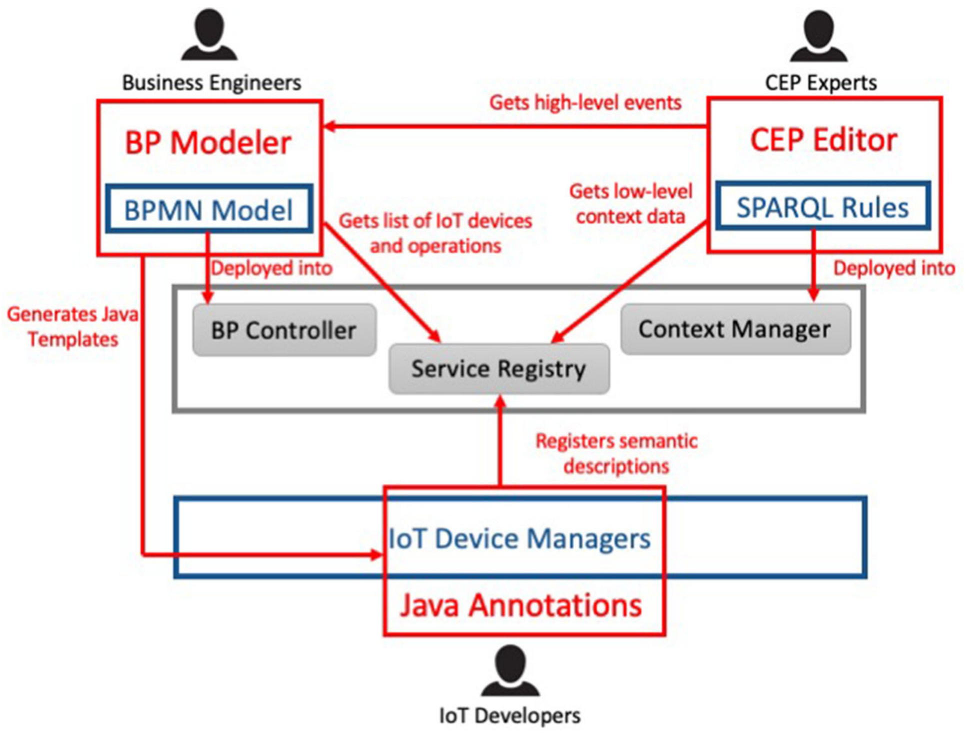


Fig. 5 Collaborative development environment for the interdisciplinary development of IoT enhanced BPs

5.1 Support for IoT Developers: Creation of IoT Device Managers

In order to facilitate the creation of IoT Device Manager microservices we provide a development support that is based on Java Annotations and both the reflection and injection capabilities of the Java environment.[8] To do so, we used some annotations provided by the framework Spring Boot to create REST APIs as a basis and have extended them with additional annotations in order to provide IoT semantics. These IoT annotations are based on concepts defined in the SOSA ontology. The main goal of introducing these semantic annotations is defining data provided by IoT devices that can be used by both the business engineers in the creation of the BPMN model when a bottom-up approach is used, and the CEP experts in the creation of SPARQL queries for processing low-level context data.

The proposed annotations are graphically shown in Fig. 6. The `IoTDeviceManager` is an annotation that can be associated to a Java class in order to create an IoT Device manager. This annotation has a readable `name` of the IoT Device that it is managed. The `RestController` annotation is used to define an API REST. The methods annotated with `RequestMapping` define the endpoints of the API through four main properties: `value`, which corresponds to the path of the REST endpoint; `method`, which corresponds to the HTTP operation; `consumes`, which corresponds to the *mime type* of data that the endpoint must receive; and `produces`, which corresponds to the *mime type* of the data that is returned by the endpoint. These methods must also be associated to the `Actuation` annotation in order to define: a readable `name` for the operation, the `resultType` of the value returned by it, and a textual `description`. The `Sensor` annotation is the counterpart of `ResquestMapping` and is used to define the sensing options of an IoT device, which are defined by methods annotated with the `Observation` annotation. Each observation is defined by a readable `name`, the `property` that is sensed, the `resultType` of the value obtained in each sensing action, and a textual `description`.

As a representative example, let us consider the *Container Detector* IoT Device Manager microservice, which is in charge of detecting the location of a container in the motivating example. This microservice plays the role of a sensor, and the data that is published by it correspond with the one introduced in Sect. 3.2. Let us consider also the *Articulated Robot* IoT Device Manager microservice that is in charge of controlling an IoT device to automatically move the pallets in the motivating example. This microservice plays the role of an actuator, and according to the BPMN model presented in Fig. 1, it should provide operations such as: move pallet to the refrigerator, move pallet to non-priority shipment area, or move pallet to

[8] The source code of the developed Java library can be found in the following Github repository: https://github.com/pvalderas/IoTDeviceManager.

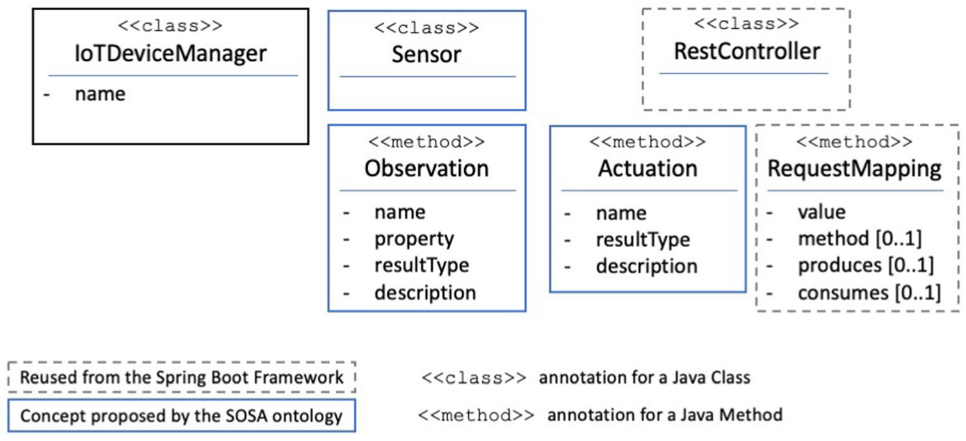
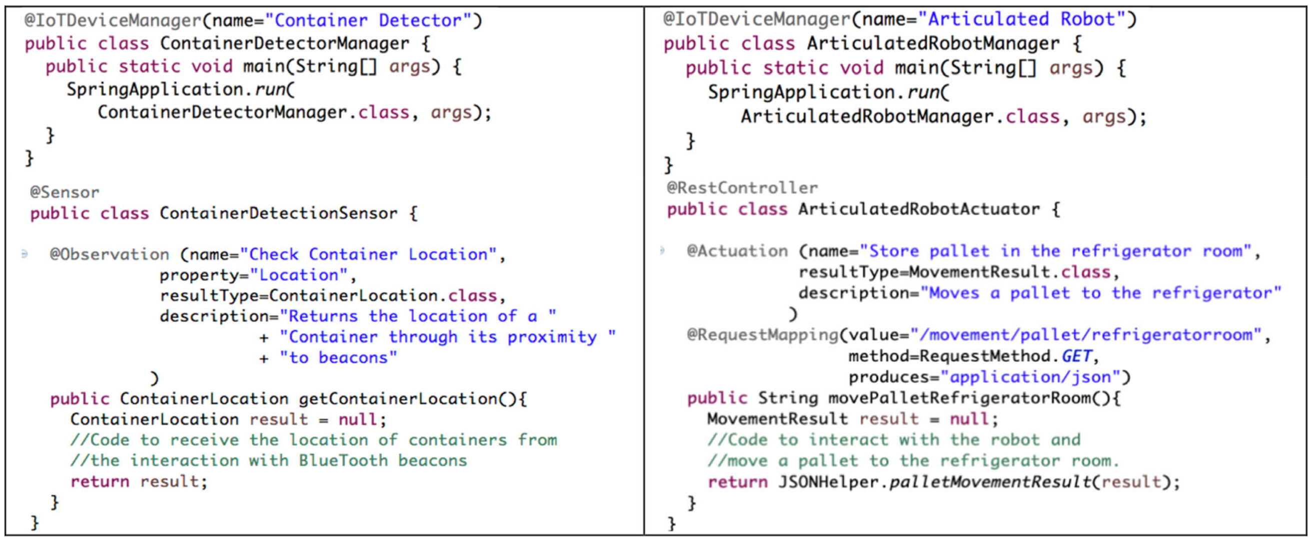


Fig. 6 Java Annotations to support the creation of IoT Device Managers



```
@IoTDeviceManager(name="Container Detector")
public class ContainerDetectorManager {
  public static void main(String[] args) {
    SpringApplication.run(
       ContainerDetectorManager.class, args);
  }
}
 @Sensor
 public class ContainerDetectionSensor {

   @Observation (name="Check Container Location",
               property="Location",
               resultType=ContainerLocation.class,
               description="Returns the location of a "
                        + "Container through its proximity "
                        + "to beacons"
              )
   public ContainerLocation getContainerLocation(){
     ContainerLocation result = null;
     //Code to receive the location of containers from
     //the interaction with BlueTooth beacons
     return result;
   }
 }
```

```
@IoTDeviceManager(name="Articulated Robot")
public class ArticulatedRobotManager {
  public static void main(String[] args) {
    SpringApplication.run(
       ArticulatedRobotManager.class, args);
  }
}
@RestController
public class ArticulatedRobotActuator {

  @Actuation (name="Store pallet in the refrigerator room",
             resultType=MovementResult.class,
             description="Moves a pallet to the refrigerator"
            )
  @RequestMapping(value="/movement/pallet/refrigeratorroom",
                 method=RequestMethod.GET,
                 produces="application/json")
  public String movePalletRefrigeratorRoom(){
    MovementResult result = null;
    //Code to interact with the robot and
    //move a pallet to the refrigerator room.
    return JSONHelper.palletMovementResult(result);
  }
}
```

Fig. 7 Example of IoT Device Managers implementation

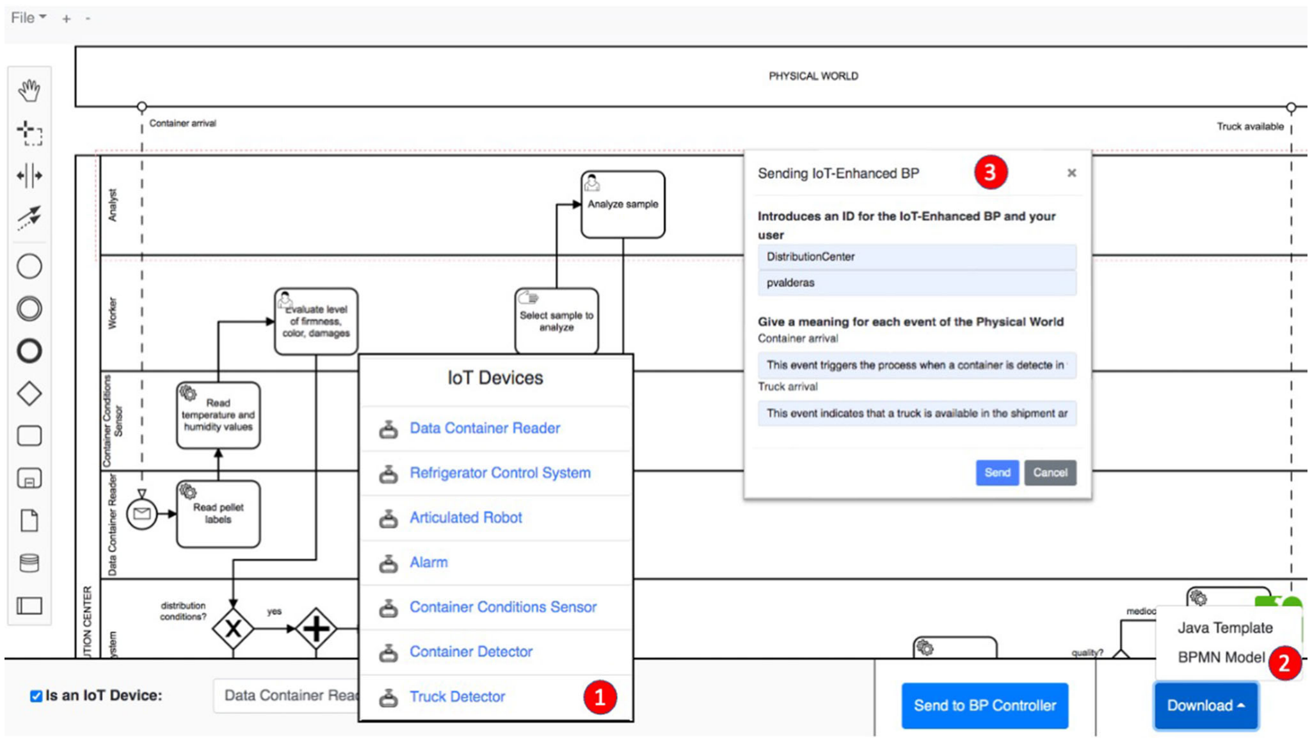
priority shipment area. Figure 7 presents a partial view of the code that IoT developers should implement in order to create the corresponding IoT Device Manager microservices.

Note that `actuation` and `observation` methods must include the code to interact with an IoT device for either the execution of an operation or the reception of sensed data. How this is done is out of the scope of this paper. IoT developers can directly use Java code, an adapter from Java to other technology, HTTP interactions, MQTT subscriptions, etc.

Finally, it is worth to remark that we have developed Java libraries that, using the reflection capabilities of the Java environment, look for the annotations presented above and perform the following actions when an IoT Device Manager microservice is executed:

- The data defined in annotations is used to register the microservice in the Service Registry including both the data associated to the REST APIs and the semantic data defined from the SOSA-based annotations. Note how the data defined in Fig. 7 for the Container Detector IoT device correspond to the graphical representation presented in Fig. 3.
- If a `Sensor` annotation is detected, the functionality for publishing context changes into the event bus is injected. Currently, the developed libraries support the interaction with the RabbitMQ[9] broker. For instance, each time the `getContainerLocation` method presented above is executed, the injected functionality intercepts the returned value and publishes it in a queue of RabbitMQ in JSON format.

[9] https://www.rabbitmq.com/.



Sending IoT-Enhanced BP

Introduces an ID for the IoT-Enhanced BP and your user

DistributionCenter

pvalderas

Give a meaning for each event of the Physical World

Container arrival

This event triggers the process when a container is detecte in

Truck arrival

This event indicates that a truck is available in the shipment ar

Fig. 8 An extended version of the modeler bpmn.io to support the modeling of IoT-enhanced BPs

5.2 Support for Business Engineers: Definition of the BPMN Model

In order to support business engineers in the creation of the BPMN model we have developed a web tool that is based on the open-source modeler bpmn.io.[10] As a representative example, Fig. 8 shows a snapshot of this tool with a partial definition of the IoT-enhanced BP of the motivation example.

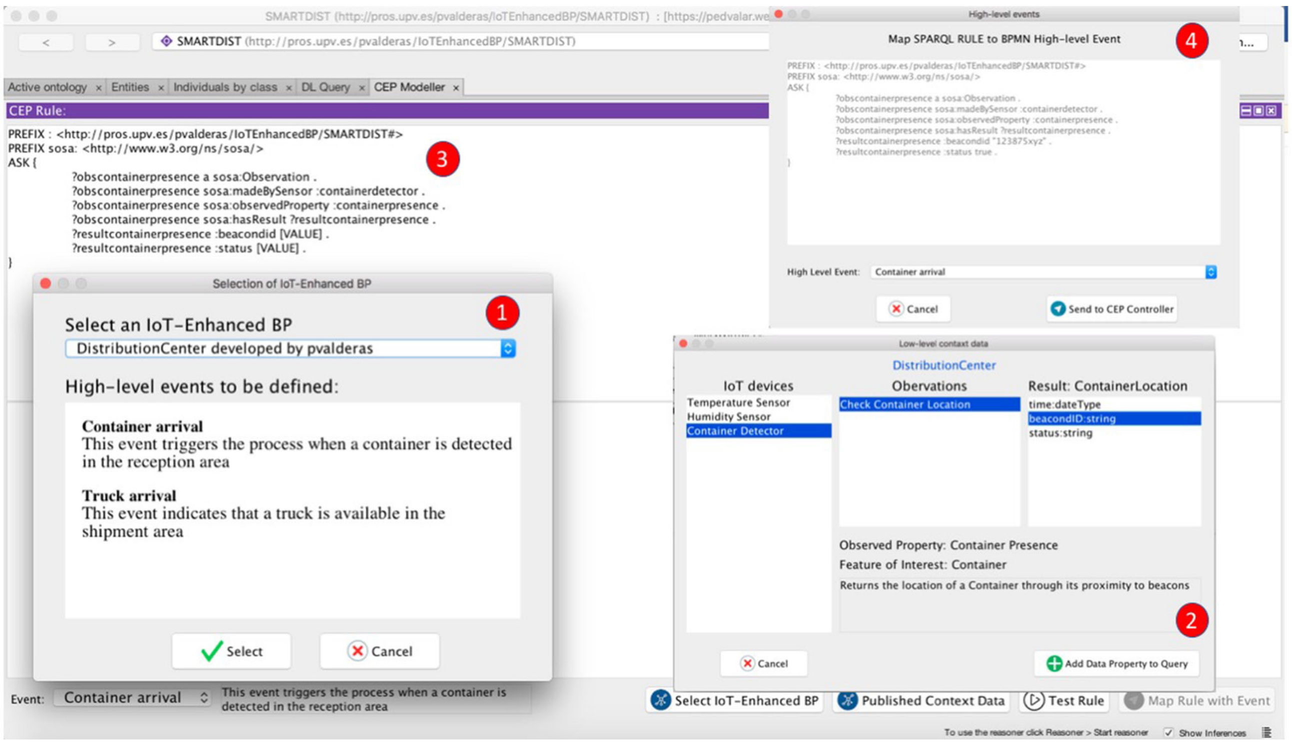
There are many solutions to model IoT-enhanced BPs with BPMN (Torres et al. 2020). To use this tool within the proposed collaborative environment, the following modeling guidelines must be followed:

(1) A pool is used to represent the whole IoT-enhance BP within an organization.
(2) Each IoT device and any other actor of an organization that participate in the process is represented by a lane within this pool.
(3) Each IoT devices' action is defined as a Service Task in the corresponding lane.
(4) The physical world is represented by a collapsed pool, which interacts with IoT devices by means of the flow sequences draw between the collapsed pool and the IoT device lanes and that represent high-level events that must be injected into the BP from the physical world.

As we can see in Fig. 8, there is a main pool that represents the "Smart Distribution Centre". It is divided in several lanes that depict the different actors that participate in the process. In the picture we can see the worker and analyst lanes that represent human actors, and two lanes that represent two IoT devices: a Data Container Reader and a Container Condition Sensor. In the complete example there are other lanes that represent the information system that performs actions on the data storage of the company, and three more lanes that represent three additional IoT devices: an Articulated Robot, a Refrigerator Control System, and an Alarm. In addition, there is also a collapsed pool named "Physical World". Note how two flow sequences are connected from this pool to a lane of the main pool in order to represent the injection of two high-level events: 'Container arrival' and 'Truck available'.

Finally, it is worth noting that this tool has been developed in order to support the two development approaches presented in Sect. 4.2. On the one hand, if a bottom-up approach is followed (i.e., IoT Device Manager microservices are created before the BPMN model) this tool is able to connect to the Service Registry in order to know the list of available microservices and their REST API. In Fig. 8 we can see how the tool provides the list of IoT devices when a lane is selected (see number 1). On the

[10] This tool is available from: http://pedvalar.webs.upv.es/iot-enhanced-bp-modeller/.


SMARTDIST (http://pros.upv.es/pvalderas/IoTEnhancedBP/SMARTDIST) : [https://pedvalar.we
SMARTDIST (http://pros.upv.es/pvalderas/IoTEnhancedBP/SMARTDIST)
Active ontology
Entities
Individuals by class
DL Query
CEP Modeller
CEP Rule:
3
Selection of IoT-Enhanced BP
1
Select
Cancel
High-level events
4
Cancel
Send to CEP Controller
Low-level contaxt data
2
Cancel
Add Data Property to Query
Select IoT-Enhanced BP
Published Context Data
Test Rule
Map Rule with Event
To use the reasoner click Reasoner > Start reasoner
Show Inferences

```
PREFIX : <http://pros.upv.es/pvalderas/IoTEnhancedBP/SMARTDIST#>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
ASK {
        ?obscontainerpresence a sosa:Observation .
        ?obscontainerpresence sosa:madeBySensor :containerdetector .
        ?obscontainerpresence sosa:observedProperty :containerpresence .
        ?obscontainerpresence sosa:hasResult ?resultcontainerpresence .
        ?resultcontainerpresence :beacondid [VALUE] .
        ?resultcontainerpresence :status [VALUE] .
}
```

Select an IoT-Enhanced BP
DistributionCenter developed by pvalderas

High-level events to be defined:

Container arrival
This event triggers the process when a container is detected in the reception area

Truck arrival
This event indicates that a truck is available in the shipment area

Event: Container arrival — This event triggers the process when a container is detected in the reception area

Map SPARQL RULE to BPMN High-level Event

```
PREFIX : <http://pros.upv.es/pvalderas/IoTEnhancedBP/SMARTDIST#>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
ASK {
        ?obscontainerpresence a sosa:Observation .
        ?obscontainerpresence sosa:madeBySensor :containerdetector .
        ?obscontainerpresence sosa:observedProperty :containerpresence .
        ?obscontainerpresence sosa:hasResult ?resultcontainerpresence .
        ?resultcontainerpresence :beacondid "123875xyz" .
        ?resultcontainerpresence :status true .
}
```

High Level Event: Container arrival

DistributionCenter

IoT devices	Obervations	Result: ContainerLocation
Temperature Sensor	Check Container Location	time:dateType
Humidity Sensor		beacondID:string
Container Detector		status:string

Observed Property: Container Presence
Feature of Interest: Container
Returns the location of a Container through its proximity to beacons

Fig. 9 Extended version of Protégé to create CEP rules based on SPARQL queries

other hand, if a top-down approach is followed (i.e., the BPMN model is defined before IoT Device Manager microservices are created) this tool is able to generate a Java template with the annotations presented above for each BPMN lane (see the two available options displayed in number 2). In any case, the tool is able to connect with the BP Controller and deploy the BPMN model once it is finished. When this is done, the business engineer is requested to introduce a textual description of the high-level events defined in the model (see number 3). This textual description is used by the editor of the CEP expert (presented in the next subsection). To do so, this web tool also publishes a REST endpoint that is used by this editor to get the high-level events defined in the BPMN model.

5.3 Support for CEP Experts: Specification of CEP Rules Based on SPARQL Queries

In order to support CEP experts in the creation of SPARQL queries we have developed a tool based on Protégé,[11] which is an open-source ontology development environment that provides tools for editing and executing SPARQL queries. According to the development process proposed in Sect. 4.2, these queries are done after (1) IoT Device Manager microservices are registered into the Service Registry, and (2) the BPMN model is defined. Thus, we have extended Protégé in order to: (1) connect to the Service Registry to get the semantic descriptions defined by IoT developers through the Java annotations presented in Sect. 5.1; and (2) connect to the REST endpoint published by the BPMN modeler presented in the previous subsection in order to get the high-level events defined by business engineers.

A snapshot of the developed CEP editor[12] is shown in Fig. 9. As we can see, this tool provides CEP experts with a user interface that facilitates the creation of SPARQL queries that process low-level context data. First (see number 1) the tool provides a window to select an IoT-enhanced BP from the list of available processes defined in the BPMN modeler. For each BP, a definition of the high-level events associated to it are displayed at the bottom of this window. This information is obtained from the BP modeler. Next (see number 2) the tool provides the developer with a new window that shows the IoT devices that are available for the selected IoT-enhanced BP as well as the context data they publish. In this case, this information is obtained from the data stored in the Service Registry by the IoT Device Manager microservices. Next (see number 3), a SPARQL query template is automatically generated from the low-level context data previously selected. At this point, the CEP expert should complete the

[11] https://protege.stanford.edu/.

[12] The source code of the current version of this tool can be found in https://github.com/pvalderas/iot-enhancedBP-protege-plugin.

template with the missing values and test it before it can be executed. Finally (see number 4), the completed SPARQL query can be associated to a high-level event and deployed into the Context Manager.

5.4 Supporting the Development Process

In order to better understand how the developed tools support the interdisciplinary development process presented in Sect. 4 let us summarize how the two presented approaches (bottom-up and top-down) are supported.

If a bottom-up approach is followed, IoT developers create the IoT Device Manager microservices by using the Java annotations presented in Sect. 5.1. When the microservices are executed, they automatically register both the data of their API REST and the semantic information in the Service Registry. Then, business engineers use the BP modeler presented in Sect. 5.2 to define the BPMN model. This tool connects to the Service Registry to provide the engineers with the list of available IoT Device manager microservices and their operations. Once the model is defined it is deployed into the BP Controller. Finally, CEP experts use the editor presented in Sect. 5.3 to create the necessary SPARQL queries. This tool connects to the Service Registry and the BP modeler in order to provide CEP experts with both the low-level data published by the IoT Device Manager microservices and the high-level events defined in the BPMN model. Once the SPARQL queries are defined, they are deployed into the CEP Manager.

If a bottom-up approach is followed, first business engineers use the BP modeler presented in Sect. 5.2 to define the BPMN model. In this case, there are no available IoT Device manager microservices. Once the model is defined it is deployed into the BP Controller. The modeler automatically generates templates with the proposed Java annotations in order to implement the IoT Device Manager microservices required by the BPMN model. Then, IoT developers use the Java templates generated by the BP modeler to develop the IoT Device manager microservices and register them in the Service Registry. Finally, CEP experts use the editor presented in Sect. 5.3 to create the SPARQL queries in the same way as in the bottom-up approach.

6 Evaluation

In this section, we present a controlled experiment with subjects to evaluate the proposed interdisciplinary development approach for IoT-enhanced BPs. In particular, the two hypotheses that we wanted to validate were the following:

H1: The proposed development approach allows each developer to perform her/his development activities with independency from the development responsibilities of other professionals.
H2: The development environment allows effective collaboration among professionals in order to develop an IoT-enhanced BP.

In order to perform this experiment, we followed the research methodology practices provided by Runeson and Höst (2009). Next, we introduce the experiment by describing its participants, design, execution, analysis of the results, and threats of validity.

6.1 Participants

A total of 24 subjects between 21 and 45 years old participated in the experiment (10 female and 14 male). They were all students of different degrees at our university and were recruited through personal invitation. The number of participants recruited was designed to facilitate their distribution in three balanced groups of eight participants each. Each group played the role of a specific developer (i.e., business engineers, CEP experts and IoT developers). In addition, although the experiment was done with students, we tried to find participants that had similar backgrounds and skills to the ones expected in the corresponding professional developers. In particular, we needed participants with programming expertise in Java in order to create the microservices and play the role of IoT developers. Other participants had to know how to define BPMN models that represent IoT-enhanced d business processes in order to play the role of business engineer. And other participants needed to have worked with ontologies and SPARQL in order to play the role of CEP expert. We formed the following three groups of students:

- Group 1, which was composed of eight students of the Computer Science Bachelor's Degree. These participants played the role of IoT developers. They had experience in Java programming and had previously worked on developing REST APIs and asynchronous communications with message brokers. However, none of them had worked with microservices.
- Group 2, which was composed of eight students of the master's degree in Computer Science. They played the role of business engineers. They had previously worked with BPMN models in several subjects of the degree. However, they had never work on the modeling of a BP that needed a real integration with IoT devices.
- Group 3, which was composed of eight participants of the master's degree in Information Management. These participants played the role of CEPs experts. They had previously worked with ontologies in several subjects

of the master's degree. However, none of them had worked previously with SPARQL.

Each group was made up of students of the same course which, in principle, guarantees that participants of the same group had all a similar profile. However, in order to be sure and detect possible shortcomings, we propose they fill in a questionnaire with some questions related to their experience and background. In addition, as we explain further, the training sessions done during the experiment were used to teach participants the technology required to apply our approach which they had not experienced. These sessions were also used to reinforce some basic notions we consider opportune from the analysis of the questionnaire results.

6.2 Design

We arranged an experiment in which each group of participants was in charge of developing a facet of the IoT-enhanced BP presented as a motivating example. Participants had to collaborate by using the proposed collaborative development environment by following a bottom-up approach (see Sect. 4.2). The evaluation of the top-down approach is analogous and was left as further work. We were in charge of setting up the collaborative development environment in such a way it was accessible through the Internet. The instruments that were used to carry out the experiment were:

- A demographic questionnaire: it was used to know the level of the users' experience in process modeling, BPMN, IoT, ontologies, and microservices.
- Work description: the description of the work that the subjects should carry out in the experiment. Each group was in charge of performing a different task:
  - Group 1, whose participants play the role of IoT developers, was in charge of implementing the IoT Device Manager microservices that were needed to implement the motivating example.
  - Group 2, whose participants play the role of business engineers, was in charge of creating the BPMN model that described the BP requirements of the motivating example.
  - Group 3, whose participants play the role of CEP experts, was in charge of defining the SPARQL rules that infer the high-level events defined in the BPMN model from the low-level context data produced by the microservices.
- A NASA-TLX questionnaire (Hart and Staveland 1988): it was used to evaluate the perceived mental/physical/temporal demand, performance, effort, and frustration on a 100-point scale with 5-point steps. This questionnaire was extended with additional open questions to allow participants to introduce additional comments about the performed activities.

6.3 Execution

To perform the experiment, we arranged three independent workshops of one day. Each of the three workshops was divided into two sessions: (1) a training session of three hours in which they filled in the demographic questionnaire and we explained the participants the knowledge required to perform the tasks proposed in the experiment; and (2) a working session of four hours in which participants had to perform these tasks. Each workshop was focused on the participation of one group in the following order:

a. The first workshop that we arranged was focused on the tasks of Group 1. In particular, the participants were invited to implement the IoT Device Manager microservices required to support the motivating example. In the training session, we explained them how the interdisciplinary development of IoT-enhanced BPs was supported by the proposed collaborative environment. After explaining the development role, they were asked to play, we trained them in the use of the Java annotations we developed to create IoT Device Manager microservices. From the questionnaire results, we detected some doubts related to the use of jar libraries in Java so we introduced also a little explanation about this point. In the working session, we presented them the motivating example and provided them with a textual description of the IoT devices that needed to be available. We also introduced the REST API that each of them must implement and the context data that microservices must publish into the Event Bus, if any. Specifically, participants had to develop seven IoT Device Manager microservices: Data Container Reader, Container Condition Sensor, Articulated Robot, Refrigerator Control System, Alarm, Truck Detector and Container Detector. Note that we wanted to evaluate the use of the Java library proposed to create IoT Device Manager microservices. The code required to implement the interaction with the physical device is out of the scope of this experiment. Thus, we provided participants with additional Java adapters to emulate this interaction. In order to do the proposed tasks, participants could use the Java development environment they preferred. Once the IoT Device Manager microservices were developed they were executed and registered into the Service Registry of the microservice architecture that supports the execution of IoT-enhanced BPs.

b. The second workshop was arranged to perform the task of Group 2. In this case, the participants needed to create the BPMN model that describes the BP requirements of the motivating example. In the training session, we explain them how the interdisciplinary development of IoT-enhanced BPs was supported by the proposed collaborative environment. After explaining the development role, they were asked to play, we trained them in the use of the web modeler developed to defined BPMN models that support IoT-enhanced BPs. In the working session, we presented them the motivating example and provided them with a textual description of the BP that they needed to support (similar to the one presented in Sect. 1.1). Participants had to use the web modeler we developed, which provided them with the list of available IoT Device Manager microservices developed in the first workshop and their operations. Once participants finished the BPMN model, they used the web modeler to deploy the model into the BP Controller of the microservice architecture that supports the execution of IoT-enhanced BPs.
c. The third and last workshop was focused on the development of the task of Group 3. The participants were invited to define the SPARQL queries required to generate the high-level events that were included in the BPMN model. In the training session, we explained SPARQL and how the interdisciplinary development of IoT-enhanced BPs was supported by the proposed collaborative environment. Since we saw from the questionnaire results that most subjects did not remember everything about Protégé, after explaining the development role they were asked to play, we trained them in the use of both Protégé and the extension we implemented. In the working session, we presented them the motivating example. In this case, we did not provide them with additional material since all the information required to perform their task was provided by the developed tool. The low-level context data that were available to be processed were obtained from the information stored in the Service Registry by the IoT Device Manager microservices developed in the first workshop. The high-level events that had to be inferred were provided by the web modeler from the BPMN model created in the second workshop. Once participants finished the SPARQL queries they used the extended version of the Protégé tool to deploy them into the CEP Controller of the microservice architecture we had set up.

After the training session of each workshop, we performed a break of one hour before starting the working session. When participants finished their task, they had to fill in the NASA-TLX questionnaire. Throughout the working session of each workshop, we observed participants and took notes on their behavior. Finally, once the IoT-enhanced BP was completely developed and deployed we tested it through its execution in the proposed microservice architecture.

6.4 Analysis of the Results

Next, we present and analyze the results obtained from the above-introduced experiment regarding the tasks performed by each group. Table 3 shows the values obtained for each group from the NASA-TLX questionnaires (average (Avg), median (Med), standard deviation (SD), best result (Best), and worst result (Worst) columns). As introduced above, the NASA-TLX questionnaire evaluates the perceived mental/physical/temporal demand, performance, effort, and frustration on a 100-point scale, where the highest scores represent the worst results. Thus, mental / physical / temporal demand, effort and frustration are rated between very low (value 0) and very high (value 100); while the performance is rated between very good (value 0) and very bad (value 100).

Table 3 NASA-TLX results

Factors	Task of Group 1 (IoT Developers)					Task of Group 2 (Business Engineers)					Task of Group 3 (CEP Experts)				
	Avg	Med	SD	Best	Worst	Avg	Med	SD	Best	Worst	Avg	Med	SD	Best	Worst
Mental Load	20,0	20,0	5,3	15,0	30,0	33,1	30,0	15,8	15,0	65,0	40,0	35,0	12,0	30,0	65,0
Physical Dem	3,1	5,0	2,6	0,0	5,0	3,1	2,5	3,7	0,0	10,0	4,4	5,0	4,2	0,0	10,0
Temp. Dem	38,8	37,5	13,3	20,0	60,0	29,4	27,5	19,2	10,0	60,0	29,4	27,5	11,2	20,0	55,0
Performance	23,8	20,0	11,9	15,0	50,0	23,8	20,0	9,9	15,0	45,0	28,8	25,0	9,2	20,0	45,0
Effort	26,3	22,5	11,3	15,0	50,0	25,0	22,5	13,6	10,0	55,0	33,8	32,5	11,9	20,0	55,0
Frustration	21,9	20,0	7,0	15,0	35,0	14,4	15,0	6,2	5,0	25,0	20,6	20,0	6,2	15,0	30,0
Global load	30,25					30,71					34,54				

In order to compare tasks, the NASA-TLX proposes to calculate a pondered global workload for each task (Hart and Staveland 1988). To facilitate the interpretation of this global score, Grier (2015) presents a descriptive analysis of over 1000 global NASA-TLX scores from over 200 publications. This analysis obtained an average global score of 48.74. The minimum and maximum scores were 8 and 80 respectively. As we can see in Table 3, the global workload obtained for each task is lower than the average obtained in this analysis, which let us consider the obtained results as good. Next, we comment on the results obtained for each group in more detail.

Group 1. Implementation of IoT Device Manager microservices. All the participants of Group 1 were able to perform the proposed task. They properly used the developed Java libraries in order to create the microservices that were required by the motivating example. In addition, the results obtained in this NASA-TLX questionnaire (see Task of Group 1 columns in Table 1) are quite encouraging.

From a general point of view, the task of developing IoT Device Manager microservices was ranked with acceptable values in the analyzed factors. Little mental load and effort was required by participants, which allows us to conclude that the proposed Java annotations are intuitive and easy to use for developers with experience in the Java programming language. The little frustration and good performance that was indicated by participants also reinforce this consideration. Regarding the temporal demand, we can see it was rated higher than other factors. Analyzing the open question included in the questionnaire we can conclude that this was related to the amount of microservices that participants had to implement. Although we think the implementation of a microservice per IoT device fits the philosophy of this architecture and provides a proper level of decoupling and independence among BPs and IoT devices, we understand that participants found the fact of implementing seven microservices a time-consuming task.

Also, some participants suggested the integration of the annotation `RequestMapping` into the `Actuation` annotation in order to have only one annotation associated to each method (as happens with the `Observation` annotation, see Fig. 7). We use these two separated annotations because the `RequestMapping` is provided by the Spring Framework, which is in charge of injecting the functionality to support a REST endpoint. Even though we agree with the suggestion made by some participants, the annotation framework provided by Java does not support inheritance of annotations and we cannot create an annotation that inherits the injection capabilities of another. Thus, further research is needed to improve the proposed solution in this direction.

Group 2. Definition of BP requirements. All participants were able to create the BPMN model for the IoT-enhanced BP successfully, although some of them need our help to properly understand the motivating example and complete the model. According to the NASA-TLX results (see Task of Group 2 columns in Table 1), this task was ranked with values that illustrate that participants feel comfortable enough when using the developed web modeler to define an IoT-enhanced BP model. The modeling guidelines that must be followed to create the BPMN model were also easily adopted by participants. Thus, we can consider the modeling approach supported by the web modeler is easy enough for business engineers with experience in the use of BPMN, which reinforce a previous experiment done in (Valderas et al. 2022). However, further research is needed to analyze how business engineers without experience in BPMN feel when using the developed web modeler.

The most remarkable factor in this experiment was the mental load, which is rated with the highest value. Analyzing the comments done by participants we concluded that the main reason of this rank was related to the initial conceptualization effort that they had to do in order to identify the actions done by each device as well as the events that must be injected from the physical world.

Other suggestion done by some of the participants was the possibility of highlighting the lanes that were associated to an IoT devices in a different color. We found this option really interesting and we are currently working on it. Finally, another significant comment done by participants was related to the input and output of each task. Currently, the web modeler allows business engineers to define the sequence of tasks considering the operations provided by the available IoT devices. However, it does not support the management of the input and output of the operations associated to each task. This is an aspect that we plan to face as further work.

Group 3. Processing of low-level context data to generate high-level events. According to the NASA-TLX results (see Task of Group 3 columns in Table 1), the task was rated with acceptable values, although this was the task that most mental load and effort demanded to participants. This is an expected result if we consider that, contrarily to groups 1 and 2, participants in Group 3 had never used the language that was required to complete the assigned task, i.e., SPARQL. Despite this, all participants were able to create the SPARQL queries that infer the high-level events defined in the BPMN. However, although the extended version of Protégé provided participants with detailed data of the low-level context data that can be processed, some participants needed additional explanations in order to understand it. Currently, this data is based on notions such as Device, Observation, Property, Result and so on of the SOSA ontology. We noticed that complementing this characterization of the low-level context data with examples of it (i.e., examples of generated values) could improve its understanding.

Besides, another interesting suggestion done by participants was the possibility of using the tool to contact with the other developers to ask for clarifications related to the data they have to work with. We think this is a really useful characteristic to improve the collaborative development environment of IoT-enhanced BPs.

6.5 Further Analysis

The previous subsection has introduced the main results obtained for each development activity in an individual and independent way. In this subsection, we introduce additional results obtained through an analysis from a global perspective.

Collaborative development environment. We evaluated the effectiveness of our development environment in order to facilitate the collaboration among the participants of each group. In particular, the main goal to be evaluated was whether the data that need to be interchanged among developers was properly provided to them through the development tools.

During the experiment, participants only focused on using the tool required to perform their corresponding task. They were not aware of who were the other developers they were collaborating with. However, we internally created development teams made up of three participants, one of each group (eight teams in total). In this way, the collaborative environment should control that the software artifacts created by the members of a team must be shared only by these members. For instance, the high-level events defined by the business engineer of a team must appear in the extended version of Protégé used by the CEP expert of this team, and only to this CEP expert. In general, the environment worked successfully and only some minor coding mistakes that affect the communication among tools were detected and fixed accordingly.

Quality of the resultant IoT-enhanced BPs. Once the subjects finish the development of the IoT-enhanced BPs (after the three workshops), we executed the resultant eight versions in the microservice architecture we had set up. In order to evaluate the execution of these examples, we analyzed the logs generated by the IoT Device Manager microservices to check whether the IoT-enhanced BP was successfully executed. The eight developed versions were executable, but four of them presented minor errors that were not detected during the workshop activities and needed to be corrected. These errors were mainly produced by either mistakes in the sequence of tasks defined in the BPMN model or a wrong condition defined in the SPARQL queries required to infer high-level events. These errors were produced by a mismatch between the requirements of the example and the BPMN model or the SPARQL rules. Although the interdisciplinary development approach and the supporting collaborative environment did not give any issue, the detection of these errors was not easy. This made us consider the addition of debugging support in our tool as further work.

Apart from a correct execution of the created solutions, we rated them to also evaluate their quality. The difficulty in rating these artifacts is comparable to grading exams. To facilitate this evaluation, a master solution was used as a reference point. The artifacts created by each group (i.e., microservice implementation, BPMN models, and SPARQL rules) were independently evaluated by two of us in order to reduce subjectivity. The two evaluators discussed the corrections and agreed upon a unique mark for each artifact. Finally, we calculated the average of the marks obtained for the three artifacts to obtain a unique mark for each complete solution. We obtained eight grades between 74 and 100%, obtaining an average mark of 84.8%, which can be considered a good result. Next, we provide some details about the evaluation done on each artifact separately.

The most significant issues detected in the Java implementation of microservices were related to the description given to some actuations or observations through the corresponding annotation property. Some of them were not properly defined or even defined in blank. This did not avoid the creation of the whole solution since business engineers had the name of the observation or actuation to be included in the BPMN model. In the same way, this aspect neither affected the execution of the created solution. However, the quality of the solutions with these issues was reduced. In this case, we obtained eight grades between 79 and 100%, obtaining an average mark of 90.1%. Regarding the BPMN models and the SPARQL rules, the most important problems that we detected were related to the misunderstanding of the requirements commented above. In the case of BPMN models, some of them did not define the flow of the process correctly (e.g., some parallel executions were defined as sequences, some condition gateways were missed), or some IoT Device operations or events associated with the BPMN elements were not the correct ones (e.g., the event ‘Truck available’ were not always included). BPMN models were graded between 76 and 100%, obtaining an average mark of 88.7%. Regarding the SPARQL rules, the most important detected problems were the definition of conditions that were not really needed to define a physical world event or the use of a wrong property (e.g., some mistakes were detected in the comparison of the properties beaconID and status with the values ‘container_%’ and ‘notProcessed’ required to define the 'Container arrival' event). These two problems may be related to the lack of experience of participants in using this query language. SPARQL rules were graded between 66 and 100%, obtaining an average mark of 75,6%.

6.6 Conclusions

The results obtained in this experiment allow us to accept Hypothesis 1 and conclude that the proposed development approach allows each developer to perform their development activities with independence of the development responsibilities of other professionals. As we have exposed in the above-introduced explanation, the participants of each group were able to perform the requested tasks without the need of participating in the development of the other software artifacts. For instance, CEP experts could implement the required SPARQL queries without the need to participate in the development of IoT Device Manager microservices or the creation of the BPMN model. In the same way, business engineers and IoT developers could perform theirs. In this sense, the application of the SoC principle in the proposed development approach has been proven as a good solution to face the interdisciplinary nature of IoT-enhanced BP.

Regarding Hypothesis 2, which focuses on the effectiveness of the development environment to support the collaboration among professionals, we are also satisfied. Although the proposed tools have space for improvement as mentioned previously in Sect. 6.4, the performed experiment has demonstrated that the collaborative environment works properly. The developed tools have achieved the interchange of the data that was required to integrate the software artifacts that were independently developed by different developers.

Finally, we think that the proposed solution can facilitate further maintenance and evolution of an IoT-enhanced BP. The decoupling of the supporting software artifacts as well as the independence provided to developers can contribute to this issue. However, maintenance and evolution are challenges that require a more precise evaluation.

6.7 Threats to Validity

The various threats that could affect the results of this experiment and the measures that we took were the following:

Validity of conclusions. This validity concerns the relationship between the treatment and the outcome. Our experiment was threatened by the random heterogeneity of subjects. This threat appears when some users within a user group have more experience than others. The participants of a group had all a similar profile since they are all students of the same course in the same academic year. This helps to minimize the heterogeneity of subjects. In addition, this threat was also minimized with: (1) the demographic questionnaire that allowed us to evaluate the knowledge and experience of each participant beforehand and detect possible shortcomings; and (2) the training sessions in which all subjects participated in order to have a similar background in the technologies required to perform the proposed tasks. In these training sessions, we taught participants the technology required to apply our approach and we also reinforced some basic notions we consider opportune from the analysis of the questionnaire results.

Internal validity. Our experiment was threatened by the hypothesis guessing threat: when people might try to figure out what the purpose and intended result of the experiment are and are likely to base their behavior on their guesses. We minimized this threat by hiding the goal of the experiment (i.e., the hypotheses to be validated were not shared with the participants). Note also that we introduced some subjectivity when grading the created solutions by comparing them with a master one. To reduce this problem each solution was evaluated twice. In addition, some participants asked for some clarifications during the experiment regarding the business requirements that the BPMN model should meet. They also asked some technical questions about the SPARQL language, which they had never used before. We answered all these questions by clarifying notions that were already introduced either in the presentation of the case study or in the training sessions. We were very careful to not introduce any help about the solution they needed to develop. Despite this, this may be considered a threat to this experiment.

External validity. This type of validity concern is related to conditions that may limit our ability to generalize the results of the experiment to industrial practice. This treat is reduced by making the experimental environment more realistic. Thus, we provided participants with an experimental setting that representative for industrial practice. Note that participants of Group 1 could use the Java environment they preferred; participants of Group 2 used an extended version of bpmn.io, one of the most used open-source BPMN modelers; and participants of Group 3 used an open-source tool that is s supported by a strong community of academic, government, and corporate users. In addition, participants did not face the development of a toy example, but they were proposed to support an example based on a real scenario (Bowman et al. 2009).

7 Conclusions and Further Work

In this work, we have presented an approach that applies the SoC principle to support the interdisciplinary development of IoT-enhanced BPs. In particular, we have considered three main facets: BP requirements, low-level data processing, and the interoperability between IoT devices and the BP. In order to support the development of each concern we have proposed the use of BPMN, ontology-based technologies, and a microservices architecture.

The microservice architecture facilitates the decoupled execution of the software artifacts that define the considered concerns of an IoT-enhanced BP. This architecture also facilitates the development of each artifact with a high degree of independence among them, which contributes to support the interdisciplinary development process that we have presented. This process proposes two main approaches in order to develop the three considered concerns: a top-down and a bottom-up approach. In the experiment done, we have validated the latter. The validation of the former is analogous and is left to further work.

The interdisciplinary development process is supported by a collaborative development environment that introduces tools that allow each professional to perform their development responsibilities in an independent way, without the need to be directly involved in the development of other artifacts. However, the environment is in charge of maintaining the tools integrated in order to allow the interchange of data required to develop each concern of an IoT-enhanced BP.

The contributions of our work are both theoretically and practical. From a theoretical point of view, we have proposed a solution based on the SoC principle in order to face the interdisciplinary nature of IoT-enhanced BPs. From a practical point of view, we have provided a collaborative development environment that supports the proposed development process. We have also presented some insights through a case-study validation, which reveals 1) that the proposed development approach allows each developer to perform their development activities with independence of the development responsibilities of other professionals, and 2) the development environment allows an effective collaboration among professionals.

As future work, we plan to continue with the different improvements identified through the validation experiment. On the one hand, the different developed tools can be enriched with the new capabilities proposed by the participants of the experiment. On the other hand, the development environment needs to be extended with tools that facilitate the debugging of the artifacts developed by different professionals. Also, as commented above, the top-down approach of the proposed development process needs to be specifically validated.

In addition, we want to enrich our solution with the consideration of additional concerns. For instance, a significant concern to be considered in the development of an IoT-enhanced BP is the risk management and mitigation issue (Conforti et al. 2011), in such a way that developers can identify risks in executing IoT-enhanced BPs and simulate them at design time. We are also investigating how goal-oriented models can be integrated with IoT-enhanced BPMN models in order to select from a list of semantically annotated microservices the best ones (e.g., those with a less resource consumption) to achieve the specified goals.

Acknowledgements This work is part of the R&D&I project PID2020-114480RB-I00 funded by MCIN/AEI/1013039/501100011033

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P (1999) Towards a better understanding of context and context-awareness. In: International symposium on handheld and ubiquitous computing. Springer, Heidelberg, pp 304–307

Albreshne A, Pasquier J (2015) A domain specific language for high-level process control programming in smart buildings. Procedia Comput Sci 63:65–73. https://doi.org/10.1016/j.procs.2015.08.313

Avison DE, Lau F, Myers MD, Nielsen PA (1999) Action research. Commun ACM 42(1):94–97. https://doi.org/10.1145/291469.291479

Baresi L, Meroni G, Plebani P (2016) A GSM-based approach for monitoring cross-organization business processes using smart objects. Lecture notes in business information processing 256. Springer, Heidelberg, pp 389–400

Bermudez-Edo M, Elsaleh T, Barnaghi P, Taylor K (2017) IoT-Lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. Pers Ubiquitous Comput 21(3):475–487. https://doi.org/10.1007/s00779-017-1010-8

Beverungen D, Buijs JCAM, Becker J, di Ciccio C, van der Aalst WMP, Bartelheimer C, vom Brocke J, Comuzzi M, Kraume K, Leopold H, Matzner M, Mendling J, Ogonek N, Post T, Resinas M, Revoredo K, del Río-Ortega A, la Rosa M, Santoro FM, Wolf V (2021) Seven paradoxes of business process management in a hyper-connected world. Bus Inf Syst Eng 63(2):145–156. https://doi.org/10.1007/S12599-020-00646-Z

Bowman P, Ng J, Harrison M, Sánchez Lopéz T, Illic A (2009) Sensor based condition monitoring. Building radio frequency identification for the global environment. http://bridge-project.eu/data/File/BRIDGE_WP03_sensor_based_condition_monitoring.pdf. Accessed 18 Jul 2022

BPMN (2010) Business process model and notation concepts. https://www.omg.org/spec/BPMN/2.0.2/PDF. Accessed 18 Jul 2022

Breiner S, Subrahmanian E, Sriram RD (2016) Modeling the internet of things: a foundational approach. ACM Int Conf Proc Ser Part F127184:38–41. https://doi.org/10.1145/3017995.3018003

Cao Q, Giustozzi F, Zanni-Merk C, de Bertrand F, Reich C (2019) Smart condition monitoring for industry 4.0 manufacturing processes: an ontology-based approach. Cybern Syst 50(2):82–96. https://doi.org/10.1080/01969722.2019.1565118

Chapline G, Sullivan S (2010) Systems engineering for life cycle of complex systems. Engineering Inovations (NASA), pp 302–318. https://er.jsc.nasa.gov/seh/536823main_Wings-ch4.pdf. Accessed 18 Jul 2022

Choo KKR, Gai K, Chiaraviglio L, Yang Q (2021) A multidisciplinary approach to internet of things (IoT) cybersecurity and risk management. Comput Secur. https://doi.org/10.1016/j.cose.2020.102136

Conforti R, Fortino G, la Rosa M, ter Hofstede AHM (2011) History-aware, real-time risk detection in business processes. In: OTM confederated international conferences LNCS (part 1). Springer, Heidelberg, pp 100–118

Corradini F, Fedeli A, Fornari F, Polini A, Re B (2021) FloWare: an approach for IoT support and application development. enterprise business-process and information systems modeling. Springer, Heidelberg, pp 350–365

Dey AK (2001) Understanding and using context. Pers Ubiquitous Comput 5(1):4–7. https://doi.org/10.1007/S007790170019

Dimitrov M, Simov A, Stein S, Konstantinov M (2007) A BPMO based semantic business process modelling environment. In: CEUR workshop proceedings, p 1613–0073. http://ceur-ws.org/Vol-251/paper13.pdf. Accessed 18 Jul 2022

Dörndorfer J, Seel C (2018) A framework to model and implement mobile context-aware business applications. Lecture notes in informatics, Springer, Heidelberg, pp 23–38. https://dl.gi.de/handle/20.500.12116/14956. Accessed 18 Jul 2022

Etzion O, Niblett P (2010) Event processing in action. http://dl.acm.org/doi/book/https://doi.org/10.5555/1894960

Fortino G, Savaglio C, Spezzano G, Zhou M (2021) Internet of things as system of systems: a review of methodologies, frameworks, platforms, and tools. IEEE Trans Syst Man Cybern Syst 51(1):223–236. https://doi.org/10.1109/TSMC.2020.3042898

Fowler M (2015) Microservice trade-offs. https://martinfowler.com/articles/microservice-trade-offs.html. Accessed 18 Jul 2022

Gao F, Zaremba M, Bhiri S, Derguerch W (2011) Extending BPMN 2.0 with sensor and smart device business functions. In: Proceedings of the 20th IEEE International workshops on enabling technologies: infrastructure for collaborative enterprises, pp 297–302

Grier RA (2015) How high is high? a meta-analysis of NASA-TLX global workload scores. Proc Hum Factor Ergonom Soc 59(1):1727–1731. https://doi.org/10.1177/1541931215591373

Harmon P, Wolf C (2011) Business process modeling survey. Business process trends. http://www.bptrends.com/surveys/Process_Modeling_Survey-Dec_11_FINAL.pdf. Accessed 18 Jul 2022

Hart SG, Staveland LE (1988) Development of NASA-TLX (task load index): results of empirical and theoretical research. Advances in psychology. Elsevier, Netherlands, pp 139–183

Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Q 28(1):75–105. https://doi.org/10.2307/25148625

Hürsch W, Lopes C (1995) Separation of concerns. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.5223. Accessed 18 Jul 2022

Janiesch C, Koschmider A, Mecella M, Weber B, Burattin A, di Ciccio C, Fortino G, Gal A, Kannengiesser U, Leotta F, Mannhardt F, Marrella A, Mendling J, Oberweis A, Reichert M, Rinderle-Ma S, Serral E, Song W, Su J, Zhang L (2020) The internet of things meets business process management: a manifesto. IEEE Syst Man Cybern Mag 6(4):34–44. https://doi.org/10.1109/msmc.2020.3003135

Khan A, Pohl M, Bosse S, Hart SW, Turowski K (2017) A holistic view of the IoT process from sensors to the business value. In: Proceedings of the 2nd international conference on internet of things, big data and security, pp 392–399

Larman C, Basili VR (2003) Iterative and incremental development: a brief history. Comput 36(6):47–56. https://doi.org/10.1109/MC.2003.1204375

Leopold H, Mendling J, Günther O (2015) Learning from quality issues of BPMN models from industry. IEEE Softw 33:26–33

Lewis J, Fowler M (2014) Microservices. https://martinfowler.com/articles/microservices.html. Accessed 18 Jul 2022

Nysetvold AG, Krogstie J (2005) Assessing business processing modeling languages using a generic quality framework. Adv Top Database Res 5:79–93. https://doi.org/10.4018/9781591409359.ch005.ch000

Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. J Manag Inf Syst 24(3):45–77. https://doi.org/10.2753/MIS0742-1222240302

Peffers K, Rothenberger M, Tuunanen T, Vaezi R (2012) Design science research evaluation. In: International conference on design science research in information systems, 7286 LNCS. Springer, Heidelberg, pp 398–410

Perera C, Zaslavsky A, Christen P, Georgakopoulos D (2014) Context aware computing for the internet of things: a survey. IEEE Commun Surv Tutor 16(1):414–454. https://doi.org/10.1109/SURV.2013.042313.00197

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131–164. https://doi.org/10.1007/s10664-008-9102-8

Sasirekha S, Swamynathan S (2016) Collaboration of IoT devices using semantically enabled resource oriented middleware. In: ACM international conference proceeding series, pp 98–105

Serral E, de Smedt J, Snoeck M, Vanthienen J (2015) Context-adaptive Petri nets: supporting adaptation for the execution context. Expert Syst Appl 42(23):9307–9317. https://doi.org/10.1016/j.eswa.2015.08.004

Suri K, Gaaloul W, Cuccuru A, Gerard S (2017) Semantic framework for internet of things-aware business process development. In: Proceedings of the IEEE 26th International conference on enabling technologies: infrastructure for collaborative enterprises, pp 214–219

Taylor K, Leidinger L (2011) Ontology-driven complex event processing in heterogeneous sensor networks. In: Extended semantic web conference, 6643 LNCS(part 2). Springer, Heidelberg, pp 285–299

Torres V, Serral E, Valderas P, Pelechano V, Grefen P (2020) Modeling of IoT devices in business processes: a systematic mapping study. In: Proceedings of the IEEE 22nd conference on business informatics, pp 221–230

Tu M, Lim MK, Yang MF (2018) IoT-based production logistics and supply chain system – part 1: modeling IoT-based manufacturing supply chain. Ind Manag Data Syst 118(1):65–95. https://doi.org/10.1108/IMDS-11-2016-0503

Valderas P, Torres V, Serral E (2022) Modelling and executing IoT-enhanced business processes through BPMN and microservices. J Syst Softw. https://doi.org/10.1016/j.jss.2021.111139

Valero C, Ruiz-Altisent M (2000) Design guidelines for a quality assessment system of fresh fruits in fruit centers and hypermarkets. Agricultural engineering international: CIGR J Sci Res

Dev. https://ecommons.cornell.edu/handle/1813/10210. Accessed 18 Jul 2022

Weske M (2012) Business process management: concepts, languages, architectures, 2nd edn. Springer, Heidelberg

Yousfi A, Batoulis K, Weske M (2019) Achieving business process improvement via ubiquitous decision-aware business processes. ACM Trans Internet Technol 19(1):1–19. https://doi.org/10.1145/3298986