# Modelling and executing IoT-enhanced business processes through BPMN and microservices

**Pedro Valderas[*], Victoria Torres[*] and Estefanía Serral[**]**

[*]PROS Research Centre, Universitat Politècnica de València, Spain
{pvalderas, vtorres}@pros.upv.es
[**]LIRIS, KU Leuven, Belgium
estefania.serralasensio@kuleuven.be

**Abstract**. The Internet of Things enables to connect the physical world to digital business processes (BP) and allows a BP to 1) consider real-world data to take more informed business decisions, 2) automate and/or improve BP tasks, and 3) adapt itself to the physical execution environment. We refer to these processes as IoT-enhanced BPs. Although numerous researchers have studied this subject, there are still some challenges to be faced. For instance, the need of a modelling solution that does not increase the notation complexity to facilitate further analysis and engineering decision making, or an execution approach that provides a high degree of independence between the process and the underlying IoT device technology. The objective of this work is defining an approach that (1) considers important intrinsic characteristics of IoT-enhanced BPs at modelling level without growing the complexity of the modelling language, and (2) facilitates the execution of the IoT-enhanced BPs represented in models independently from IoT devices' technology. To do so, we present a modelling approach that uses standard BPMN concepts to model IoT-enhanced BPs without modifying its metamodel. It applies the Separation of Concern (SoC) design principle: BPMN is used to describe IoT-enhanced BPs while low-level real-world data is captured in an ontology. Finally, a microservice architecture is proposed to execute BPMN models and facilitate its integration with the physical world. This architecture provides high flexibility to support multiples IoT device technologies as well as their evolution and maintenance. The evaluation done allows us to conclude that the application of the SoC principle using BPMN and ontologies facilitates the definition of intrinsic characteristics of IoT-enhanced BPs without increasing the complexity of the BPMN metamodel. Besides, the proposed microservice architecture provides a high degree of decoupling between the created models and the underlying IoT technology.

**Keywords:** IoT, BPMN, microservices

## 1 Introduction

Nowadays, it is increasingly common to find physical computing devices supporting all kind of business activities (e.g., monitoring vital signs, tracking products' location, measuring temperature and humidity in a building or a field, or controlling production units at factories). These computing devices rely on the so-called *context*, i.e., relevant data from the physical world (Abowd et al, 1999; Dey, 2001), to perform either a sensing or actuating task over it. While sensors are used to collect and transfer data about the physical world (e.g., temperature sensor, camera, hearth rate sensor, etc.), actuators are used to control the physical world (e.g., air conditioner or heating, watering systems, security systems, etc.).

Business Processes (BPs) defining and implementing company's goals can clearly benefit from the IoT domain (Zhang et al., 2011; Jalali & Wohlin, 2012; Janiesch et al. 2020; Beverungen et al., 2020). On the one hand, sensors can provide BPs with real-time data to take more informed decisions based on context (i.e., relevant data from the physical world) (Janiesch et al. 2020). For instance, the completion of manual

activities can be detected automatically through sensors, preventing the need of humans to manually indicate when they have finished a task. On the other hand, actuators can be used as digitalized physical resources that participate processes as artificial actors to automate and improve the execution of some of their tasks (Beverungen et al., 2020). For instance, bridges on a port can be automatically opened upon arrival of a ship. In contrast to traditional BPs where context data are entered manually by humans, in the IoT domain there is a shift to automation, where services, machines, and things can take the role and responsibility of performing some of the process tasks. We refer to this type of BPs as *IoT-enhanced BPs* (Torres et al., 2020), which are processes that make use of IoT technology to carry out the process tasks in order to achieve a specific goal.

However, these two domains (IoT and BPM) operate at a very different abstraction levels, what imposes some challenges to be solved (Janiesch et al., 2020). Next, we illustrate the intrinsic characteristics of IoT-enhanced BPs in which we focus in this work by using a real-life example, specifically showing the problems that we face in this article.

### 1.1 Motivating example

Let us illustrate the IoT-enhanced BP concept with an example from the logistics domain, specifically to transport perishable products whose safety and quality highly depend on controlling temperature and humidity from origin (harvest fields) to consumption (Bowman et al., 2009). Note that this IoT-enhanced BP will be used as a running example in the rest of the paper to illustrate the use of our approach.

Imagine a smart distribution centre, where received products from warehouses are distributed to supermarkets. Following the quality control proposal presented in (Valero & Ruiz-Altisent, 2000), perishable products are checked and stored prior to their distribution. Figure 1 presents, in a BPMN model, the flow of such process. The process starts when a container with a pallet of a same product arrives to the smart distribution centre.

The first thing to do at the distribution centre is to check the quality of the products of the pallet (level of firmness, colour, and possible damages). This is done by a worker who is in charge of registering the results of the checking in the system. Next, information about the product received (e.g., product name, product variety, harvest date, etc.) is automatically identified by reading the pallet labels (e.g., a QR code), and the conditions in which the products have been transported, i.e., the container's temperature and humidity, are also automatically sensed. Based on these conditions and the quality evaluation introduced by the worker, 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 centre and placed into a transportation line to be stored in a cooling chamber climatized 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 moulds, yeast, and certain bacteria have 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 colour 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.

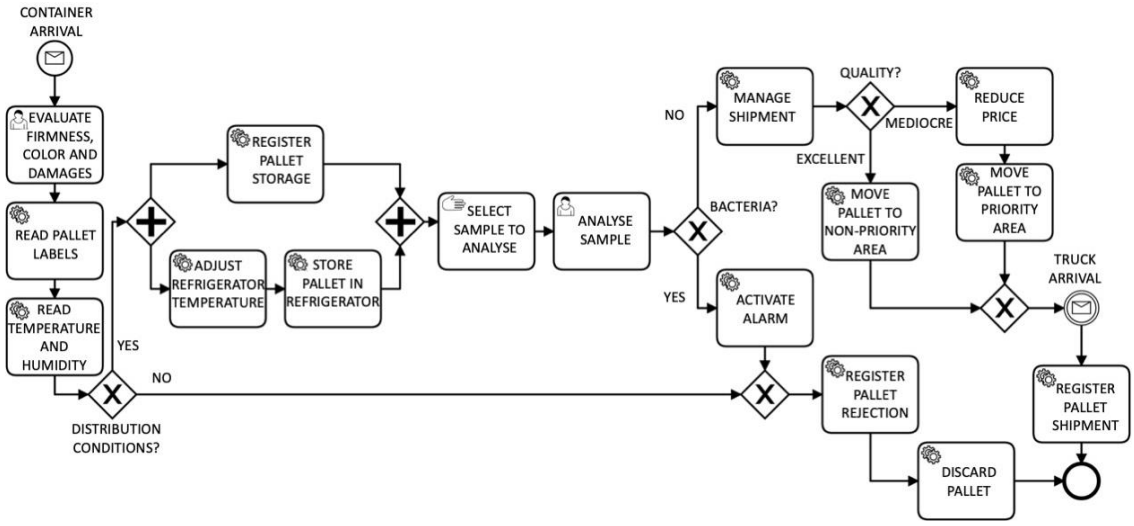


**Figure 1.** BPMN representation of the motivating example

While the container arrival start event as well as the detection of available trucks would be considered as a *push interactions*, i.e., the physical world injects data into the process, all the tasks that require interaction with IoT devices by demand (e.g., sensing container conditions, control refrigerator configuration, request robots to move pallets, etc.) are classified as *pull interactions*, i.e., tasks that are triggered from the BP. This interaction can be performed either to get information about the context in specific moments, i.e., to sense the environment conditions (e.g., container temperature and humidity), or to perform actions that may change the context, i.e., to actuate over "things" to change the environment conditions (e.g., to adjust the refrigerator chamber temperature).

Physical devices and context play an important role in the execution of IoT-enhanced BPs as Figure 1 shows. However, the general-purpose nature of BPMN does not allow to specify explicitly these important aspects, putting the focus instead on the specification of the control flow. A similar thing happens if we use other well-known modelling languages such as BPEL, Petri Nets (PNs), EPC, etc. It is not clear how IoT-specific aspects can be represented in a BP model since the semantics provided by the constructs of these languages were not conceived originally to address the specific necessities introduced by the IoT domain. In other words, BP modelling languages do not provide means to explicitly represent the IoT devices that execute each action, the context that needs to be sensed, or the events that are triggered from the physical world.

Note also that the lack of this information limits the possibility of executing a BP model that needs to interact with the physical world, since there is not enough information to do so. Thus, models like the one shown in Figure 1 will be relegated to simple documentation purposes. Therefore, we need to investigate how IoT-specific aspects can be integrated into the description of a BP in order to facilitate both the modelling and proper execution of an IoT-enhanced BP. Among the different characteristics that are intrinsic to an IoT-enhanced BP, this paper focuses on supporting the ones illustrated with the motivating example, which can be generalized to any IoT-enhanced BP. In particular:

1) As traditional BPs, the *flow of coordinated tasks* that are required to achieve the goal of the BP.
2) The *IoT devices* that participate in the BP.
3) The *context data* that need to be considered in order to maintain the BP well informed and to properly execute the flow of activities.
4) The *pull and push interactions* that occur between the BP and the physical world to contribute to the achievement of the BP goal.

Obviously, these characteristics can directly be moved to the execution level. Business process execution refers to the actual run of a process by a process engine, which is responsible for instantiating and controlling the execution of BPs (Weske, 2012). Thus, BP engines must be able to consider the above introduced characteristics to properly execute IoT-enhanced BPs. On the one hand, tools to control the flow of the process are needed. On the other hand, mechanisms to execute actions on IoT devices (pull interactions) and inject context changes into the BP (push interactions) must be provided.

A lot of efforts have been done to extend a well-known BP modelling language such as BPMN to integrate it with IoT requirements. However, as we conclude further in the analysis of the state of the art, most of the existing solutions present two main drawbacks: (1) extensions introduced in BPMN do not support all the intrinsic characteristics presented above or highly increase the complexity of the modelling language, which hinders the use of BPMN models as communication tools among stakeholders; (2) the proposed solutions to model IoT-enhanced BPs are not executable or the provided execution mechanisms are highly dependent on specific IoT technologies, making it difficult to evolve the system when technological changes are needed.

### 1.2 Problem statement

The problems that this work addresses can be stated by the following research questions:

1. *How can we consider all the intrinsic characteristics of IoT-enhanced BPs at modelling level without growing the complexity of the BP modelling language?*
2. *How can we execute IoT-enhanced business processes that are represented in BP models independently from IoT devices' technology?*

### 1.3 Main Contributions

This work improves the state of the art by proposing a solution to model and execute IoT-enhanced BPs that addresses the two above stated problems. This solution is based on the two following decisions. On the one hand, we propose to use the Business Process Model and Notation (BPMN), a well-known and accepted standard by academia and industry, as the modelling language to represent such BPs. On the other hand, we propose to deploy BPs following a microservice architecture to execute these BPs, which provides a high degree of independence from IoT technology as well as flexibility to evolve and maintain the system.

As such, the contributions of this paper are two-fold:

(1) A modelling approach based on BPMN that reuses the concepts introduced by this language in order to model IoT-enhanced BPs. In order to not increase the complexity of the BP modelling task, we analyse the constructors provided by the BPMN metamodel and define a proposal to specify IoT devices and pull interactions without modifying its metamodel. In addition, we apply the SoC design principle in order to complement BPMN models with an ontology that is used to model context and push interactions.

(2) A microservices architecture aimed at facilitating the integration of business processes with the physical world. This architecture provides a high degree of decoupling between the created models and the underlying IoT technologies. This facilitates the integration of IoT devices that are supported by different technologies.

### 1.4 Structure of the paper

The remainder of the paper is structured as follows. Section 2 presents some background. Sections 3 introduces an analysis of the state of the art. Section 4 outlines the modelling solution to specify IoT-enhanced BPs. Section 5 presents the microservice architecture designed to support the execution of such models. Section 6, 7 and 8 present the experiments done to evaluate our work. Afterwards, a discussion about the proposed solution is presented in Section 9. Finally, Section 10 concludes the paper and provides insights into directions for future work.

## 2 Background

This section provides some background on which the proposed work is based. This include Business Process Modeling, Context and Ontologies and Microservices.

**Business Process Modelling**. It is the activity where business processes are explicitly represented using a graphical notation (e.g., BPMN). Specifically, according to Weske (2012), a business process is defined as "a set of activities performed in coordination in an organizational and technical environment". Analysing this definition, a business process defines what (activities) has to be performed, how they should be performed, and by whom (organizational and technical environment). Currently, there are many languages that can be used to build a BP model. The most used and well-known BP modeling languages include BPMN, BPEL, Petri Nets (PNs), EPC, and UML Activity Diagrams (AD). The level and the purpose of the BP model being created will determine the modeling language used in each case. For example, while the BPMN modeling language is more appropriate to represent high-level BPs, PNs work better for low-level BPs that in addition can be analyzed from a mathematical point of view. As commented above, in this work we use BPMN in order to represent IoT-Enhanced BPs at a high level of abstraction.

**Context and Ontologies**. In a business process, context can be defined as the minimum set of variables that contains all the important information that impacts their design, implementation, and execution (Rosemann & Recker, 2006). Although an exhaustive recent comparison of techniques to model context can be found in Perera et al. (2013), ontologies are one of the best choices to do so (Baldauf et al., 2007; Chen et al., 2003; Ye et al., 2007). In the context of IoT-enhanced BPs, ontologies are also one of the most used solutions to extend BPs with context data (Gao et al., 2011; Suri et al., 2017). Several ontologies already exist to describe sensor devices and the data they capture. For instance, the Sensor Model Language (SensorML[1]) which focuses on describing physical and functional characteristics of physical processes focusing on the process of measurement and observation. The Semantic Sensor Network (SSN[2]) ontology which can describe sensors in terms of capabilities, measurement processes, observations and deployments. The IoT-Lite[3] ontology, which is a lightweight instantiation of SSN to represent IoT resources, entities and services. The Sensor, Observation, Sample and Actuator (SOSA[4]) ontology, which is based on SSN, and describes sensor observations, sampling, actuations and procedures. Finally, the Stream Annotation Ontology (SAO[5]) that can be used to represent IoT data streams.

**Microservices**. They are the key pillar of an architectural style where applications are decomposed into small independent building blocks (the microservices), each of them focused on a single business capability (Fowler & Lewis, 2014). Microservices communicate to each other with lightweight mechanisms, and they can be deployed and evolved independently, which leads to more agile developments and technological independence between them (Fowler, 2015). Apart from the microservices that implement the business capabilities of a system (hereafter business microservices), a microservice architecture usually includes other microservices that are focused on supporting infrastructure issues. Examples of this type of microservices are the *Service Registry* that gives support to service discovery, containing the network locations of microservice instances. In addition, some supporting tools are also provided to, for instance, monitor microservices' status, log their executions, or manage asynchronous communication among microservices (e.g., message brokers).

[1] https://www.ogc.org/standards/sensorml
[2] https://www.w3.org/TR/vocab-ssn/
[3] https://www.w3.org/Submission/2015/SUBM-iot-lite-20151126/
[4] https://github.com/w3c/sdw/blob/gh-pages/ssn/integrated/sosa.ttl
[5] http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/sao

# 3 State of the art

This section presents the state of the art related to the integration of IoT capabilities at the BP modelling level (cf. Sections 0 and 3.2) and also reviews the works that apply the SoC design principle to their proposals (cf. Section 3.3). Finally, a summary of this analysis is presented (cf. Section 3.4).

## 3.1 Extending the BPMN metamodel with new constructs

There are some works that extend the original BPMN notation with new concepts to model requirements imposed by IoT systems. Appel et al. (2014) introduce a new concept called Event Stream Processing Units (SPUs) to integrate the amount of real-time data that is generated in Cyber-physical systems. Mandal et al. (2017) introduce new event annotations to specify the binding points between external events and the BP model. Chiu & Wang (2015) extend the event concept with new types of events. Dörndorfer & Seel (2018) extend the BPMN metamodel with four new elements in order to represent context data: Context Description Expression, Intermediate Context Event, Context Annotation, and Context Group. Schönig et al. (2018) extend BPMN with Data variables to enrich BP models with data obtained from physical objects and to specify how and where connected IoT devices influence the process. Cheng et al. (2019) introduce three new classes (Sensor Device, Sensor Service, and Handler) to explicitly represent sensor devices. Graja et al. (2016) propose to specialize BPMN service tasks as physical tasks and cyber tasks, which are tasks that are executed by a piece of software. Meyer et al. (2013) extend the BPMN metamodel with three new classes: PhysicalEntity, SensingTask and ActuatingTask. Petrasch & Hentschke (2016) propose to represent IoT devices as BPMN partitions in the BP model. Then, based on the task extension proposed by Meyer et al. (2013), the interaction with such devices is represented through Sensing Tasks and Actuating Tasks. Mottola et al. (2017) introduce the WSN task concept to represent generic actions to sense, actuate, and aggregate operations executed by a Wireless Sensor Network. Sperner et al. (2011) propose to extend the BPMN metamodel to represent physical entities and their interaction with devices with several new concepts: PhysicalObject, SensingTask, ActuatingTask, SensingAssociation and ActuatingAssociation. Suri et al. (2017) extend BPMN with the ResourceExtension to include IoT Devices and their quality attributes. Yousfi et al. (2019) propose to extend the event and task BPMN elements to represent IoT input technologies, and the data object element to represent physical objects. Finally, Gao at al. (2011) propose to extend the BPMN model by attaching an extended attribute to a task, tasks groups and sub-processes, and using this attribute to reference an external model.

The main drawback of all these approaches is that extending the BPMN's metamodel prevents current existing tools to be used to execute IoT-enhanced business processes. As commented above, to support the execution of these processes we need tools to control the flow of the process. Currently, there exists a myriad of BPMN engines that could be used to support this task. However, if modelling solutions modify BPMN's metamodel they are no longer compatible with existing engines. In this sense, most of the above-introduced approaches that use BPMN do not support the execution of their modelling solution. Some exceptions are Appel et al. (2014) and Cheng et al. (2019), which extend an existing process engine to support their new constructs; and Mottola et al. (2017), which generate code to execute the part of the BP model that cannot be executed in a process engine. We think that these solutions provide execution platforms that are extremely coupled with a specific technology and that are difficult to be maintained and evolved if technology requirements change.

In addition, handling IoT concepts into BP modelling can introduce a higher cognitive complexity. In the BPM field, we find many works aimed at measuring this complexity associated to BP modelling (Melcher et al., 2019; Zugal et al., 2011; Zugal et al., 2011, October). These works put forward that more conceptual abstractions could make BP modelling more complicated and could create the need for additional validation mechanisms. Note also that one of the major purposes of a BP model is to serve as an efficient communication mechanism between different BP stakeholders (e.g., between business engineers

and process designers). If BP models are extended or enriched with too much information, we run the risk of losing this powerful mechanism.

### 3.2 Using BPMN models as-is

In contrast to extending the BPMN metamodel, other studies propose to use the original BPMN constructs to model IoT-enhanced business processes. In this case, the BPMN notation is mainly used just for the construction of a non-executable modelling artefact that needs to be transformed into another language/technology to be executed. For example, Baresi et al. (2016) propose defining BPs in the BPMN standard and generating from them declarative extended Guard-Stage-Milestone (GSM) specifications in a semi-automatic way. Such GSM specifications are deployed and executed on smart objects and a specific infrastructure needs to be deployed in each of them. Caracaş & Kramp (2011) propose the use of standard BPMN tasks, flows and pools to capture the behavioural aspects of WSN applications. Then, Java and C# code is generated in order to be deployed on the Mote Runner WSN platform. Dar et al. (2015) propose integrating smart objects into the BP using jBPM, a BPMN compliant software suite where the application logic is expressed by composing local and remote service tasks using the BPMN based workflow model. In this case, a programming framework developed in Java behaves as intermediary between these smart objects and the BP definition, and generates all the artefacts that are necessary to achieve this integration. Domingos & Martins (2017) propose to use the BPMN Performer and Resource classes to represent IoT devices and integrate information about them into the model. Then, standard BPMN models are translated into Callas, an IoT neutral-platform programming code that can be executed in every IoT device for which there is a Callas virtual machine available. Friedow et al. (2018) propose to define BPs at the process layer using standard BPMN and achieve the integration between IoT devices and BPs at the technical level through the Bosch IoT Things service. Finally, Wehlitz et al. (2017) propose a work-in-progress architectonic solution that uses BPMN to deal with the modelling and execution of IoT-enhanced BPs for smart environments. In this case devices are represented as resources (swim lanes) in a BPMN model and use service tasks to manage them.

The major benefits of these approaches are that 1) they do not increase the cognitive complexity of the modelling notation and 2) they can use existing BPMN engines to execute the BPs. However, most of the works following this approach use the BPMN notation just as a modelling artefact that needs to be transformed into another language/technology to be executed. Again, this solution is highly coupled with specific technologies, and a more flexible solution should be proposed. For instance, Baresi et al. (2016) require deploying a specific infrastructure in each IoT device making them totally dependent from this infrastructure. Domingos & Martins (2017) only support IoT devices for which a Callas virtual machine is available; Caracaş & Kramp (2011) generate Java and C# code to be deployed on the Mote Runner WSN platform; and Dar et al. (2015) depends on the jBPM toolkit in order to interact with a Java framework. An exception is the work proposed by Wehlitz et al. (2017) which uses the external task pattern to interact with functionality implemented in any technology. However, this work obviates the processing of low-level context data and delegate this problem to programming artefacts in contrast to our work that face it through high-level descriptions based on an ontology. In general, the description of context at modelling level is not considered by the above introduced approaches and they pay little attention to describe how the low-level data obtained from sensors (e.g., container's temperature) can be processed to obtain high-level information data that may be more appropriate for the BP (e.g., damaged goods due to high temperatures).

Our approach also proposes to specify IoT-enhanced BPs using the original primitives of the BPMN language. However, in our case, BPs models can be deployed and executed in any BPMN compliance engine, independently of the technology in which IoT devices work with. This technological flexibility is possible thanks to the microservice architecture in which our proposal relies on (cf. Section 5). In our proposal , microservices behave as intermediaries between the BP and the IoT devices participating in the BP. Even though microservices provide a standard way to interact with the corresponding IoT devices (i.e., by means of an API), these can be implemented in different languages and frameworks that may be more

appropriate depending on the type of device being wrapped (e.g., Swift to handle iOS devices, Python for Raspberry).

### 3.3 Application of the SoC principle

Some of the works presented above apply the SoC design principle to design their modelling proposals. Dörndorfer & Seel (2018) propose the sensor model (SenSoMod) to specify sensors, context and how these relate to each other. This model is linked with the proposed BPMN extension (Context4BPMN). Gao et al. (2011) propose linking BPMN models with the Functional Model to import a sensor ontology and its instance data. Suri et al. (2017) propose providing a semantic description of the BPMN models by means of an ontology that integrates concepts from the BP and the IoT domains. Finally, Yousfi et al. (2019) combine their BPMN extended proposal (uBPMN) with a Decision Model where ubiquitous decisions based on an important amount of data (e.g., location, traffic status, gas level, etc.) are defined.

Our approach applies the SoC principle by combining BPMN models with an ontology as most of the above analysed works do. However, in contrast to these works, the proposed ontology is only introduced to manage low-level context data. The IoT devices that participate in the BP and the high-level events that must be managed within the process are represented in the BPMN model by using the standard notation. Thus, the high-level requirements of an IoT-enhanced BP are all defined in one model, which provides a more intuitive and cohesive view to facilitate their analysis. Moreover, it is not clear how the above analysed works support the execution of the models they propose. We propose a microservices architecture to support the execution of IoT-enhanced BP models.

### 3.4 Summary

Table 1 shows a summary of the analysed works. The columns of this table are as follows: (1) *BPMN*: the modelling language is extended (+) or is used as is (=); (2) *IoT*: IoT devices and/or the interaction with them are considered at modelling level; (3) *Context data*: the management of context data at modelling level is supported; (4) *Execution support*: the execution of IoT-enhanced BPs is supported; (5) *Technology independent*: the execution of IoT-enhanced BP models is technology independent or need either a specific engine or a proprietary solution; and (6) *SoC*: the approach applies the SoC principle to avoid increasing the complexity of BPMN models.

**Table 1.** Comparison of the analysed works

| Approach | BPMN | IoT | Context Data | Execution Support | Technology independent | SoC |
|---|---|---|---|---|---|---|
| Graja et al. (2016); Meyer et al. (2013); Petrasch & Hentschke (2016); Sperner et al. (2011) | + | yes | no | no | - | no |
| Mandal et al. (2017); Chiu & Wang (2015); Schönig et al. (2018); | + | no | yes | no | - | no |
| Suri et al. (2017); Gao (2011); Dörndorfer & Seel (2018); Yousfi et al. (2019) | + | yes | yes | no | - | yes |
| Appel et al. (2014) | + | no | yes | yes | No. Ext. engine | no |
| Cheng et al. (2019) | + | yes | no | yes | No. Ext. engine | no |
| Mottola et al. (2017) | + | yes | no | yes | No. Prop. code | no |
| Baresi et al. (2016) | = | yes | no | yes | No. Prop. Infrastr. | no |
| Caracaș & Kramp (2011) | = | yes | no | yes | No. Mote Runner | no |
| Dar et al. (2015) | = | yes | no | yes | No. jBPM toolkit | no |
| Domingos & Martins (2017) | = | yes | no | yes | No. Callas | no |
| Friedow et al. (2018) | = | yes | no | yes | No. Bosch IoT | no |
| Wehlitz et al. (2017) | = | yes | no | yes | yes | no |

Our approach provides the following key contributions to the state of the art:

1. Our modelling approach provides an integrated and cohesive solution capable of representing: (1) the flow of coordinated activities; (2) the IoT devices that participate in the BP; (3) the context data that need to be considered at both low and high level, and (4) the pull and push interactions that may occur between the BP and the IoT devices. To do so, we based on a BPMN model and a context ontology.
2. Our modelling approach does not increase the complexity of the BPMN metamodel and is compatible with existing BPMN process engines. To do so, we apply the SoC principle in order to integrate a BPMN model with a context ontology without extending the BPMN metamodel.
3. The execution of IoT-enhanced BPs is supported in such a way that the created models are highly decoupled from the underlying IoT technology. To do so, we provide an execution architecture based on microservices.

Next sections introduce the solution that we propose to model and execute IoT-enhanced BPs in detail.

# 4 A modelling approach for IoT-enhanced BPs

In this section, we present a modelling solution to describe IoT-enhanced BPs that pays special attention to support the intrinsic characteristics described in Section 1 without increasing the complexity of BPMN models. To do so, we apply the SoC design principle and propose a modelling solution based on two models, a BPMN model and an ontology-based context model. Thus, the modelling process that we propose is defined by two main steps:

1 *Business Process modelling*. We create a standard BPMN model that describes the flow of coordinated activities among participants of different type (e.g., humans and IoT devices) considering both pull interaction with IoT devices and push interactions triggered from high-level context events. How low-level context is sensed and processed to create high-level events is considered in the next step.

2 *Context Definition*. An ontology-based context model is used to define the low-level data that must be sensed from the physical world, and how it must be processed to inject (push interaction) the high-level events into the IoT-enhanced BP defined in the BPMN model.

Note that the application of the SoC principle also allows us to support multidisciplinary working groups: experts on the capture of the low-level environmental data can focus on defining the context ontology, while process engineers can focus on defining the BPMN model. Next subsections explain each step in more detail.

## 4.1 Business Process modelling

IoT devices and the interaction with them should be represented in a BPMN model without extending its metamodel. To do so, existing BPMN concepts must be used. The main foundations of our modelling approach are as follows:

**Explicit representation of IoT devices.** If we consider the approaches analysed in the state of the art, we can see that the most used solution to achieve this consists in using the *pool* or *lane* concepts in order to represent a device. See for instance, works such as Cheng et al. (2019), Petrasch & Hentschke (2016) or Wehlitz et al. (2017). We were inspired by these approaches when proposing our modelling solution. According to good practices in BPMN, pools should be used to represent organizations, and lanes to represent the actors of an organization that participate in a process. Thus:

- Guideline 1. A pool is used to represent the whole IoT-enhance business process within an organization.
- Guideline 2. Each IoT device or any other actor of an organization that participate in the process is represented by a lane of this pool.

**Representation of IoT Devices' actions**. In BPMN, the tasks that are contained within a lane define the actions of the actor represented by the lane. According to the standard BPMN (2011), service tasks are

those carried out by software. Therefore, in the case of IoT devices, we think that such tasks are the best option to represent their actions. BPMN normally assumes that this software is developed as a web service, though it can be implemented differently. In our solution, IoT devices are going to be managed by microservices (further explained in Section 4). Therefore, a BPMN Service Task is a very suitable BPMN element to represent IoT devices' actions since they are conceptually defined to be linked to an API provided by an external system. Thus:

- Guideline 3. Each IoT devices' action is defined as a Service Task.

**Supporting pull interaction.** A push interaction occurs when the BP receives context data from the physical world, which triggers events and injects data into the BP upon their occurrence. This type of interaction can be considered as an event-driven communication where the BP is waiting for the occurrence of interesting events that occur in the physical word.

- Guideline 4. The execution of the Service Tasks that represent actions of IoT devices supports pull interactions.

**Supporting push interaction.** A push interaction is done when the BP receives context data that is waiting for from the physical world. In this case, the data is injected into the BP from the physical world. This type of interaction can be considered as an event-driven communication. The BP is interested in the events that occur in the physical word and it is waiting for the occurrence of these events. The physical world is the element that triggers events and informs the BP upon their occurrence.

BPMN provides the *message start event* and the *message intermediate catch event* to define that a process must wait for the reception of an event to either be started or to continue its execution after pausing it, respectively. These elements can be used to represent that an IoT-enhanced BP must wait for the occurrence of an event in the physical world.

In order to represent that these events are generated from the physical world, we need to represent the physical world as a new actor of the process. However, note that we do not control the physical world, we just know that we receive events from it. To represent this situation in BPMN, a collapsed pool is recommended. Thus:

- Guideline 5. The physical world is represented by a collapsed pool.
- Guideline 6. Push interactions are represented by flow sequences whose source is the collapsed pool that represent the physical world and whose target is a *message start event* or a *message intermediate catch event* defined in a lane.

In Figure 2, we have used these six foundations to re-define the BPMN model used in the motivation example presented in Section 1.1. As we can see, there is a main pool that represents the "Smart Distribution Centre" and that it is divided in seven lanes that represent the seven actors that participate in the process: the Information System lane that represents the software that performs actions on the data storage of the company, the Worker and Analyst lanes that represent human actors and finally, the rest of lanes that represent four IoT devices that participate in the process: the Robot, Refrigerator Control System, the Alarm, and the Truck Container Sensor. Besides, the re-defined model includes an extra task to have an additional push interaction, i.e., the Decrease Refrigerator Temperature included at the end of the flow which allows to adjust the temperature of the refrigerator chamber if it is too warm.

Note how IoT devices are represented in the same way as any other actor in the process. This aspect provides a high level of abstraction with respect to the physical world and provides a cohesive way of representing actors of any type (software systems, humans, IoT devices), which reduces the complexity of the model but maintains a high level of expressiveness. Note also that the physical world is explicitly represented, which allows easily identifying the events that are generated from it. These events are used to represent push interactions in the BPMN model and are defined at a high level of abstraction (e.g., container arrival, refrigerator too warm). How high-level events are obtained from low-level data sensed from the physical world is defined at the context ontology that we present in the next subsection.

This modelling solution provides an intuitive way of defining IoT-enhanced BPs that facilitate further analysis to take engineering decisions. It allows business process engineers to easily understand the pull interactions that may occur with the IoT devices that participate in the process; in the same way, push interactions are quickly identified through the high-level events associated to the physical world. In

addition, we describe the IoT-enhanced BP in an abstract way, highly independent from the underlying IoT technology.

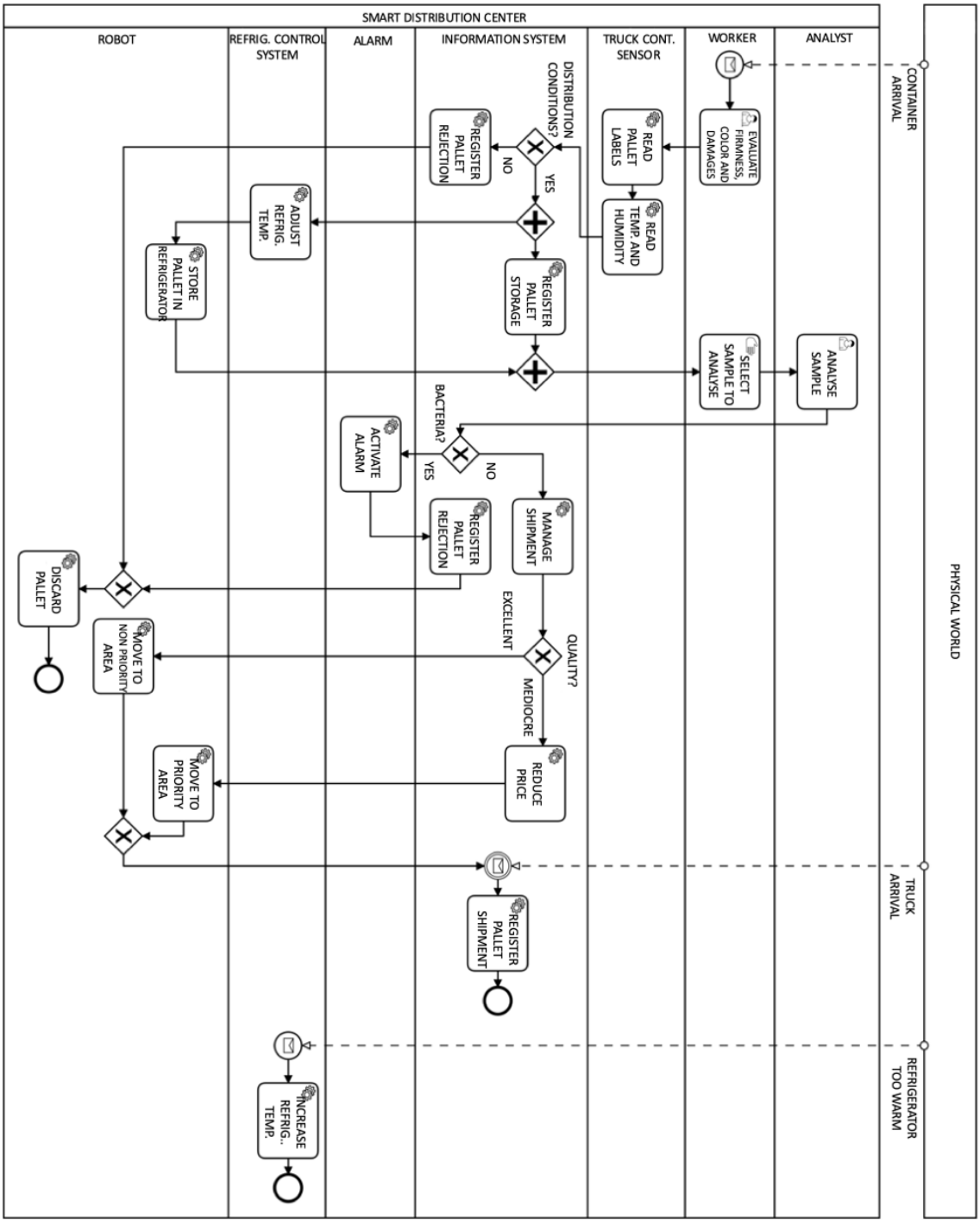


**Figure 2.** BPMN representation of an IoT-enhanced business process

### 4.2 Context definition

In this work, we use ontologies as a separated modelling artefact from BPs to model context in an IoT-enhanced process. This solution provides two main benefits. On the one hand, it facilitates to apply the SoC design principle which helps to not increase the complexity of BP models. On the other hand, ontologies provide a valuable mechanism to describe and analyse context data in order to obtain (as we explain below) the high-level data that is needed in the business process to be executed from low-level data captured by sensors installed in the physical world.

Considering IoT-Enhanced BPs, a context ontology should represent the digital and physical data impacting a business process. This includes the physical data itself but also the sensors that are used to capture such data, the characteristics of the data (e.g., time in which it was captured, format, and metric used), and other relevant digital data the company may have (e.g., data about products, customers, employees, and facilities). All these data are necessary since a business process may depend on data events that are not only derived from IoT data, e.g., 5 degrees is a too warm temperature for ice storage, but it is an appropriate temperature for preserving vegetables. For deriving these events, product information as well as temperature information in the storage room are needed.

To support the motivating example, we could combine the SOSA ontology with the logistics ontology created by Knoll et al. (2019), which offers constructs to describe products, their packaging, and the logistics processes. As a representative example, Figure 3 shows part of the ontology used in the motivating example. We can see some classes imported from the SOSA ontology including the `Observation` class. Its subclass `TemperatureObservation` is instantiated to capture the temperature of a container.

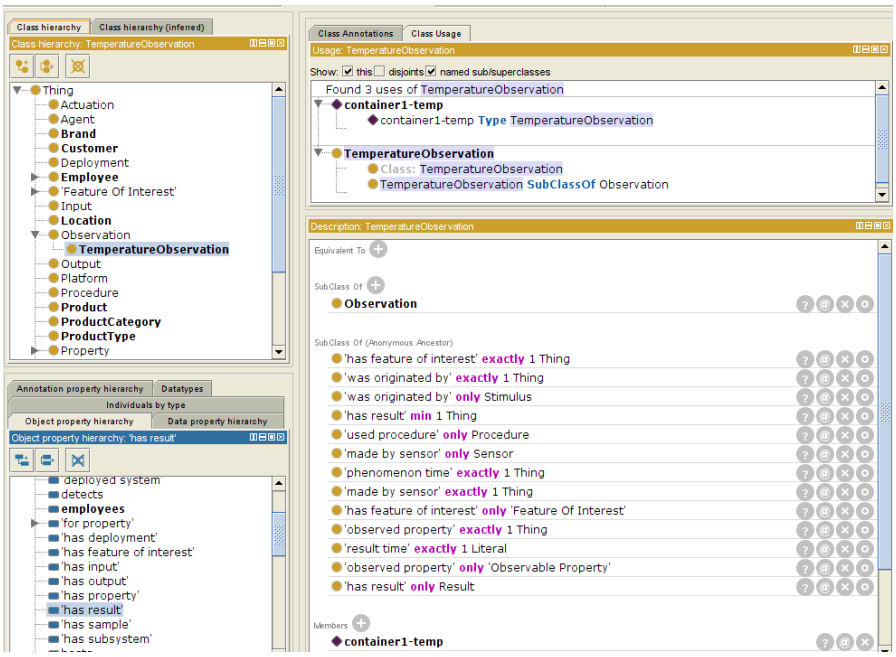


**Figure 3.** Snapshot of ontology used in the motivating example in Protégé

**Generating High-Level Events from Context Data.** In our approach, low-level context data must be processed to inject high-level events into the BPMN process (push interactions). This task is highly related to the Complex Event Processing (CEP) area, which focuses on offering an abstraction layer that hides the complexity in detecting such events. In this way, the business-level application, the BPMN process in our case, can concentrate on realizing appropriate actions whenever a specific event occurs. Specifically, we

followed the main ideas proposed by Taylor & Leidinger (2011) which demonstrated that ontologies are a valuable tool to represent context data in order to process complex events.

Complex Event Processors are supported by query languages that allow domain experts to describe when a relevant event occurs. Ontologies offer mechanisms such as SWRL rules or SPARQL queries to derive such events. These mechanisms can be used to transform low-level data into the high-level events that will be consumed by the process. As a representative example, the following two SWRL rules identify when a container has arrived at the warehouse as well as when the temperature of the warehouse is too warm for the products that are stored in that warehouse. Note that these two SWRL use low-level data of the environment to generate new knowledge that can be used to create the high-level events defined in the BPMN model presented above, i.e., *Container Arrival* and *Cooling Chamber Too Warm:*

```
Container(?container) ^ locatedIn(?container,ContainerReception)
                                    -> status(?container,\"arrivalInWarehouse\")
```

```
TemperatureObservation(?o) ^ hasResult(?o, ?result) ^ value(?result, ?v) ^
Product (?p) ^ maximumTemperature (?p, ?t) ^ swrlb:greaterThan (?v, ?t)
                                      -> temperatureCondition (?p, \"TooWarm\")
```

In the same way, SPARQL queries can be also used to generate high-level events from low-level context data. As representative example, the following query returns `true` when containers are not detected in a period of 5 minutes, which could be used, for instance, to generate a high-level event that puts the system in standby mode.

```
PREFIX ofn:<http://www.ontotext.com/sparql/functions/>
ASK {
  ?container rdf:type Container .
  arrivalTimeStamp ?timeStamp .
  ofn:millisBetween(NOW()^^xsd:dateTime,
                                      ?timeStamp^^xsd:dateTime) < 300000 .
}
```

As we can graphically see in Figure 4, the low-level context data produced by IoT devices are registered into the Context Ontology. Next, SWRL and SPARQL can be used to analyse this low-level data in order to generate the high-level events defined in the BPMN model.

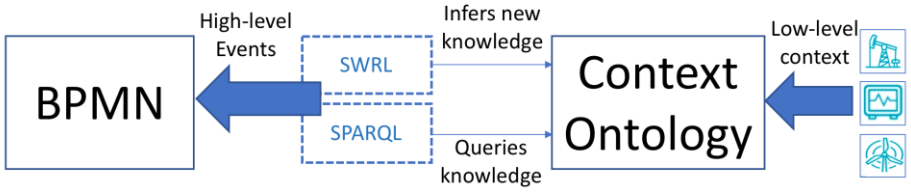


**Figure 4.** From low-level context data to high-level BPMN events

The next section introduces a microservice architecture in which we can see how the low-level context data is asynchronously published by the microservices that manage IoT devices in an event bus, and how a Context Monitor microservice plays the role of Complex Event Processor in order to: analyse this context data, identify high-level events, and inject them into the business process.

## 5 Supporting microservice architecture

The previous section has presented a modelling solution to abstractly define an IoT-enhanced BP that provides high independence from the technology used by IoT devices. In this section, we present a microservice architecture in order to support the execution of these IoT-enhanced BPs. This architecture provides a solution to nimbly manage the technological heterogeneity of IoT devices and provides high flexibility to support evolution and maintenance. In addition, it facilitates the possibility of using existing BPMN engines in order to control the flow of the process. This architecture is characterized as follows (see Figure 5):

The **business microservices** are those in charge of managing the different actors of an IoT-enhanced BP. These microservices manage IoT devices. There is no restriction about how these microservices must be implemented. Any operating system or implementation technology can be used. Any type of IoT device can be managed by a microservice, independently of its supporting technology or manufacturer. There are only two requirements that a business microservice must satisfy in order to participate in the proposed architecture:

- It must provide a mechanism that allows another microservice to request, in a decoupled way, the execution of the operations that can be performed by the IoT device. This mechanism can be for instance a synchronous REST API or be based on an asynchronous communication through an event-based bus.
- It must publish any context change to consider within an IoT-enhanced BP into an event-based bus to check if this is later linked as a high-level event.

To analyse the published context changes, we propose the *Context Monitor* microservice, which will inject the necessary high-level events into the BP (see more details below). Regarding the **supporting tools**, we propose the use of an asynchronous event-based bus to support the communication between business microservices and the *Context Monitor* microservice.

The following **infrastructure microservices** is proposed:

- *Service Registry*: this microservice is in charge of maintaining the list of business microservices that are in the system. For each microservice, this registry stores its invocation data.
- *BP Controller*: this microservice is endowed with an existing BPMN engine that is in charge of controlling the activity flow of the process. Note that the modelling solution presented in the previous section is totally based on the standard so we can use any existing BPMN engine. This engine does not interact with business microservices directly. Instead, it sends execution requests to the *Action Performer* microservice presented below. To do so, the only restriction that we must consider is that the service tasks associated to the lanes that represent actors should be bind to the API provided by the *Action Performer* microservice. Note that BPMN engines provide their own mechanisms to associate an external API to a service task so we must use those provided by the selected engine.
- *Action Performer*: this microservice plays the role of middleware among the BP Controller, the Service Registry, and the business microservices. It must publish an API to which BPMN service tasks must be bind in order to execute an action. When a service task is executed by the BPMN engine of the BP Controller, this engine sends an execution request to the Action Performer, which interact with the Service Registry in order to know the invocation data of the required business microservice. Then, it calls the corresponding operation either through a direct REST call to the business microservice or by publishing a message into the event bus.
- *Context Monitor*: this microservice is registered in the event-based bus in order to access the context changes published by business microservices. When a context change happens, this microservice inserts it into the OWL context ontology and use SWRL rules and/or SPARQL queries such as the ones presented in Section 4.2 in order to generate high level events. These events are injected into the BPMN engine microservice. To do so, the mechanisms provided by the selected engine in order to interact with it from an external system must be used.

Finally, note that an IoT-enhanced business process can include user tasks, which are activities within a process that are performed by end-users through an external application. In these cases, the *BPController* microservice must interact with these end-user applications in order to launch these user tasks.

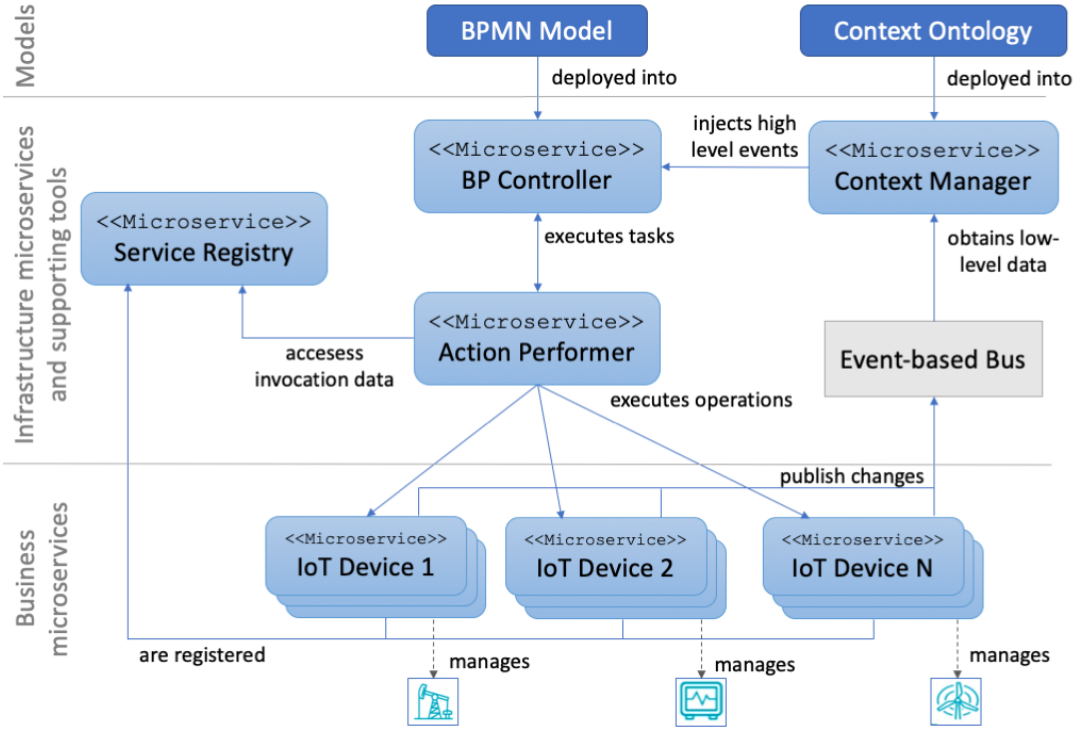


**Figure 5.** Microservice architecture to execute IoT-enhanced BPs

In what follows we explain the interaction among the architecture elements in more detail. Let us consider the initial tasks of the re-defined BP model shown in Figure 2, which need both push and pull interactions. The process starts with a push interaction through which a high-level event is injected from the physical world. This high-level event is generated by the *Context Monitor* microservice by analysing the context data published by IoT devices that may not participate in the process. In this case, a *Container Detector* IoT device is in charge of publishing the location of containers in the context ontology. Note that this IoT device does not participate directly in the business process and it is not aware of it. This device is just focused on publishing container's location. The *Context Monitor* decides if this data should generate a high-level event into a business process or not. Thus, note the high level of independence between the process and the IoT environment that our approach provides. Once the process is started, the user task *Evaluate level of firmness, colour, damages* is performed. Afterwards, the service task *Read pallet labels* is performed by the *Truck Container Sensor* microservice (pull interaction). The interaction among the participants in this part of the motivating example is as follows (see Figure 6):

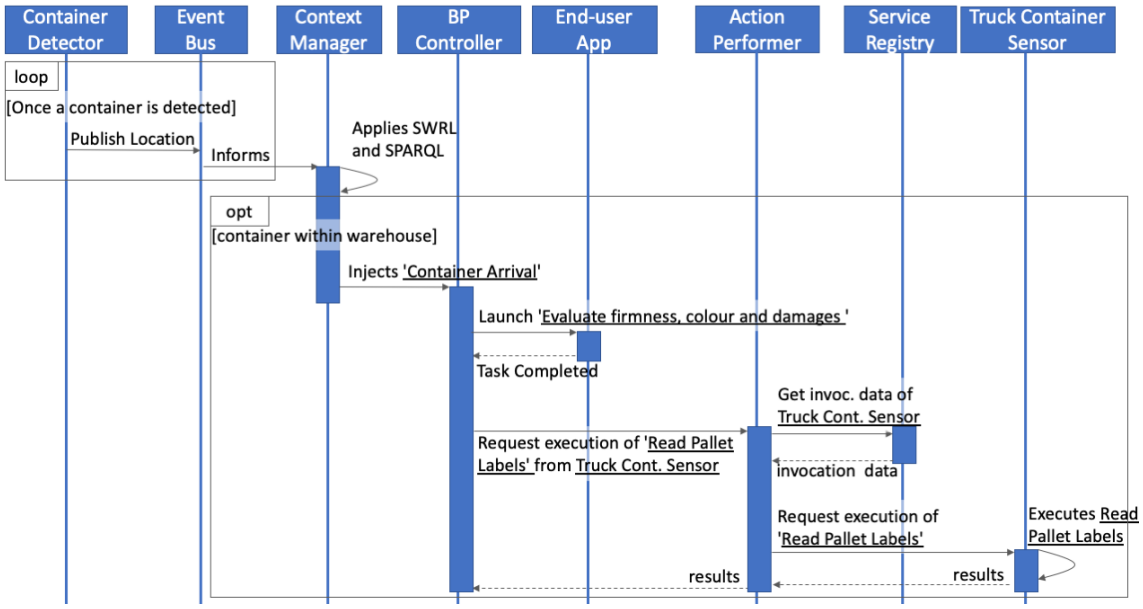


**Figure 6.** Example of interaction among architectural elements.

1. The *Container Detector* microservice starts to publish the location of a Container into the Event-based bus once it has been detected.
2. The *Context Manager* microservice is receiving these low-level context data updating the OWL ontology, and applying SWRL rules and SPARQL queries. When the location of the Container is the proper to consider the Container within the Warehouse, the *Context Manager* microservice injects the high-level event *Container arrival* into the *BP Controller*.
3. The *BP Controller* microservice starts executing the part of the process associated to this high-level event. According to Figure 2, the user task "Evaluate firmness, colour and damages" must be executed. To do so, the *BP Controller* microservice must interact with an external end-user application.
4. Once the external end-user application informs the *BPMN Engine* microservice about the completion of the user task, the service task "Read Pallet Labels" must be performed by the *Truck Container Sensor* microservice. To do so, the *BP Controller* delegates the execution this task to the *Action Performer* microservice.
5. The *Action Performer* microservice receives the request of executing the "Read temperature and humidity values" operation of the *Truck Container Sensor* microservice. To do so, it inquiries the *Service Registry* in order to know the invocation data of an instance of this microservice.
6. The *Action Performer* microservice uses the data obtained from the Service Registry in order to request the *Truck Container Sensor* microservice to execute the "Read temperature and humidity values" operation. In this example, we have considered that this microservice publishes a REST API, so the *Action Performer* microservice uses this API to directly interact with it. In case a microservice requires a publish/subscribe communication, the Service Registry provides the Action Performer the data required to communicate with the microservice through the event-base bus of the architecture.
7. The *Truck Container Sensor* microservice executes the requested operation and informs the *Action Performer* about the result.
8. The *Action Performer* informs the *BP Controller* about the execution of the operation.
9. *The BP Controller* continues with the process considering the result of the executed Service Task.

### 5.1 Tool Support

To support developers in the implementation of the infrastructure microservices introduced above we developed distributable Java libraries by using the framework Spring Boot. This framework provides simple annotations and configuration files in order to develop and deploy the different components of an architecture. It uses reflection mechanisms to detect specific annotations and inject the corresponding functionality.

Thus, we created two Java libraries[6] that encapsulate the functionality of the *Action Performer* and the *Context Monitor*. Each of these libraries include an annotation (`@ActionPerformer` and `@ContextMonitor`) in order to easily create these elements. In this way, developers just need to: (1) create a Spring Boot project that includes our Java libraries, and (2) create a Java class with the corresponding annotation. As representative example, the following code creates an *Action Performer* microservice. As we can see, the implementation efforts are minimum. When executing the project, the annotation is detected and the functionality of the *Action Performer* microservice is injected. The `@ContextMonitor` annotation is used in an analogous way.

```
@ActionPerformer
public class ActionPerformer {
   public static void main(String[] args) {
       SpringApplication.run(ActionPerformer.class, args);
   }
}
```

Next, we explain the functionality that each annotation injects in detail:

- `@ActionPerformer`: This annotation creates a microservice that plays the role of *Action Performer*. It injects the following functionality:
  - A module that publishes an API REST to be used by the BPMN engine in order to ask for the execution of business microservice's operations.
  - A module to inquiry the Service Registry and obtain the data required to interact with business microservices. Currently, we support Netflix's Eureka[7], which allows registering different instances of microservices. Eureka also provides a REST API in order to interact with it through the HTTP protocol, which allows the Action Performer to access microservice invocation data easily. In addition, this REST API can also be used to allow the registration of business microservices implemented in a myriad of technologies such as .Net, Java, Python, etc.
  - A module to execute REST invocations in order to interact with business microservices.
- `@ContextMonitor`: This annotation creates a microservice that plays the role of *Context Monitor*. It injects the following functionality:
  - An adapter to subscribe the Context Monitor to an event bus in order to detect context changes published by business microservices. Currently, we support the RabbitMQ[8] queue-based message broker, although others can easily be integrated. We use this broker because it provides a complete support to the MQTT protocol, which is one of the most used technologies in the IoT field. In addition, this broker provided client adapters to allow the integration of a myriad of technologies.
  - A module that transforms the context data published in the event bus (in a specific JSON structure) into a specification based on ontology concepts that are registered the OWL file that implemented the ontology. In addition, the SWRL API and Jena libraries are also included in order to apply SWRL and SPARQL rules respectively.

---

[6] The implementation of the provided tool support can be found in the following GitHub site: https://github.com/pvalderas/iot-enhanced-business-process-infrastructure

[7] https://github.com/Netflix/eureka

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

- An adapter to interact with the BPMN engine. Initially, we supported the interaction with Camunda[9], and after the evaluation experiment presented in Section 7, Bonita was also supported. We use the REST API provided by these engines to inject high-level events into the BP. Other engines can easily be supported as well through the implementation of a new adapter if they allow their management though a REST API.

# 6 Prototype evaluation

According to Völter (2016), a way of preliminarily evaluate the proposal of a new architecture is through the development of a prototype. In this section, we introduce a realization of the architectural solution presented above as a prototype involving mapping technology choices onto the solution concepts. In addition, we used this implementation to perform a preliminary evaluation in which the hypothesis that we want to validate were the following:

H1: It is feasible to execute IoT-Enhanced BPs modelled with BPMN and ontologies with the proposed architectural solution.
H2: The provided Java supporting tools automatically inject the functionality required by the architectural elements properly.
H3: IoT devices supported by different technology can be integrated in the proposed architectural solution.

## 6.1 Proof-of-concept implementation

We performed a proof-of-concept implementation[10] to support the running example presented in this paper. Figure 7 graphically illustrates the realization done of the proposed architecture. Microservices were implemented in two different technologies: Java and .Net. Three business microservices were deployed in a RaspberryPi with the Raspbian operating system. The rest of microservices were deployed in dockerized machines with two different operating systems: Windows and Linux.

Regarding the business microservices, note that this implementation includes both the business microservices required by the BPMM engine to support push interactions (i.e., Refrigerator Control System, Alarm, Information System, Truck Container Sensor, and Arm Robot), and the business microservices required to publish context changes in the event bus in order to allow the Context Monitor to inject high level events (push interactions) into the BPMN engine (i.e., Container Detector, Temperature Sensor, and Truck Detector).

The *Action Performer* and the *Context Monitor* infrastructure microservices were developed by using the Java libraries presented above and the *BP Controller* was supported by the Camunda BPMN engine. We chose Camunda because it was initially supported by the library developed to create the Context Monitor.

Regarding the Service Registry and the Event Bus, we used those solutions supported by the Java libraries presented above. In particular, Netflix's Eureka was used as Service Registry and the RabbitMQ broker was used as Event Bus.

## 6.2 Testing the prototype

Once the prototype was implemented, we evaluated its correct performance through the execution of the running example. To do so, we deployed the BPMN model in Figure 2 into the Camunda engine. Also, the

[9] https://github.com/camunda
[10] The implementation of the running example can be found in the following GitHub site: https://github.com/pvalderas/iot-enhanced-business-process-example

context ontology and the SWRL and SPARQL rules presented in Section 4.2 were deployed into the Context Monitor.

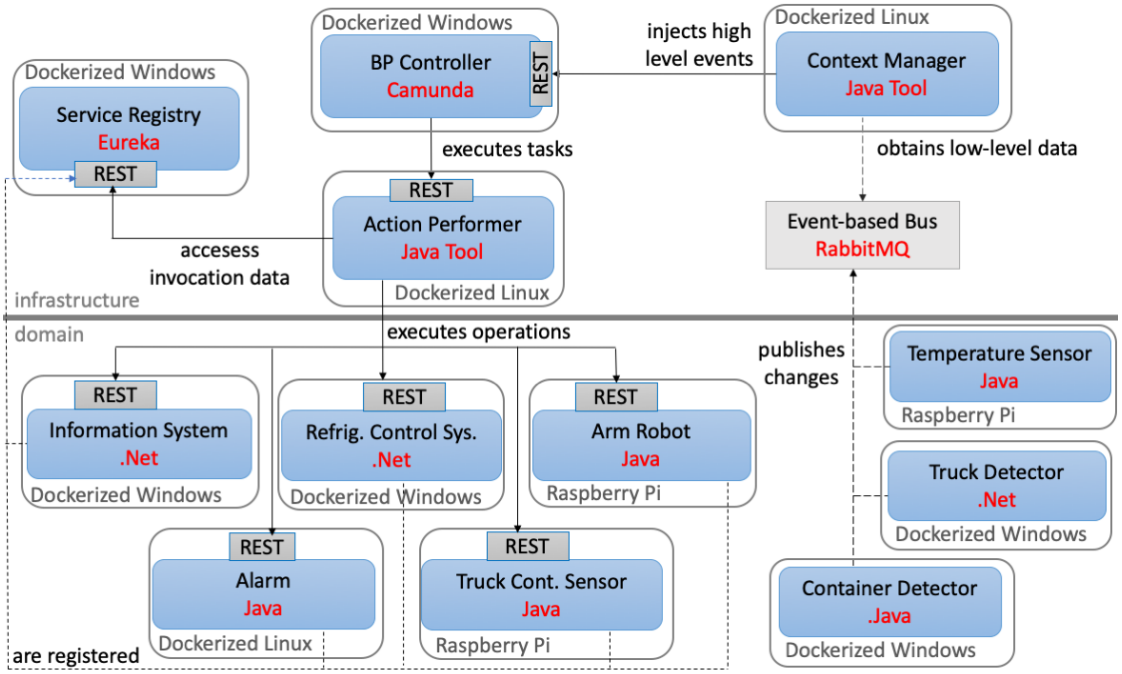


**Figure 7.** Microservice architecture implemented as a proof of concept

According to the BPMN model (see Figure 2), the business process must start when a Container is detected. In addition, a "Truck Available" event must be also injected into the BPMN engine to complete the execution. These high-level events must be injected by the Context Monitor after analysing the context data published in the Event Bus by the Container Detector and Truck Detector microservices. In a real scenario, these microservices must obtain the context data from specific sensors (e.g., Bluetooth Beacons) deployed into specific physical areas. In this testing experiment, we emulated these physical sensing actions by allowing the implemented microservices (i.e., Container Detector, Truck Detector, and Temperature Sensor) to publish context data that was manually introduced by us. Finally, note also that the BPMN model of the running example contains user tasks in which the engine pauses the execution until it is informed of the completion of the tasks. To do so, we used the REST API provided by Camunda.

Thus, in order to start the process, we make the Container Detector to publish into the Event Bus the context data that describe the presence of a Container in the reception area. Then, the Context Monitor analyses it and inject the "Container Detected" high-level event into Camunda. Afterwards, Camunda executes the BPMN model by interacting with the Action Performer in order to execute the operations of IoT devices. As the BPMN model has conditional gateways, we prepared the environment in order to execute the process different times in such a way the engine must follow a different path in each execution. In order to analyse the correct execution of the process we made each business microservice to log the execution of each operation. After the execution of the running example was completed we analysed the generated logs in order to check that operations were executed as it was defined in the BPMN model. As a representative example, Figure 8 shows the logs obtained for one of the executions.

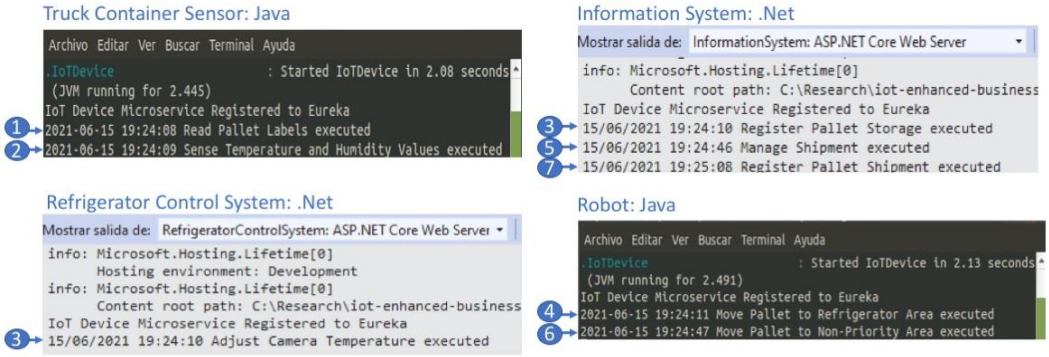


**Figure 8.** Logs obtained in one execution of the running example

According to the generated logs, we could conclude that the realization of the proposed architecture successfully executed the running example. This means that: (1) the Context Monitor correctly analysed the context data published in the Event Bus in order to inject high-level events into Camunda; (2) Camunda properly interacted with the Action Performer in order to ask for the execution of operations; and (3) the Action Performer properly interacted with the Service Registry and the business microservices in order to execute the operations asked by Camunda.

### 6.3 Replicability

The code of the running example implementation is available in a Github repository[15]. The reader can access it in order to replicate the presented experiment.

### 6.4 Further tests

In addition of the running example, we tested the proposed architecture and tool support with the execution of additional examples of IoT-Enhanced BPs. We proceeded in an analogous way to the experiment done with the running example. Note that all the infrastructure elements of the architecture (i.e., Service Registry, Event Bus, BP Controller, Context Monitor and Action Performer) could be reused from the previous implementation. Thus, in order to test new examples, we only had to: (1) define the BPMN model and deploy it into the BP Controller; (2) define the context ontology with the SWRL and SPARQL rules and deploy them into the Context Monitor; and (3) create the business microservices that manage the IoT Devices.

In particular, we tested three additional examples[11]: (1) *Smart Irrigation Management*. This example is based on the one presented by Martins et al (2019) in order to automatically control an irrigation system. It was implemented by five business microservices. Two of them implemented in Java, other two in .Net, and the last one in PHP. (2) *Ventilation System Controller*. This example is based on the one presented by Casati et al. (2012) in order to control the ventilation of a smart home. It was implemented by four business microservices. Two of them were implemented in Java, and the other two in .Net and Python. (3)*Health Care System*. This example is based on the one presented by Serral et al. (2015) in order to help elderly people when they accidentally fall at home. It was implemented by five business microservices. Two of them were implemented in Java and the other three in .Net, Python and PHP respectively.

In general, all the examples were executed successfully. Only some minor coding mistakes were detected and corrected.

[11] The implementation of these examples can be found in the following Github repository: https://github.com/pvalderas/iot-enhanced-business-process-additional-examples

### 6.5 Conclusions

According to the results obtained by the implementation and execution of different IoT-enhanced BPs, we could conclude that the feasibility of the proposed approach (hypothesis 1) is validated. The proposed architecture is able to execute IoT-enhanced BPs defined by the presented modelling approach based on BPMN models and context ontologies.

Also, the tool support presented above allowed us to successfully develop the proposed architectural elements (Action Performer and Context Monitor) with little programming efforts. In this sense, the developed Java libraries automatically inject properly the required functionality (hypothesis 2).

Finally, note that business microservices of the running example were implemented in two different technologies (Java and .Net). In the additional examples presented in Section 6.4, we implemented business microservices with other technologies such as PHP or Python. In this sense, we can conclude that the proposed architectural solution allows the integration of IoT Devices that are managed by different technologies (hypothesis 3). Business microservices only need to use the REST API of Eureka to register as available microservices, and provide their own REST API in order to allow the Action Performer execute their operations. If they need to publish context data, they only need to implement the code for interacting with the event bus (RabbitMQ in the above example). In addition, note also that the technological heterogeneity of business microservices was totally transparent for the BPMN engine (in this case, Camunda). The BPMN engine only needed to interact with the Action Performer, achieving a high level of technology independence between the BPMN engine and the IoT devices. Note that this contributes to achieve one of the improvements identified in the analysis of the state of the art (see Section 3.4): the solution proposed provides a high degree of decoupling between the created models and the underlying IoT technology.

## 7 Case study evaluation 1. Creation and deployment of BPMN models

In this section, we evaluate the proposed modelling approach from the perspective of BP modelling and deployment. In particular, the hypotheses that we want to validate are the following:

H1: The proposed BPMN model and the supporting architecture are usable to support the following characteristics of an IoT-Enhanced BP: the flow of coordinated tasks, the IoT devices that participate in the BP, and the pull and push interactions that the process must have with IoT devices.
H2: BPMN models can be defined and deployed with independence of the technology used to manage context data and implement business microservices.

To do so, we arranged a usability experiment in which participants played the role of business process engineers, which were asked to define and deploy the BPMN model presented as running example. We set up the proposed architecture and created the ontology to allow the Context Monitor to generate high-level events from low-level context data. We applied a case study-based evaluation by following the research methodology practices provided by Runeson & Höst (2009). These practices describe how to conduct and report case studies and recommend how to design and plan the case studies before performing them. Next, we introduce the experiment by describing its participants, design, execution, analysis of the results, and threats of validity.

### 7.1 Participants

A total of 15 subjects between 24 and 45 years old participated in the experiment (six female and nine male). Four participants worked on external computer science companies; three of them belonged to the PROS research centre; and the remaining eight participants were doctoral students of the Universitat Politècnica de València. All participants had some experience in the modelling of BP with BPMN but only

five of them had worked previously in an IoT project. Only three participants had expertise in microservices.

### 7.2 Design

In order to perform usability experiments, it is necessary to clarify how usability can be measured (affected variables). According to the standard ISO 9241-11 (1999), the main affected variables concerning usability requirements are: (1) effectiveness, (2) efficiency, and (3) user acceptance. To measure effectiveness and efficiency we based on Vogel-Heuser (2014). The effectiveness was measured as the grade of task completion that is obtained when comparing the result of a task with a predefined master result. The efficiency was measured as the time needed to complete a task. Inspired by Zou et al. (2007) this time was compared with the time obtained by an expert on the modelling approach when performing the same task. Regarding the user acceptance it was measured by means of a NASA-TLX questionnaire (Hart & Staveland, 1988). Thus, the instruments that were used to carry out the experiment are as follows:

- A demographic questionnaire: a set of questions to know the level of the users' experience in process modelling, BPMN, IoT, and microservices.
- Work description: the description of the two activities that the subjects should carry out: (1) using our modelling approach to define the IoT-enhanced process that support the scenario of perishable product storing; and (2) deploying and executing the created BPMN into the proposed architecture.
- A NASA-TLX questionnaire: 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 an additional open question.
- A time form: it was defined to capture the start and completion times of the proposed activities.

### 7.3 Execution

To perform the experiment, we organized a two-day workshop. In the first day, two sessions of three and four hours were arranged. In the first session, participants were asked to fill in a demographic questionnaire to capture their background and were trained in our modelling approach. Regarding the use of tools, all subjects had experience on using some BPMN editors such as Camunda, BPMN.io, or Bonita and we decided to allow them to use the BPMN editor they preferred. In the second session of the first day, participants were invited to create the BPMN model that support the scenario of perishable product storing. After this task, each participant had to fill in the NASA-TLX questionnaire. Throughout this session, we observed participants and took notes on their behaviour. Participants wrote down the starting and end times of the task.

In the second day, we arranged also two sessions of three and four hours. In the first session of this second day, participants were trained in the proposed architecture to allow them to understand how microservices are used to manage IoT devices, how they can be called through the Action Performer, and how the Context Monitor should be configured to inject high-level events into a BPMN engine. In the second session, participants performed the activities required to configure the architecture and deploy the model created in the previous session into the proposed architectural solution. In particular, the tasks that participants had to do in this session were: (1) set up the BP Controller microservice in order to use the BPMN engine they chose; (2) deploy the BPMN model that describe the IoT-enhanced BP into the BPMN engine and update the model to connect to the Action Performer in order to call IoT device microservices; and (3) configure the Context Monitor in order to interact with their BP Controller microservice. Again, participants wrote down the starting and end times of this session and completed, at the end of the session, the NASA-TLX questionnaire. We also observed participants and took notes on their behaviour throughout the entire session.

### 7.4 Analysis of the results

**Effectiveness.** Regarding the first task, we measured the effectiveness as the grade of task completion in such a way a BPMN model was totally completed if it was logically and syntactically correct. To facilitate this evaluation a master model was used as a reference point. The models created for each participant were independently evaluated by two of us in order to reduce subjectivity. Next, both corrections were analysed together, and an agreed mark was decided for each model by the two evaluators. We obtained grades between 65% and 95%, obtaining an average mark of 82.2%. Thus, we can consider that our modelling approach is effective enough to support IoT-enhanced BPs. Note that all the participants had some experience using BPMN, so this task was quite familiar for them. Most of the detected problems were related to the use of *pools* or *lanes* since some participants defined IoT devices as independent *pools* when we proposed to define IoT devices as *lanes* of a unique *pool*. Another problem that we identified raised from the use of different BPMN editors and the definition of *message flows* between the *pool* that represent the physical world and the *message start events* defined in the *lanes* that represent IoT Devices (see the message flows Container Arrives and Too Warm in Figure 2). Although this type of connection is defined in the metamodel of BPMN 2.0, editors provide different support to represent it graphically. While Camunda and BPMN.io allow defining these connections, Bonita does not. In the case of Bonita, a *message end event* was needed in the *pool* that represent the physical world to connect it with the *message start event* of IoT devices. Although this solution allowed them to apply our modelling approach with Bonita, it overloads graphically the model and provides a slightly less clear description. As further work, we plan to do a further analysis to identify how different BPMN editors support this aspect in order to be considered in our approach. Another aspect to highlight from the evaluation we did of the models created by participants is the way some of them modelled push interactions. As commented above, we proposed the use of *message start events* to represent the reception by an IoT device of a high-level event triggered from the physical world. However, some participants used *receive tasks* instead. Although this solution is semantically equivalent to ours, we think that the use of message start events simplifies the model since *receive tasks* need to be combined with a *start event* when they initiate a process.

Regarding the second task, we measured the effectiveness by comparing the log obtained when executing the solutions of participants with a master log obtained by a solution prepared by us. In this case, we graded participant's solutions with marks between 75% and 100%. As far as the setting up of the BP Controller, few problems were detected since participants had experience in the deployment and configuration of the selected engine. We detected some mistakes in the definition of the invocation of the microservices that manage IoT devices thought the REST API of the Action Performer. This task was easier for those participants that use Bonita since this editor provided a graphical wizard that guides participants in the definition of calls to an external REST APIs. The participants that used Camunda needed to implement a Java class, which produced more syntax mistakes in the invocation of the Action Performer API. Some participants needed additional help to implement a REST call from Java. Other problems were detected in the configuration of the Context Monitor to connect it with the BP Controller since it must be done through YAML files that some participants had never used before.

**Efficiency.** It was measured comparing the times obtained by participants in the performance of the two proposed tasks with the times obtained by expert users. Table 2 shows these times. We can see that the efficiency in the first task is better that the efficiency in the second task. This was an expected result. Note that Task 1 consisted in the creation of a BPMN model and participants had previous experience in this modelling language. On the contrary, Task 2 consisted in the deployment of this model into a microservice architecture that participants had never worked with before as well as the execution of the whole solution. Independently of this, we obtained an efficiency of 0.81 and 0.69 for Task 1 and Task 2 respectively, which are quite acceptable values.

**User Acceptance.** The results of the NASA-TLX questionnaire are shown in Table 3. In this questionnaire, 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); and the performance

is rated between very good (value 0) and very bad (value 100). Table 3 shows the average (Avg), the median (Med), the standard deviation (SD), the best result (Best), and the worst result (Worst).

**Table 2.** Result of the efficiency study in experiment 1. Times in minutes

| Subjects | Task 1 | Task 2 |
|---|---|---|
| **Experts 1 and 2** | 74, 67 | 109, 117 |
| **Average (Experts)** | **70.5** | **113** |
| **Participant 1-15** | 84, 92, 81, 79, 85, 98, 87, 86, 75, 94, 74, 92, 96, 99, 82 | 155, 157, 178, 152, 156, 165, 152, 164, 175, 164, 149, 158, 176, 169, 172 |
| **Average (Participants)** | **86.93** | **162.8** |
| **Efficiency** | **0.81** | **0.69** |

**Table 3.** NASA results in experiment 1

| Factors | Task 1 | | | | | Task 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Avg** | **Med** | **SD** | **Best** | **Worst** | **Avg** | **Med** | **SD** | **Best** | **Worst** |
| Mental Load | 19.67 | 15 | 10.43 | 10 | 45 | 29.67 | 30 | 13.95 | 15 | 60 |
| Physical Dem. | 3.00 | 5 | 3.26 | 0 | 10 | 7.67 | 5 | 3.72 | 5 | 15 |
| Temporal Dem. | 35.67 | 30 | 10.67 | 25 | 55 | 37.33 | 30 | 12.52 | 25 | 65 |
| Performance | 30.67 | 30 | 8.63 | 15 | 50 | 35.33 | 35 | 11.87 | 15 | 60 |
| Effort | 28.67 | 25 | 10.08 | 10 | 45 | 31.00 | 30 | 11.21 | 10 | 55 |
| Frustration | 11.00 | 10 | 8.70 | 0 | 35 | 31.00 | 25 | 14.17 | 15 | 65 |

From a general point of view, both tasks were ranked with acceptable values in the analysed factors. Task 1 obtained slightly better results than Task 2, which, as happened with the efficiency, was an expected result due to the experience of participants in the use of BPMN. The obtained values lead us to consider that participants felt comfortable enough when creating an IoT-enhanced BP model and deploying it into the proposed architecture. Regarding Task 1, little mental demand and effort was required by participants, which allows us to conclude that our approach for modelling IoT-Enhanced BPs is easy enough for business process engineers with experience in the use of BPMN. The little frustration and good performance that was indicated by participants also reinforce this consideration. However, further research is needed to analyse how business process engineers without experience in BPMN would feel when using our approach.

The values obtained in Task 2, although we think they are acceptable, lead us to think that additional support is required to facilitate business process engineers in the deployment of IoT-Enhanced BPs into a microservice architecture such as the proposed one. Note that the mental load and frustration factors were significantly higher in this task than in Task 1. By analysing the comments given by participants in the open question included in the NASA-TLX questionnaire we can conclude that the main reason was the need of managing an architecture that required the configuration of so many elements (microservices). Although we think the proposed microservices are the ones required to provide a proper level of decoupling and independence among BPs and IoT devices, we understand that additional research is required to facilitate the integration of the BP Controller with both the Context Monitor and the Action Performer, and to reinforce the separation of roles between business process engineers and experts in the IoT environment.

### 7.5 Conclusions

Based on the experiment results, we can conclude that the modelling approach based on BPMN is usable enough to face the description of the intrinsic characteristics of an IoT-enhanced BPs (Hypothesis 1): the flow of coordinated tasks, the IoT devices that participate in the BP, and the pull and push interactions that the process must have with devices. As we have deeply analysed in the above-introduced explanation, participants found our proposal intuitive to define these characteristics, which were defined by using the notions of the standard BPMN, without extending it with new concepts. Only some minor misunderstandings were detected in the definition of push interactions due to the several modelling options that BPMN provides to define event-based communications.

Note that one of the improvements introduced by our approach (see Section 3.4) was the proposal of a modelling approach for IoT-enhanced BPs that does not increase the complexity of the BPMN metamodel and is compatible with existing BPMN process engines. If we consider that participants of the experiment deployed standard BPMN models into commercial engines such as Camunda or Bonita to execute the running example, we think that we achieved the proposed improvement.

Regarding the deployment of the BPMN model into the proposed architecture, additional efforts are required to facilitate this task, since several configurations are required and participants found them a little frustrating. To improve this problem, we are working on a Java library similar to the ones that support the Context Monitor and the Action Performer in order to create a microservice that plays de role of BP Controller. This library will automatically inject a Camunda engine and configure part of the interaction of the BP Controller with the rest of architectural elements.

Finally, the second hypothesis of this experiment was focused on analysing whether or not BPMN models can be defined and deployed independently of the technology used to manage context data and implement business microservices. As we have explained along the presentation of this experiment, participants only worked with BPMN models and the BPMN engine in which these models were deployed. All participants could complete the proposed tasks without knowing either how the business microservices that execute the BPMN tasks were implemented or how context published from the physical world were managed to inject high-level events into the BPMN engine. Both, business microservices and the context ontology, were developed and managed by us. Participants only needed to know the API REST required to interact with the different architectural elements. In this sense, we can conclude that Hypothesis 2 of this experiment is validated and our proposal provides a high level of independence between the BPMN model and the technologies used to manage context and implement business microservices. Note that this aspect contributes to achieve the third improvement identified in the analysis of the state of the art, i.e., the execution of IoT-enhanced BPs with a high degree of decoupling between the models and the underlying IoT technology.

### 7.6 Threats to validity

*Conclusion validity*. It was threatened by the random heterogeneity of subjects, which was minimized with: (1) the demographic questionnaire that allowed us to evaluate the knowledge and experience of each participant beforehand; and (2) the training sessions in which all subjects participated to have a similar background in the management of ontologies and SWRL/SPARQL as well as our proposed microservice architecture.

*Construct validity*. The threat of the hypothesis guessing (people might try to figure out what the purpose and intended result of the experiment are) was minimized by hiding the goal of the experiment (i.e., which were the validation hypothesis).

*Internal validity*. This experiment was also threatened by the reliability of measures taken (e.g., the activity completion time to evaluate efficiency was measured manually), which was reduced by observing the subjects while they were performing the different tasks to guarantee their exclusive dedication in the activities and supervise the times that they wrote down. Note also that we introduced some subjectivity when grading the proposed tasks by comparing the solutions made by participants with a master one. To reduce this problem each delivered model was evaluated twice.

*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. In order to make the experimental environment more realistic participants made use of one of the most used open-source tools for the management of ontologies and faced the development of an IoT-Enhanced BP based on a real scenario (Bowman et al., 2009). However, just one case study was used in the experiment, which can threaten the generalizability of this experiment. Although we have done a prototype evaluation with several examples (see Section 6), usability experiments with additional case studies are needed. Also, many of the participants in the experiments were PhD students, which could threaten the generalization to another population, so additional experiments are needed.

# 8 Case study evaluation 2. Definition of high-level events from context data

In this section, we present an evaluation to analyse the proposed modelling approach from the perspective of context management. In particular, the two hypothesis that we want to validate are as follows:

H1: The proposed context ontology and the SWRL/SPARQL rules are usable to support the processing of context data in order to generate the high-level events that an IoT-Enhanced BP needs.
H2: High-level events can be produced with independence of both the technology used to implement business microservices and the selected BPMN engine.

To do so, we arranged an experiment in which participants populated the context ontology required to implement the running example and created the SWRL and/or SPARQL rules required to inject high-level events into the BP Controller. We set up the BP Controller with a Camunda engine and deployed the BPMN model of the running example. We also implemented the business microservices required to support the experiment. We applied a case study-based evaluation by following the same research methodology practices than in the previous experiment (Runeson & Höst, 2009).

## 8.1 Participants

A total of 11 subjects between 26 and 38 years old participated in this experiment (five female and six male). Six of the participants were doctoral students that belonged to the PROS research centre, which also participated in the previous experiment; and the remaining five participants were students of the Master's Degree in Information Management in the Universitat Politècnica de València. The doctoral students had all experience in UML conceptual modelling but had never worked with ontologies. The students of the Master's Degree had previously worked with ontologies in several subjects of the degree. None of the participants had worked with SWRL or SPARQL. All of them had experience in the Java programming language but none of them had worked with microservices.

## 8.2 Design

In order to perform this usability experiments, we evaluated the same variables as in the previous one: (1) effectiveness, (2) efficiency, and (3) user acceptance. We used the same instruments as in the previous experiment: a demographic questionnaire, a work description, a NASA-TLX questionnaire, and a time form. In this case, subjects had to carry out the following two activities in the experiment: (1) the population of the context ontology to define the IoT devices and the creation of the required SWRL or SPARQL rules to process low-level context data to generate high-level event; and (2) the creation of a Context Monitor microservice by using the developed Java library (see Section 5.1) and the deployment of the ontology with the SWRL or SPARQL rules.

## 8.3 Execution

To perform the experiment, we organized a two-day workshop. In the first day, two sessions of five and four hours were arranged. In the first session, participants were asked to fill in a demographic questionnaire to capture their background and were trained in the technologies they must use. In particular, we provided the subjects with a tutorial of both the management of OWL ontologies with Protégé and the SPARQL and SWRL languages. Note that we propose the use of Protégé[12], which is an open-source tool that is a widely used in the management of ontologies. In the second session, participants were invited to: (1) populate the context ontology with the concepts required to describe the IoT devices that play the role of sensors

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

(Temperature Sensor, Object Detector) and the sensed context data (Degree, Container, Truck, etc); and (2) define the SWRL/SPARQL rules required to generate the high-level events that we included in the BPMN model. To do so, they used Protégé. After this task, each participant filled in the NASA-TLX questionnaire. Throughout this session, we observed participants and took notes on their behaviour. They wrote down the starting and end times of the task.

In the second day, we arranged two sessions of two hours. In the first session, participants were trained in the proposed architecture to allow them to understand how microservices interact to each other, how a Context Monitor can be created by using the developed Java library, and how the context ontology and the SWRL/SPARQL rules must be deployed. In the second session, participants created a Context Monitor configured to interact with the BP Controller we had set up and deployed the context ontology and the SWRL/SPARQL rules created in the previous session. Participants wrote down the starting and end times of this session and, at the end, each participant completed again the NASA-TLX questionnaire. We also observed participants and took notes on their behaviour throughout the entire session.

### 8.4 Analysis of the results

**Effectiveness.** Regarding the first task, we considered the effectiveness of populating the context ontology and creating the SWRL/SPARQL rules. This effectiveness was measured as the grade of task completion in such a way the task was totally completed if it was logically and syntactically correct. As in the previous experiment, we used a master ontology and rules as a reference point. To reduce subjectivity, the tasks performed by participants were independently evaluated by two of us and an agreed mark was decided by the two evaluators. To grade these tasks, we analysed if the IoT devices of the running example that need to publish data context were correctly defined by using the concepts provided by the ontology. In addition, we also analysed whether the SWRL/SPARQL rules were properly defined to generate the high-level events required by the BPMN model of the running example. We obtained grades between 58% and 95%, obtaining an average mark of 72.8%. Considering that none of the participants had worked previously with SWRL and SPARQL and some of them had little experience in the use of OWL ontologies we think the obtained mark is acceptable to consider that our approach is effective enough in the management of the context required by an IoT-Enhanced BP.

The main problems that we found were as follows. First, those participants that had never worked with ontologies but had experience in conceptual modelling found a little confusing how the relationships between concepts must be defined. Note that in UML, for instance, a relationship is identified between the two specific concepts that it associates, while in an ontology, a relationship is defined by another concept (Object Property in OWL) that can be used to link a pair of concepts. Something similar happened with the OWL data properties, since they are not defined for a specific class (as happens with UML class attributes) but they can exist independently of a class, and can be associated to multiple classes. The second problem was detected in the creation of SWRL/SPARQL rules. Note that SWRL is a language that allows the creation of rules to infer new knowledge, and SPARQL is a language that allows querying an ontology to check if a specific condition is satisfied. In our approach, SWRL can be used to generate knowledge that helps to simplify SPARQL queries. However, its use is not always mandatory. We think that the possibility of using both languages provides developers with a high degree of versatility and expressiveness. However, this dichotomy in the specification of rules for generating high-level concepts was not well understood by participants, and some of them needed our help to decide on which of these two languages they needed to use. These two problems may be related to a lack of experience in using ontology-based technologies. We are currently working in a plugin for Protégé that helps developers to populate the proposed context ontology and to create the SWRL and SPARQL rules that are required to define high-level events. More details about this plugin are given in Section 8.5.

Regarding the second task, i.e., creating a Context Monitor and deploying the context ontology and the SWRL/SPARQL rules, we evaluated it by running the example and analysing if high-level events were injected into the BP Controller properly. To do so, we compared the logs generated by the Camunda engine when executing the solutions created by participants and those obtained in a solution created by us. In this

case, we graded participant's solutions with marks between 85% and 100%. In general, minor problems were detected in this task and most of the participants could create and set up the Context Monitor properly. The most significant issue was related to names that participants gave to the high-level events generated with SWRL/SPARQL rules. These names must be exactly the same as the ones defined in the BPMN model. Although we provided these names to participants, some of them defined similar ones (but not exactly the same) or introduced some typo in the names, which produced that the high-level event that was injected to the BP Controller was not the required one.

**Efficiency.** It was measured comparing the times obtained by participants in the performance of the two proposed tasks with the times obtained by expert users (cf. Table 4). We can see that the efficiency in the second task is better than the efficiency in the first task. This was an expected result since Task 2 consisted in the creation of a Context Monitor by using the Java library we provided participants with. Note that this library injects all the required functionality by using just a Java annotation and some minor configurations, and participants had previous experience in programming with Java. On the contrary, Task 1 consisted in the population of the context ontology and the creation of SWRL/SPARQL rules, which imply the use of technologies in which participants had little or no prior experience. Despite of this, we obtained an efficiency of 0.61 and 0.84 for Task 1 and Task 2 respectively, which we think are acceptable.

**Table 4.** Result of the efficiency study in experiment 2. Times in minutes

| Subjects | Task 1 | Task 2 |
|---|---|---|
| **Experts 1 and 2** | 58, 66 | 33, 38 |
| **Average (Expert)** | **62** | **35.5** |
| **Participants 1-11** | 94, 101, 110, 96, 98, 109, 102, 107, 105, 95, 98 | 38, 42, 39, 44, 41, 39, 41, 43, 44, 45, 47 |
| **Average (Participants)** | **101.36** | **42.09** |
| **Efficiency** | **0.61** | **0.84** |

**User Acceptance.** According to the NASA results (cf. Table 5), the two tasks were ranked with values that are consistent with the results obtained when analysing the effectiveness and efficiency. Although both tasks were ranked with acceptable values, participants found Task 1 more demanding in all the analysed parameters than Task 2. In fact, Task 2 of this experiment is the most well ranked task of the four tasks performed in the two usability experiments we have presented. Analysing the comments given by participants in the open question included in the NASA-TLX questionnaire we noticed that participants found easy the use of the Java library based on annotations to automatically inject functionality. On the contrary, some comments reinforce the conclusions presented above about the usage of SWRL and SPARQL to define high-level events, which resulted confusing. In addition, some participants suggested that having a wizard to create SWRL/SPARQL rules by selecting the context data generated by IoT devices would be a valuable tool to perform this task. Driven by this feedback and the proven difficulty of managing ontology-based technologies without previous experience, we decided to develop a Protégé plugin.

**Table 5.** NASA results in experiment 2

| Factors | Task 1 | | | | | Task 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **Avg** | **Med** | **SD** | **Best** | **Worst** | **Avg** | **Med** | **SD** | **Best** | **Worst** |
| Mental Load | 35,45 | 35,00 | 9,86 | 20 | 60 | 15,45 | 15,00 | 4,72 | 10 | 25 |
| Physical Dem. | 5,45 | 5,00 | 4,72 | 0 | 15 | 3,18 | 5,00 | 2,52 | 0 | 5 |
| Temporal Dem. | 40,00 | 40,00 | 11,40 | 25 | 60 | 23,64 | 25,00 | 9,51 | 10 | 35 |
| Performance | 37,73 | 30,00 | 13,67 | 25 | 65 | 25,45 | 25,00 | 5,22 | 20 | 35 |
| Effort | 28,18 | 25,00 | 11,24 | 15 | 50 | 23,18 | 25,00 | 8,15 | 10 | 35 |
| Frustration | 24,55 | 20,00 | 8,20 | 15 | 40 | 10,45 | 10,00 | 4,16 | 5 | 15 |

### 8.5 Conclusions

The results obtained in this experiment allow us to conclude that we can consider our approach usable enough to process context data to generate high-level events (Hypothesis 1). However, it would be interesting to provide additional support to facilitate the adoption of our proposal. In particular, some tool should be provided to support the population of the context ontology and the creation of SWRL/SPARQL rules. As we commented above, we are currently working on a Protégé plugin[13] that helps developers in these tasks. To do so, we have created a new Java library that provides annotations based on the SOSA ontology to facilitate the definition of semantic data in the creation of a business microservices. This semantic data can be stored in the Eureka server when business microservices are registered. Then, the Protégé plugin gets this data and provides a user interface that allows developers to select context data and to automatically define SPARQL rules (SWRL rules are still not supported).

Regarding the Java library presented in Section 5.1, we can conclude that it facilitates the creation of infrastructure microservices such as the Context Monitor. The feedback provided by participants indicates that they found the definition of annotations useful and easy to use to automatically provide Java-based microservices with the functionality required by the infrastructure elements of our architecture.

Finally, we can also accept Hypothesis 2, which focuses on the definition of high-level events with independence of both, the technology used to implement business microservices and the selected BPMN engine. In this experiment, participants focused only on working with the context ontology and the SWRL/SPRQL rules, while the BPMN model and the business microservices of the running example were independently developed by us. The only technological aspect participants needed to do to perform their tasks was the REST API provided by the BP Controller in order to configure the Context Monitor. However, they did not need to know which BPMN engine we were using, or the technology used to implement business microservices. Thus, Hypothesis 2 is validated. This contributes to achieve the third improvement identified in the analysis of the state of the art, which focused on providing a high degree of decoupling between the created models and the underlying IoT technology.

### 8.6 Threats to validity

This experiment shares the same validity threats explained in the experiment presented in Section 7. This explanation is omitted in order to not overload the paper.

## 9 Discussion

In this section, we discuss how the solution presented in this work faces the research questions stated in Section 1. The first question was related to considering important intrinsic characteristics of IoT-enhanced BPs at the modelling level without increasing the complexity of the BP modelling language. The main intrinsic characteristics of IoT-enhanced BPs that we focus on in this work were the representation of (1) the flow of coordinated activities; (2) the IoT devices that participate in the BP and the pull interactions with them; (3) the context data that need to be considered, at both low and high level, and (4) the push interactions required to inject them into the BP.

In order to face this question, we have applied the SoC principle to propose a modelling solution based on BPMN and ontologies. BPMN is used to describe IoT-enhanced BPs at a high level of abstraction, without considering low-level data or technological issues. Low-level data is defined in an ontology and technological aspects to execute BPMN models are delegated to a microservice architecture. We have not extended the BPMN metamodel to introduce new concepts. Instead, we have reused existing BPMN

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

constructors to properly describe the main semantics of IoT-enhanced BPs. In this sense, our proposal provides a solution based on BPMN's metamodel to represent, at a high level of abstraction: the process control flow, the IoT devices involved in the process, and the interaction with the physical world through pull and push interactions. Typically, BPMN is used by business engineers, which are experts on the notation, to define business processes, but also by other process stakeholders such as end customers, marketing professionals, or finance employees that just need to analyse the processes (Nysetvold & Krogstie, 2006; Harmon & Wolf, 2011; Leopold et al., 2016). Our solution provides all these process engineers and stakeholders with the possibility of defining and analysing IoT-enhanced BPs without the need of learning a new notation or new concepts that are not included in the BPMN standard. In addition, this BPMN solution focuses on high level concepts of an IoT-enhanced BP and does not include complex definitions related to the capture and processing of low-level context data. This aspect was evaluated through an experiment based on a case study in which we played the role of experts on the physical environment and participants with little expertise on IoT could easily define an IoT-enhanced BP.

The capture and processing of low-level context data is, however, a key pillar to properly execute an IoT-enhanced BP. Our solution considers this aspect at modelling level through an ontology. As commented above, ontologies are one the most used solutions to model context. As we have shown in the two experiments presented in Section 7 and 8, modelling context in a separated ontology allows us to provide a solution that facilitates de separation of development responsibilities. While experts on the context data produced by the physical environment can focus on defining how it must be processed to generate the high-level events required by the business process, business engineers can focus on defining the BPMN model. In addition, this helps to face the challenge of not growing the complexity of BPMN. However, we have also checked that people with little experience in the use of ontology-based technologies may require additional tool support in order to adopt our approach. To improve this problem a Protégé plugin is under development.

Regarding the second research question that we stated, it focused on how IoT-enhanced business processes that are represented in BP models can be executed independently from technology.

On the one hand, since our solution does not extend BPMN's metamodel, IoT-enhanced BPs can be executed using any existing BPMN engine. For instance, in the case-study evaluation presented in Section 6, we implemented an IoT-enhanced BP by using two different BPMN engines: Camunda and Bonita. On the other hand, it is true that only a BPMN engine is not enough to properly execute an IoT-enhanced BP modelled with our approach. We need changes in the physical world to be injected into the process. And we also need to manage the interaction of the process with IoT devices from a technological point of view. In the related work section, we have analysed some solutions that either extend BPMN engines to achieve this goal or complement these engines or the IoT devices with software components that make these solutions technology dependent. In our solution, we propose a microservice architecture that provides a high degree of independence among architectural elements, which interact among them through light communication solutions such as HTTP REST connections or event-based messages. This can be checked in the prototype evaluation presented in Section 5, in which several IoT-enhanced BPs are supported through the implementation of business microservices developed in different technologies such as Java, .Net, PHP or Python. Thus, we can conclude that our solution satisfies the requirement of executing an IoT-enhanced BP independently from technology.

Finally, this architecture can facilitate further maintenance and evolution of any architectural element. Note that IoT devices are controlled by a dedicated microservice that can be implemented in any technology or operating system. If an IoT device needs to be changed, we just need to update the corresponding microservice maintaining its REST API and the interaction with the event bus. The rest of architectural elements are not affected. In the same way, if we need to change the BPMN engine deployed in the BP Controller we just need to configure the new engine to interact with the REST API of the Action Performer, and update the Context Monitor correspondingly to inject the high-level events. IoT devices and the other architectural elements are not affected by the change of the BPMN engine. However, the maintenance and evolution issues require a more precise evaluation that will be faced as further work.

## 10 Conclusions and further work

In this work, we have presented a solution that applies the SoC principle to model IoT-enhanced BPs. This solution proposes (1) a modelling approach that combines standard BPMN with ontologies, and (2) a microservice architecture to execute IoT-enhanced BPs defined by this approach.

The contributions of our work are both theoretically and practical. From a theoretical point of view, we have identified some intrinsic characteristics of this type of processes through the study of the state of the art, and have proposed modelling guidelines based on the state of the art to use the standard elements of BPMN together with ontologies in order to represent these characteristics. In addition, we have designed a microservice architecture that supports the execution of this type of processes in such a way a great level of technology independence is achieved. We have defined the elements that constitute this architecture as well as the interaction that they must have.

From a practical point of view, we have provided an implementation of such architecture and published it in a Github repository. We have also presented some insights through the implementation of several prototypes and two case study evaluations, which revealing that the proposed use of the BPMN notation together with context ontologies can be a valuable mechanism to define and understand an IoT-enhanced business process. We have also concluded that the SoC design principle proposed through the use of BPMN and ontologies can facilitate the collaboration of different professionals such as business engineers and experts on IoT technologies in the creation of an IoT-enhanced business process.

As future work, further evaluation experiments should be considered. One the one hand, additional case studies must be used to evaluate our approach in order to validate its generalizability. On the other hand, a comparison with other approaches from a pragmatic way would also be desirable. The most appropriate approaches to be used in this comparison are those that support the modelling of IoT-Enhanced BPs and their execution (e.g., Wehlitz et al. 2017; Domingos & Martins, 2017). We would require to compare the usability and efficiency of these approaches to ours with respect to: (1) the modelling of the intrinsic characteristics of IoT-Enhanced BPs considered in this work; (2) the execution of IoT-Enhanced BPs through the use of existing BPMN engines; and (3) the execution support for IoT-Enhanced BPs decoupled from the underlying IoT technology. To do so, we can use the guidelines presented by Kitchenham et al. (1995) in a similar way as we did in previous works such as Valderas et al. (2020) in which we compared the efficiency of a new modelling approach concerning an ad-hoc solution.

In addition, we plan to enrich our solution with goal-oriented capabilities. In this way, instead of specifying the tasks of IoT-enhanced BPs explicitly, business engineers would just need to state the goals that a process must satisfy. Then, business microservices can be semantically annotated in order to provide a mechanism to select those IoT devices that better can achieve the defined goals. Further enhancements could include risk management and risk mitigation approaches (Conforti et al., 2011), identifying risks in executing IoT-enhanced BPs and simulating them at design time.

## Acknowledgment

This work is part of the R&D&I project PID2020-114480RB-I00 funded by MCIN/AEI/ 10.13039/501100011033

## References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context-awareness. Int. symposium on handheld and ubiquitous computing (pp. 304-307). Springer, Berlin, Heidelberg.

Appel, S., Kleber, P., Frischbier, S., Freudenreich, T., & Buchmann, A. (2014). Modelling and execution of event stream processing in business processes. Information Systems, 46, 140-156.

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing, 2(4), 263-277.

Baresi, L., Meroni, G., & Plebani, P. (2016). A GSM-based approach for monitoring cross-organization business processes using smart objects. In International Conference on Business Process Management (pp. 389-400). Springer, Cham.

Bermúdez-Edo, M., Elsaleh, T., Barnaghi, P.M., Taylor, K.: Iot-lite: a lightweight semantic model for the internet of things and its use with dynamic semantics. Personal and Ubiquitous Computing 21 (2017) 475-487

Beverungen, D., Buijs, J.C.A.M., Becker, J. et al. Seven Paradoxes of Business Process Management in a Hyper-Connected World. Bus Inf Syst Eng (2020). https://doi.org/10.1007/s12599-020-00646-z

Bowman, P., Ng, J., Harrison, M., Lopez, T.S., Illic, A. (2009) Sensor based condition monitoring

BPMN. (2011). Business Process Model and Notation (BPMN). Version 2.0. Object Management Group. URL: https://www.omg.org/spec/BPMN/2.0/PDF/ Last time accessed: April 2020

Casati, F., Daniel, F., Dantchev, G., Eriksson, J., Finne, N., Karnouskos, S., ... & Voigt, T. (2012, June). Towards business processes orchestrating the physical enterprise with wireless sensor networks. In 2012 34th International Conference on Software Engineering (ICSE) (pp. 1357-1360). IEEE.

Caracaş, A., & Kramp, T. (2011). On the expressiveness of BPMN for modelling wireless sensor networks applications. In International Workshop on Business Process Modelling Notation (pp. 16-30). Springer, Berlin, Heidelberg.

Chen, H., Finin, T., & Joshi, A. (2003). An ontology for context-aware pervasive computing environments. The knowledge engineering review, 18(3), 197-207.

Cheng, Y., Zhao, S., Cheng, B., Chen, X., & Chen, J. (2019). Modelling and deploying IoT-aware business process applications in sensor networks. Sensors, 19(1), 111.

Chiu, H. H., & Wang, M. S. (2015). Extending event elements of business process model for internet of things. In 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (pp. 783-788). IEEE.

Conforti, R., Fortino, G., La Rosa, M., & Ter Hofstede, A. H. (2011). History-aware, real-time risk detection in business processes. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 100-118). Springer, Berlin, Heidelberg.

Dar, K., Taherkordi, A., Baraki, H., Eliassen, F., & Geihs, K. (2015). A resource-oriented integration architecture for the Internet of Things: A business process perspective. Pervasive and Mobile Computing, 20, 145-159.

Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7.

Dimitrov, M., Simov, A., Stein, S., Konstantinov, M.: A BPMO based semantic business process modelling environment. In Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM 2007, Innsbruck, Austria, June 7, 2007. Volume 251 of CEUR Workshop Proceedings., CEUR-WS.org (2007)

Domingos, D., & Martins, F. (2017). Using BPMN to model Internet of Things behavior within business process. *International Journal of Information Systems and Project Management*, *5*(4), 39-51.

Dörndorfer, J., & Seel, C. (2018). A Framework to Model and Implement Mobile Context-Aware Business Applications. Modellierung 2018.

Fowler, M. & Lewis, J. (2014). Microservices. ThoughtWorks.

Fowler, M. (2015). Microservices trade-offs. URL: http://martinfowler.com/articles/ microservice-trade-offs.html Last time accessed: April 2020

Friedow, C., Völker, M., & Hewelt, M. (2018). Integrating IoT devices into business processes. In International Conference on Advanced Information Systems Engineering (pp. 265-277). Springer, Cham.

Gao, F., Zaremba, M., Bhiri, S., & Derguerch, W. (2011). Extending bpmn 2.0 with sensor and smart device business functions. In 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (pp. 297-302). IEEE.

Graja, I., Kallel, S., Guermouche, N., & Kacem, A. H. (2016). BPMN4CPS: A BPMN extension for modelling cyber-physical systems. In 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 152-157). IEEE.

Grefen, P., Ludwig, H., Tata, S., Dijkman, R., Baracaldo, N., Wilbik, A., & D'hondt, T. (2018). Complex collaborative physical process management: a position on the trinity of BPM, IoT and DA. In Working Conference on Virtual Enterprises (pp. 244-253). Springer, Cham.

Harmon, P., & Wolf, C. (2011). Business process modelling survey. Business process trends, 36(1), 1-36.

Hart, S.G., Staveland, L.E. (1988). Development of NASA-TLX (TaskLoadIndex): results of empirical and theoretical research. AdvPsychol. 52, 139–183.

ISO. (1999). International Organization for Standardization. Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs)-Part 11: Guidance on Usability, EN ISO 9241-11:1998. Beuth, Berlin.

Jalali, S., & Wohlin, C. (2012). Systematic literature studies: database searches vs. backward snowballing. In Proceedings of the 2012 ACM-IEEE international symposium on empirical software engineering and measurement (pp. 29-38). IEEE.

Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin et al. (2020). "The Internet-of-Things meets Business Process Management. A manifesto", IEEE SMC Magazine, https://doi.org/10.1109/MSMC.2020.3003135

Kitchenham, B., Pickard, L. and Pfleeger, S. L. (1995). Case studies for method and tool evaluation, Software, IEEE, vol. 12, no. 4, pp. 52–62, 1995.

Knoll, D., Waldmann, J., & Reinhart, G. (2019). Developing an internal logistics ontology for process mining. Procedia CIRP, 79, 427-432.

Leopold, H., Mendling, J., Günther, O. (2016). Learning from Quality Issues of BPMN Models from Industry. IEEE Software 33(4): 26-33

Mandal, S., Hewelt, M., & Weske, M. (2017). A framework for integrating real-world events and business processes in an IoT environment. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 194-212). Springer, Cham.

Martins, F., Domingos, D., & Vitoriano, D. (2019). Automatic Decomposition of IoT Aware Business Processes with Data and Control Flow Distribution. In ICEIS (2) (pp. 516-524).

Melcher, J., Mendling, J., Reijers, H. A., & Seese, D. (2009). On measuring the understandability of process models. In International Conference on Business Process Management (pp. 465-476). Springer, Berlin, Heidelberg.

Meyer, S., Ruppen, A., & Magerkurth, C. (2013). Internet of things-aware process modelling: integrating iot devices as business process resources. In International conference on advanced information systems engineering (pp. 84-98). Springer, Berlin, Heidelberg.

Mottola, L., Picco, G. P., Opperman, F. J., Eriksson, J., Finne, N., Fuchs, H., ... & Römer, K. (2017). makeSense: Simplifying the Integration of Wireless Sensor Networks into Business Processes. IEEE Transactions on Software Engineering.

Nysetvold, A. G., & Krogstie, J. (2006). Assessing business process modelling languages using a generic quality framework. In Advanced Topics in Database Research, Volume 5 (pp. 79-93). IGI Global

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. IEEE communications surveys & tutorials, 16(1), 414-454.

Petrasch, R., & Hentschke, R. (2016). Process modelling for Industry 4.0 applications: Towards an Industry 4.0 process modelling language and method. In 2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE) (pp. 1-5). IEEE.

Rosemann, M., & Recker, J. C. (2006). Context-aware process design: Exploring the extrinsic drivers for process flexibility. In The 18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium (pp. 149-158). Namur University Press.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical software engineering, 14(2), 131–164

Schönig, S., Ackermann, L., Jablonski, S., & Ermer, A. (2018). An integrated architecture for iot-aware business process execution. In Enterprise, Business-Process and Information Systems Modelling (pp. 19-34). Springer, Cham.

Serral, E., Smedt, J.D., Snoeck, M., Vanthienen, J. (2015). Context-adaptive petri nets: Supporting adaptation for the execution context. Expert Systems with Applications 42 9307 - 9317

Sperner, K., Meyer, S., & Magerkurth, C. (2011). Introducing entity-based concepts to business process modelling. In International Workshop on Business Process Modelling Notation (pp. 166-171). Springer, Berlin, Heidelberg.

Suri, K., Gaaloul, W., Cuccuru, A., & Gerard, S. (2017). Semantic framework for internet of things-aware business process development. In 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 214-219). IEEE.

Torres, V., Serral, E., Valderas, P., Pelechano, V., Grefen, P. (2020). Modeling of IoT devices in Business Processes: A Systematic Mapping Study. In 22nd IEEE Conference on Business Informatics (CBI 2020) (pp. 221-230).

Taylor, K., & Leidinger, L. (2011, May). Ontology-driven complex event processing in heterogeneous sensor networks. In Extended Semantic Web Conference (pp. 285-299). Springer, Berlin, Heidelberg.

Valderas, P., Torres, V., & Pelechano, V. (2020). A microservice composition approach based on the choreography of BPMN fragments. Information and Software Technology, 127, 106370.

Valero, C. & Ruiz-Altisent, M. (2000). Design Guidelines for a Quality Assessment System of Fresh Fruits in Fruit Centers and Hypermarkets. Agricultural Engineering International: The CIGR e-journal. 2

Völter, M. (2006). Software architecture: A pattern language for building sustainable software architectures. EuroPLoP 2006 - 11th European Conference on Pattern Languages of Programs. Mar, pp. 31–66.

Vogel-Heuser, B. (2014). Usability experiments to evaluate UML/SysML-based model driven software engineering notations for logic control in manufacturing automation. Journal of Software Engineering and Applications, 7(11), 943.

Wehlitz, R., Rößner, I., & Franczyk, B. (2017). Integrating smart devices as business process resources–concept and software prototype. In International Conference on Service-Oriented Computing (pp. 252-257). Springer, Cham.

Weske, M. (2012). Business Process Management: Concepts, Languages, Architectures, Springer, New York.

Ye, J., Coyle, L., Dobson, S., & Nixon, P. (2007). Ontology-based models in pervasive computing systems. The Knowledge Engineering Review, 22(4), 315-347.

Yousfi, A., Batoulis, K., & Weske, M. (2019). Achieving business process improvement via ubiquitous decision-aware business processes. ACM Transactions on Internet Technology (TOIT), 19(1), 1-19.

Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. Information and Software Technology, 53(6), 625-637.

Zou, Y., Zhang, Q. & Zhao, X. (2007). Improving the usability of e-commerce applications using business processes. IEEE Transaction Software Engineering 33(8) 37–855.

Zugal, S., Pinggera, J., & Weber, B. (2011). Assessing process models with cognitive psychology. Enterprise modelling and information systems architectures (EMISA).

Zugal, S., Pinggera, J., Weber, B., Mendling, J., & Reijers, H. A. (2011, October). Assessing the impact of hierarchy on model understandability–a cognitive perspective. In International conference on model driven engineering languages and systems (pp. 123-133). Springer, Berlin, Heidelberg.