Enabling Digital Twin Prototypes of IoT Systems through BPMN-Driven Modeling and Simulation

Massimo Callisto De Donato[1,*,†], Flavio Corradini[1,†], Niccolò Francioni[1,*,†], Fabrizio Fornari[1,*,†], Barbara Re[1,†] and Damiano Serpetta[1,†]

Abstract

Digital Twin Prototypes can support the early simulation and validation of Internet of Things (IoT) systems before their physical deployment, especially in scenarios that are difficult, costly, or unsafe to reproduce in reality. However, early-stage design and validation remain challenging, especially for IoT systems whose behavior emerges from distributed and event-driven interactions among devices, events, and software components.

This paper proposes a BPMN-based design and validation approach for Digital Twin Prototypes of IoT systems. The approach supports the identification of relevant scenario entities, the specification of their behavior as executable BPMN models through an explicit IoT-to-BPMN mapping, and the simulation-based analysis of their interactions. To support the execution and validation of these models, a corresponding Digital Twin architecture is introduced for coordinating IoT-aware process execution in a distributed setting. A prototype implementation of the core execution and coordination mechanisms is applied to an earthquake emergency scenario, providing preliminary evidence of the feasibility of the approach for the early behavioral validation of IoT systems.

Keywords

Digital Twin Prototype, Internet of Things, BPMN, Behavioral Modeling, Simulation and Validation

1. Introduction

The concept of Digital Twin has emerged as a relevant paradigm for representing and managing complex systems in the digital domain [1]. A Digital Twin can be described as a virtual representation of a physical asset or system that maintains a continuous connection with its physical counterpart through data exchange [2]. By integrating telemetry data, contextual information, and analytical capabilities, Digital Twins support monitoring, analysis, and optimization of physical systems across their lifecycle [3]. Their adoption has been strongly connected with the diffusion of Internet of Things (IoT) technologies, which enable the collection of data from distributed sensors and devices. In this context, Digital Twins are increasingly used to represent IoT-enabled environments, industrial systems, and smart infrastructures [4].

A relevant application of the Digital Twin paradigm in IoT systems concerns the possibility of testing and validating device behavior before the physical system is fully deployed, or under conditions that are difficult, costly, or unsafe to reproduce in reality. This is the case, for example, of emergency scenarios in which devices may not yet be installed, may operate in environments that are expensive to replicate, or may be exposed to rare and extreme events that are difficult to reproduce consistently for testing purposes [5].

To address these situations, the notion of *Digital Twin Prototype* (DTP) can be adopted [1]. Unlike a Digital Shadow, which assumes an existing physical counterpart and automatic synchronization from the physical to the digital representation, a DTP refers to a system whose physical counterpart is not yet fully deployed, but whose structure and expected behavior can already be modeled and analyzed. By

CBI & EDOC 2026, September 15–18, 2026, Enschede, NL

*Corresponding author.

†These authors contributed equally.

massimo.callisto@unicam.it (M. Callisto De Donato); flavio.corradini@unicam.it (F. Corradini); niccolo.francioni@unicam.it (N. Francioni); fabrizio.fornari@unicam.it (F. Fornari); barbara.re@unicam.it (B. Re); damiano.serpetta@studenti.unicam.it (D. Serpetta)

0000-0001-6745-4785 (M. Callisto De Donato); 0000-0001-6767-2184 (F. Corradini); 0009-0007-3031-7593 (N. Francioni); 0000-0002-3620-1723 (F. Fornari); 0000-0001-5374-2364 (B. Re); 0009-0008-9635-014X (D. Serpetta)

providing a virtual environment in which devices and their interactions can be instantiated, configured, and simulated before deployment, a DTP supports early reasoning on system behavior, scenario exploration, and pre-deployment validation. While in this work the DTP is used at design time, the resulting behavioral models may later be refined and connected to the deployed IoT system rather than being treated as throw-away artifacts. This perspective was shown to be useful in our previous work, where it supported the anticipation of simulation and validation activities for an IoT emergency scenario [6].

The development of a DTP in the early stages of an IoT system often involves stakeholders with heterogeneous backgrounds, including system engineers, IoT developers, and domain experts familiar with the target operational context. Since the actual system may not yet exist and the devices themselves may still be unavailable, reasoning directly in terms of low-level implementation details is often impractical. A higher level of abstraction is therefore needed to describe device behavior and interactions in a way that is both precise and understandable across stakeholders. In this context, Business Process Model and Notation (BPMN) represents a suitable modeling notation, as it enables IoT device behavior and interaction logic to be represented through process models that can support simulation and validation activities, also in the Digital Twin context [7, 8].

However, defining such an abstraction for IoT DTPs is not straightforward. IoT scenarios are typically composed of distributed entities that interact through events and messages, while their behavior depends on sensor observations, device states, communication mechanisms, and contextual conditions. The integration of IoT and Business Process Management requires mechanisms to bridge the gap between raw sensor data, device-level events, and higher-level process knowledge [9]. Similarly, the Digital Twin of Business Processes perspective emphasizes the need for explicit, high-fidelity, and executable models that can support simulation, monitoring, and analysis of process behavior [10]. These challenges are particularly relevant when IoT component behavior needs to be understood, discussed, and validated before the physical deployment is available.

Motivated by these challenges, this paper proposes a design and validation approach to support the development of DTPs for IoT systems. The approach covers the identification of relevant scenario entities and interactions, their behavioral specification through executable BPMN models based on an explicit IoT-to-BPMN mapping, and the coordinated execution and analysis of the resulting behavioral models. To support these activities, the paper introduces a reference architecture for IoT DTPs comprising a Simulation Manager, a process execution engine, an event bus, a middleware component for message correlation, and an IoT platform. A prototype implementation of the core execution and coordination mechanisms has been developed and applied to an emergency scenario derived from [5]. The application provides preliminary evidence that the selected IoT behaviors can be represented as separate executable models and coordinated through exchanged events and messages, supporting early behavioral validation before the physical deployment is available.

The remainder of this paper is organized as follows. Section 2 presents related work on the integration of Business Process Management and IoT. Section 3 introduces the proposed design and validation approach for DTPs of IoT systems. Section 4 presents the reference architecture and the corresponding prototype implementation. Section 5 reports the application of the approach to the SAFE emergency scenario. Section 6 discusses the results and limitations of the proposed solution. Finally, Section 7 concludes the paper and outlines future work.

2. Related Work

Digital Twins have been investigated as a means to support testing, simulation, and validation activities for IoT and cyber-physical systems. This is particularly relevant when physical devices are not yet available, when the deployment environment is difficult to reproduce, or when the target scenario involves conditions that are costly, rare, or unsafe to recreate.

Several works have investigated the integration of Business Process Management and IoT, where IoT data and device capabilities are used to increase the fidelity of process models and process executions

with respect to the physical environment. In [11], the authors provide a systematic mapping study on the modeling of IoT devices in business processes, highlighting the role of BPMN in the IoT-aware business process literature. In [12], the authors propose a BPMN-based approach for modeling and executing IoT-enhanced business processes. The solution reuses standard BPMN constructs to represent IoT devices, device actions, and interactions with the physical world, while context data are modeled separately through an ontology. In [13], the authors introduce FloBP, a model-driven approach for modeling, developing, and executing IoT-enhanced business processes. The approach combines feature models to represent IoT domain knowledge with BPMN models for specifying business process logic involving IoT devices. In [14], the authors investigate approaches for IoT-enhanced predictive process monitoring, focusing on the integration of IoT time-series data with traditional process event logs.

Previous research on IoT-aware business processes has already investigated the modeling of IoT device behavior within the Process of Things perspective, in which business process modeling notations are used to represent the internal behavior of IoT devices and their cooperation [15]. This work is aligned with this perspective, but considers such behavioral models in the context of IoT DTPs, where multiple executable BPMN models represent the behavior of scenario entities and can be executed and coordinated before the complete physical system is available.

Table 1
Comparison of related works with respect to the focus of this work.

Ref.	Main focus	Modeling perspective	Role of IoT behavior
[11]	IoT-aware business processes	Mapping study of IoT elements in process models	Reviews IoT device representations in process models
[12]	IoT-enhanced process execution	Process-centric BPMN and context ontology	Device actions and physical interactions within BPs
[13]	Model-driven IoT-enhanced BPs	BPMN and feature models	IoT capabilities involved in BP logic
[14]	Predictive process monitoring	Event logs and IoT time-series data	IoT data support process prediction
[16]	Distributed IoT system analysis	Domain-specific languages	Behavior modeled for analysis and anomaly resolution
[17]	IoT testing and simulation	DT-based testing environment	IoT system behavior tested against simulated conditions
[18]	Scalable IoT validation	High-fidelity simulation	Behavior evaluated through DT-driven testing
[19]	Security validation	Device DT modeling and simulation	Device behavior considered for security testing
This work	**Pre-deployment IoT DTP analysis**	**Executable BPMN behavioral models**	**Internal component behavior explicitly modeled and coordinated**

Beyond IoT-enhanced business processes, several works have investigated the use of Digital Twins to support testing and validation activities for IoT systems. In [16], the authors introduce a method for developing Digital Twins of large-scale distributed IoT systems, addressing verification and validation challenges in operational settings. Their approach relies on domain-specific languages to model and analyze system behavior and to support anomaly resolution. In [17], the authors present a Digital Twin-based tool for testing and simulating IoT environments and predicting errors in real IoT systems. The tool supports functional and non-functional testing by connecting the physical system with a software version deployed in the Digital Twin. In [18], the authors investigate a Digital Twin-driven testing strategy as a scalable simulation-based method for IoT system validation, focusing on test coverage, scalability, and reliability. Their methodology relies on high-fidelity simulations to improve test coverage and support the identification of hidden faults. In [19], the authors propose a modeling and simulation approach to support the integration, verification, and functional validation of security

mechanisms in devices. Their approach uses Digital Twins of devices to support integration and testing activities.

Overall, the reviewed works address complementary aspects of IoT process integration and Digital Twin-based testing, as summarized in Table 1. IoT-enhanced BPM approaches mainly focus on incorporating devices, data, and physical interactions into business process execution or monitoring, while Digital Twin-based approaches primarily target testing and validation at the system or infrastructure level. This work brings these perspectives together by using executable and interacting BPMN models as the behavioral layer of an IoT DTP for pre-deployment analysis.

Accordingly, the proposed approach addresses the analysis of scenario entities and interactions, their behavioral specification through standard executable BPMN models based on an explicit IoT-to-BPMN mapping, and their execution and coordination within a supporting architecture. The objective is to make IoT behavior explicit, executable, and observable, supporting scenario-based simulation and preliminary behavioral validation before deployment.

3. Proposed Approach

This section describes the proposed approach for designing and validating IoT DTPs through executable BPMN behavioral models.

3.1. Design and Validation Process

The proposed approach is organized as an iterative design and validation process, depicted in Fig. 1. The process targets IoT scenarios that are still under design, where IoT components and related digital infrastructures may not yet be available. In such cases, designers still need to reason about the expected behavior of the system, the entities involved, and the interactions that may occur among them in order to realize a specific use case.

The use of BPMN facilitates the involvement of domain experts in co-design activities. Domain experts can contribute to the definition of the scenario by providing indications about how the system is expected to work, which events are relevant, and how the involved entities should react under specific conditions. These inputs can then guide the definition and refinement of the corresponding behavioral models.

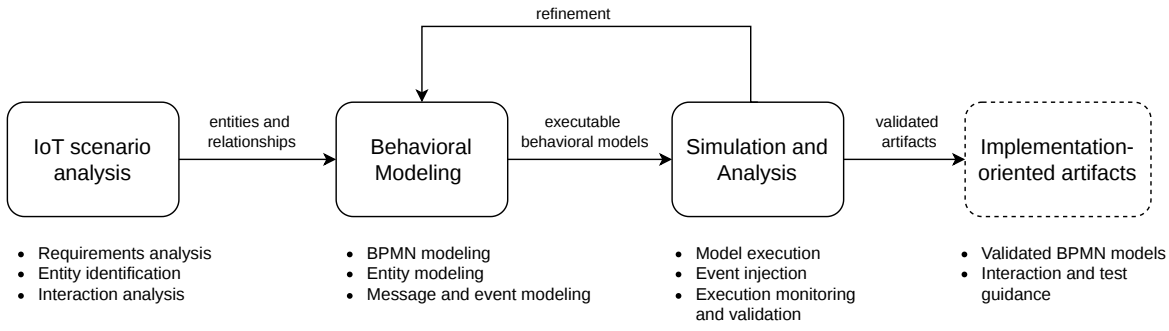

Figure 1: Design and validation process of the proposed approach.

As shown in Fig. 1, the process starts with the analysis of the IoT scenario, where requirements, relevant entities, and interactions are identified. The resulting scenario-level description provides the basis for the behavioral modeling phase, in which BPMN is used to define executable models of the entities and their event-driven interactions.

The BPMN models are then executed during the simulation and analysis phase. In this phase, controlled events can be injected, the execution can be monitored, and the observed behavior can be compared with the expected scenario logic. When inconsistencies, missing interactions, or unexpected behaviors are identified, the models are refined and executed again.

Once the modeled behavior is considered consistent with the expected logic, the process leads to the consolidation of implementation-oriented artifacts. These artifacts include BPMN models refined through simulation-based analysis, explicit interaction logic, and guidance for subsequent implementation, validation, and testing activities.

3.2. IoT Scenario Analysis

At the beginning of the design and validation process, the objective is to identify the relevant components involved in the IoT system to be represented within the DTP, their roles, and the interactions among them. This analysis identifies the elements that are relevant for behavioral modeling and simulation before moving to BPMN modeling.

The analysis is carried out by IoT experts and system designers, with the support of domain experts and stakeholders who can provide knowledge about the operational context. The activity can rely on different sources, such as project deliverables, technical specifications, requirements documents, descriptions of the target environment, previous prototypes, stakeholder discussions, and deployment constraints. These sources are used to extract the behavioral aspects that are relevant for the scenario, distinguishing them from purely structural, physical, or low-level implementation details.

Based on the available information, IoT experts, system designers, and stakeholders identify the relevant entities to include in the scenario, such as IoT devices, gateways, edge components, software services, environmental elements, and external actors. Each entity is analyzed according to the role it plays in the scenario. For instance, an IoT device may sense environmental conditions and communicate data, a gateway may collect messages and propagate commands, while an environmental element may generate events that affect the behavior of other entities.

The analysis also aims to identify the interactions among the identified entities, including exchanged messages, commands, events, and environmental signals. Interaction analysis also considers scenario constraints, such as communication timeouts, acknowledgment messages, and conditions that affect the control flow. For each interaction, the analysis should clarify the involved entities, the triggering condition, and the expected effect on the receiving entity. This is particularly relevant in IoT scenarios, where components are often loosely coupled and interact asynchronously.

The output of the IoT scenario analysis is a scenario-level description that provides the basis for modeling, executing, and analyzing the expected behavior of the IoT system.

3.3. Behavioral Modeling

The behavioral modeling phase transforms the entities and interactions identified during the IoT scenario analysis into explicit behavioral specifications. The corresponding BPMN models are manually derived from this information and represent the process-like behavior of the identified entities. The objective is not to model a traditional organizational business process, but to capture the event-driven behavior of IoT devices, edge gateways, and supporting components that realize the DTP of the IoT scenario.

In this perspective, an IoT scenario is modeled as a set of interacting BPMN behavioral models, where each model describes the logic of a single entity or logical component. Rather than representing several IoT entities within the same process model, this decomposition reflects the distributed and asynchronous nature of IoT systems [9], preserves explicit component boundaries, and makes interactions through events and messages explicit. The overall behavior of the scenario emerges from the interaction among these models through exchanged events and messages.

Table 2 reports the conceptual mapping adopted between recurrent IoT elements and BPMN constructs. In particular, IoT devices and supporting components can be represented as BPMN processes, pools, or lanes, depending on the required level of detail. Events generated by sensors, the environment, or other components are represented through message or signal events. Periodic behaviors, such as keep-alive messages or other recurring device operations, are modeled with timer events. The mapping targets behaviorally relevant events and actions rather than low-level sensor sampling. Internal operations, including data processing, state updates, communication activities, or command execution,

are represented as tasks. Decision points based on sensed data, received commands, or internal variables are modeled through gateways.

Table 2
Conceptual mapping between IoT elements and BPMN constructs.

IoT element	BPMN construct	Purpose
Device/component	Pool, lane, process model	Represent an interacting entity.
Sensor reading or external event	Message/signal event	Trigger behavior from data or environment.
Periodic behavior	Timer event	Trigger periodic behavioral actions
Local computation/action	Task/service task	Represent internal operations.
Condition or mode check	Exclusive gateway	Model branching decisions.
Alternative awaited events	Event-based gateway	React to the first occurring event.
Message exchange	Message event/flow	Model interactions among entities.
Operating mode	Variable, subprocess, or model variant	Capture behavior-dependent states.
Timeout	Timer or boundary event	Model missing responses or delays.

This decomposition also introduces a trade-off. As scenario complexity increases, the number of behavioral models and their interactions may also grow, requiring additional effort for their definition, maintenance, and coordination.

The obtained BPMN models provide the input for the simulation and analysis phase, where they are executed to simulate the scenario and assess the functional consistency of the modeled behavior.

3.4. Simulation and Analysis

After the behavioral modeling phase, the relevant entities are represented as executable BPMN artifacts. In the simulation and analysis phase, these artifacts can be deployed in an execution environment and instantiated according to the IoT scenario to be simulated. The objective is to observe how the modeled entities behave when relevant events occur and to assess whether the simulated behavior is aligned with the expectations defined during scenario analysis and behavioral modeling.

The execution of the models is event-driven. Events may be generated by simulated components, injected by users during controlled simulation sessions, received from other behavioral models, or, in later stages, collected from an IoT platform connected to real or simulated devices. These events can activate process instances, trigger intermediate events, update process variables, and cause message exchanges among the modeled entities.

Since an IoT scenario may involve several interacting models, the execution phase requires coordination among model instances. For example, a device model may send data to a gateway model, a gateway model may propagate commands to one or more devices, and an environmental event source may trigger a change in the behavior of other components. Therefore, events and messages must be routed to the proper model and, when multiple instances are running, to the proper process instance. Consequently, the models need to agree on the information exchanged through messages and events so that each exchange can be correctly correlated with the intended process instance. This introduces the need for event correlation mechanisms, which are discussed at the architectural level in Section 4.

The analysis focuses on the functional behavior observed during simulation. Designers and stakeholders can inspect whether the modeled entities react to events as expected, whether messages are exchanged in the intended order, whether timers and decision conditions activate the expected branches, and whether the simulated behavior reaches the expected states. The analysis can rely on execution states, exchanged messages, logs, and process traces produced during the simulation. Furthermore, the resulting BPMN models could support subsequent formal analysis of behavioral properties, such as deadlock freedom, using suitable process-analysis techniques [9].

When the observed behavior does not match the expected logic, the results of the analysis are used to refine the BPMN models. This may involve adding missing events, revising decision conditions, changing message exchanges, or restructuring parts of the behavioral model. The refined models can then be executed again, closing the iterative loop introduced in Fig. 1. Once the modeled behavior is considered consistent with the expected logic, the process leads to the consolidation of implementation-oriented artifacts. These artifacts include BPMN models refined through simulation-based analysis, explicit interaction logic, and guidance for subsequent implementation, validation, and testing activities.

4. Digital Twin Prototype Architecture

This section presents the reference architecture, its main components, and a related prototype implementation developed to execute and inspect the BPMN behavioral models.

4.1. Reference Architecture

The proposed DTP architecture is intended to support the execution and behavioral analysis of the models representing the IoT system under design. The architecture introduces a set of components required to configure and control simulation runs, manage the digital representation of the involved IoT entities, execute the behavioral models, coordinate interactions among components, and inspect the resulting behavior.

Fig. 2 shows the reference architecture, which is organized around five main components: the Simulation Manager, the IoT Platform, the Event Bus, the Middleware, and the BPMN Execution Engine.

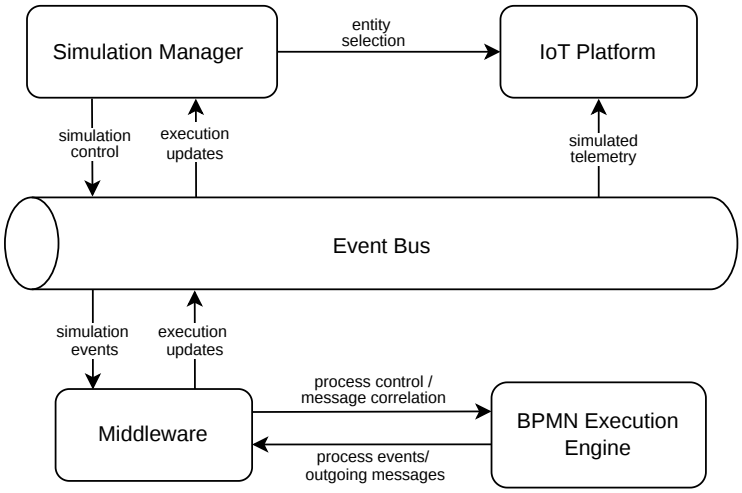

Figure 2: Reference architecture for the Digital Twin Prototype.

The Simulation Manager supports the configuration and control of simulation runs. It allows designers to select the IoT entities involved in a scenario, associate to them the corresponding BPMN behavioral models, define the events injected during the simulation, and start or stop simulation runs. In the reference architecture, this component acts as the entry point for controlling the simulation, while execution evidence can be inspected through the BPMN Execution Engine and the supporting logs.

The IoT Platform manages the digital representation of the entities involved in the scenario. It supports the registration of devices and simulated devices, the definition of their attributes, and the collection of simulated telemetry produced during simulation runs. In the reference architecture, it provides the information needed by the Simulation Manager to configure the IoT scenario and can receive simulated telemetry from the Event Bus and associate it with the corresponding digital entities under simulation.

The Event Bus enables asynchronous communication among the architectural components. It supports the exchange of simulation events, simulated telemetry, commands, execution updates, and

outgoing process messages. By decoupling event producers from event consumers, it allows multiple behavioral models and supporting components to interact without direct dependencies.

To integrate the execution of the BPMN models in the simulation scenario, the architecture introduces a Middleware component that provides an adaptation layer between the Event Bus and the BPMN Execution Engine. This adaptation operationalizes the conceptual mapping introduced in Table 2 by translating simulated sensor readings and external events into BPMN events, routing message exchanges to the proper process instances, and using entity identifiers as correlation information to connect simulation events with the corresponding behavioral models. It can also support process control operations, such as starting or stopping process instances according to simulation control events.

Finally, the BPMN Execution Engine runs the behavioral models defined during the modeling phase, instantiating and managing their execution. It handles process instances, events, tasks, gateways, and process variables, exposes execution states, and produces process events or outgoing messages that are propagated to the other components through the Middleware and the Event Bus.

Overall, the architecture separates the main system components, supporting asynchronous and decoupled interactions among behavioral models, while the Middleware keeps process-control and message-correlation logic separate from the BPMN models and the Event Bus. This decoupling facilitates the independent deployment and evolution of the components and supports the integration of additional components in future extensions.

4.2. Prototype Implementation

A prototype implementation of the core execution and coordination mechanisms of the DTP architecture has been developed to support the execution of BPMN behavioral models according to the approach described in Section 3. In particular, the prototype addresses the integration between the simulated IoT scenario and the BPMN process execution environment, with the objective of assessing whether the conceptual mapping introduced in Table 2 can be operationalized in a controlled execution setting.

The prototype implements the core execution loop involving a Kafka-based Event Bus, a Middleware component, and a BPMN Execution Engine. The BPMN Execution Engine is implemented using Camunda 8.3, while the Event Bus relies on Kafka 3.8. The Middleware has been implemented as a Dockerized microservice written in Go. The source code of the prototype is publicly available in the project repository.[1]

In the current implementation, Kafka is mainly used to control the simulation and to inject the information required to configure the involved scenario entities, such as the identifiers of simulated devices and gateways. These data provide the contextual information needed to instantiate and correlate the BPMN process instances involved in a simulation run.

The Middleware consumes simulation control events from Kafka and interacts with Camunda to instantiate the process models defined during the behavioral modeling phase. When a process instance is created, the identifiers of the corresponding scenario entities are stored as process variables and used as correlation information. Once the process instances are running, the Middleware acts as the coordination layer for the communication among them. It handles the messages produced by the BPMN models, extracts the routing information from the message payload, and uses the entity identifiers to correlate each message with the process instance that is expected to consume it. In this way, messages generated by one behavioral model can be delivered to the appropriate receiving model without requiring direct point-to-point dependencies among the BPMN processes.

Camunda is used to deploy and execute the BPMN models, manage process instances, handle BPMN events and tasks, and expose the execution state of running and completed instances. This makes it possible to inspect the execution of the modeled behavior through the engine interface and the logs produced by the involved services. The concrete BPMN models are presented in the SAFE use case described in Section 5.

[1] https://github.com/PROSLab/BPMN-DTP-IoT

5. SAFE Use Case Scenario

This section presents the application of the proposed approach to the SAFE scenario, focusing on scenario analysis, BPMN behavioral modeling, and controlled simulation execution.

5.1. IoT Scenario Analysis

The proposed approach was applied to the SAFE scenario, an IoT-based system designed to support protection, localization, and rescue operations in the event of an earthquake. SAFE investigates anti-seismic furniture for schools and offices, where traditional furniture is integrated with battery-powered wireless IoT devices and an ICT infrastructure to support the detection and localization of people after a seismic event [20].

This work focuses on the behavioral aspects of the SAFE scenario that are relevant for the execution of the DTP. In particular, the scenario analysis considers the SAFE Passive Infrared (PIR) device, which detects presence under the furniture and adapts its behavior when an earthquake-related condition occurs. The analysis identified the main entities to be modeled: the SAFE PIR device, the stationary gateway, the presence-detection event source, and the earthquake-related event source.

The relevant interactions include presence-detection events, earthquake-related events, messages between the SAFE PIR and the stationary gateway, acknowledgments, commands, timers, mode changes between normal and emergency behavior, and the activation of broadcast communication in emergency mode. These elements provide the basis for the BPMN behavioral models described in the next subsection.

5.2. BPMN Behavioral Models

Following the modeling approach introduced in Section 3, the SAFE scenario was decomposed into a set of BPMN behavioral models representing either a scenario entity or a simulation-related component, while the overall behavior emerges from their interaction through events and messages.

The implemented set includes four BPMN models: the SAFE PIR model, the presence-signal model, the stationary gateway model, and the earthquake-signal model. Table 3 summarizes their roles in the prototype.

Table 3
BPMN behavioral models implemented for the SAFE scenario.

Model	Role	Main behavior represented
SAFE PIR	IoT sensing device	Models the behavior of the PIR device, including peace/war mode, presence detection, timers, communication mode, and data transmission.
Presence signal	Simulation input	Provides a controlled user task to set the presence-detection status during simulation.
Stationary gateway	Gateway / edge component	Receives requests from PIR devices and sends PEACE or WAR commands according to the current earthquake status.
Earthquake signal	Environmental event source	Provides a controlled user task to set whether an earthquake condition is active during simulation.

The SAFE PIR model, shown in Fig. 3, represents the behavior of the sensing device integrated into the anti-seismic furniture. The model starts from the device configuration and distinguishes between peace and war mode. Depending on the current mode, the model evaluates presence-detection information, handles communication timers, selects the communication behavior, and sends presence-related data. Message events represent the interactions with the stationary gateway, while gateways and process variables encode the mode-dependent behavior.

The stationary gateway model, shown in Fig. 4, is modeled as a separate BPMN process. Its role is to receive requests from SAFE PIR devices and respond with the appropriate command. In particular,

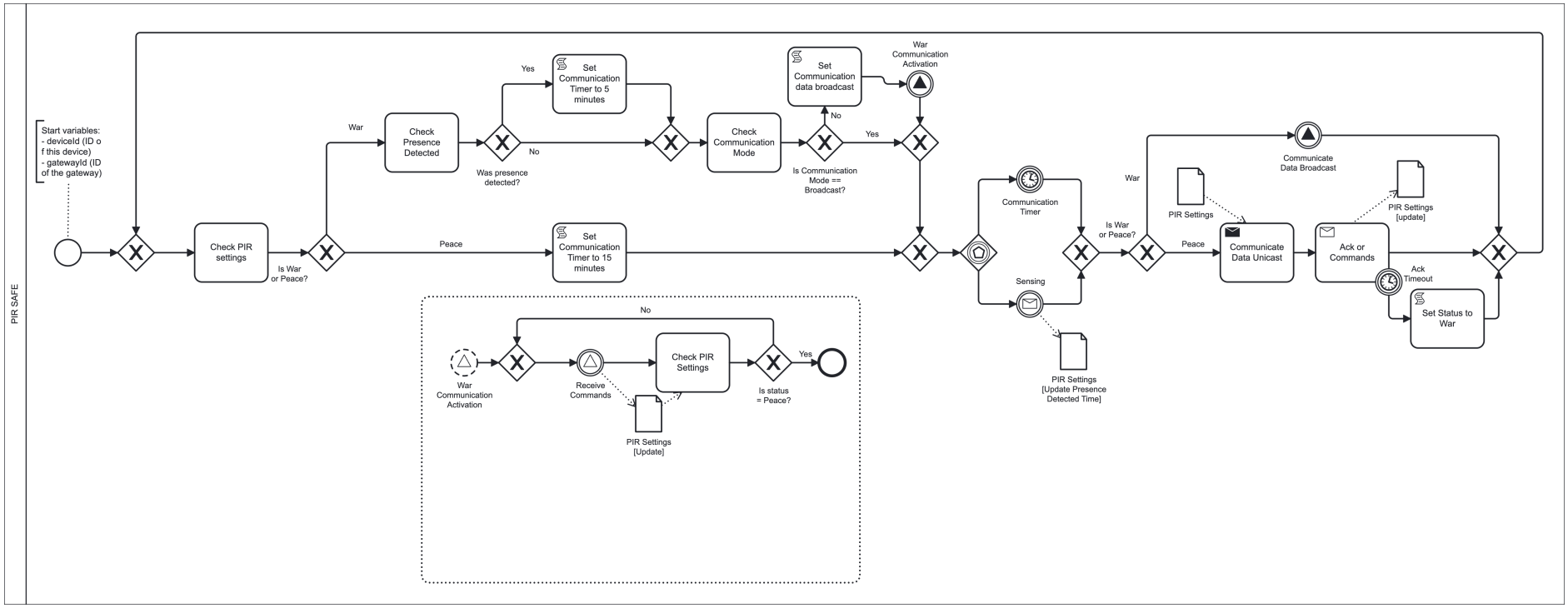

Figure 3: BPMN behavioral model of the SAFE PIR device.

the gateway sends a PEACE or WAR command according to the current earthquake status. This status is updated by the earthquake-signal model, which represents the environmental event source used to control the occurrence of an earthquake condition during simulation.

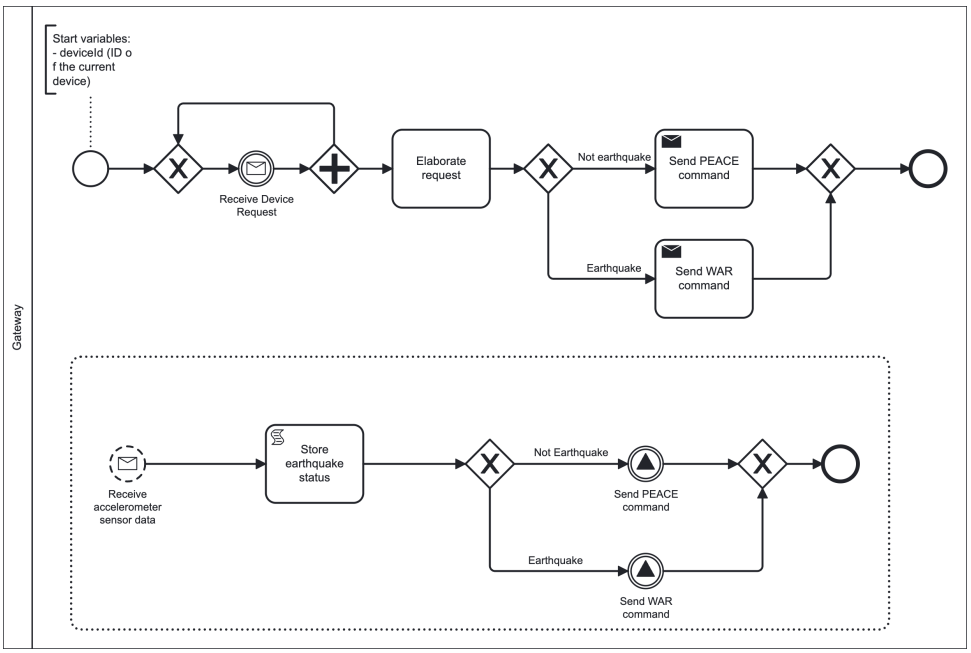

Figure 4: BPMN behavioral model of the stationary gateway.

The presence-signal and earthquake-signal models, shown in Fig. 5, are introduced to support controlled simulation. The models expose user tasks through which the user can inject controlled values during the simulation. In particular, Fig. 5(a) shows the model used to set whether presence has been detected, while Fig. 5(b) shows the model used to set whether an earthquake condition is active. The selected values are then propagated as events and consumed by the SAFE PIR model and the stationary gateway model, respectively.

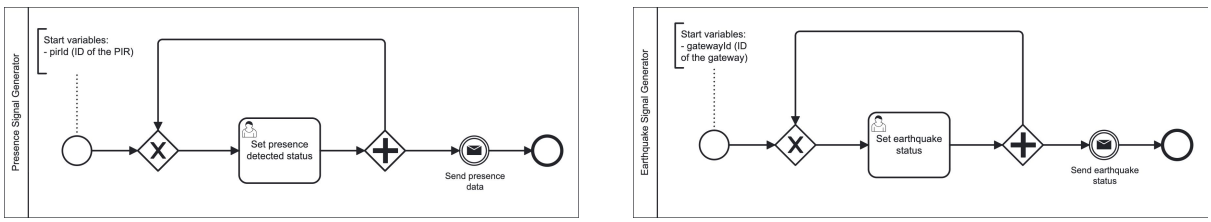

Figure 5: Simulation-control BPMN models: (a) presence-signal model and (b) earthquake-signal model.

The separation of these behaviors into different BPMN models makes the interactions among scenario entities explicit and allows the DTP of the IoT system to be executed and inspected as a set of coordinated behavioral models.

5.3. Simulation Execution

To execute a simulation run, the BPMN behavioral models defined in the previous phase are first deployed in Camunda. The simulation is then configured by specifying the logical identifiers of the entities involved in the scenario. These identifiers provide the addressing and correlation information needed to coordinate the simulated entities during execution. For example, the SAFE PIR model is configured with the logical identifier of the simulated device and with the identifier of the gateway

to which its messages must be sent. Similarly, the earthquake-signal model is configured with the identifier of the gateway that must receive earthquake-related events, while the stationary gateway model is instantiated with the same logical identifier used by the other models to address it. These configuration data correspond to the kind of addressing information that would also be required in an actual deployment.

In the current prototype, the simulation configuration is encoded as JSON messages and sent to Kafka together with start commands for the involved models. The Middleware listens to the corresponding Kafka topic and, when a start command is received, interacts with Camunda to create the required process instances. During process instantiation, the configuration data are injected as process variables. These variables include the logical identifiers of the involved entities and are later used as correlation information for routing messages among the running process instances.

Once the process instances are active, the execution proceeds through the exchange of BPMN messages among the models. When a model instance produces a message, the Middleware receives the message, inspects its payload, and identifies the target entity according to the configured identifiers. It then uses the corresponding correlation information to deliver the message to the process instance that is expected to consume it. In this way, the Middleware coordinates the communication among independently executed BPMN models while preserving the logical relationships defined during simulation configuration.

After the process instances have been created and correlated, the simulation can be controlled through the user tasks included in the simulation-related BPMN models. Fig. 6 illustrates how the SAFE simulation can be controlled and inspected in Camunda. The SAFE PIR process waits for either the polling timer or a presence-related event (Fig. 6(a)), while the stationary gateway receives messages from the PIR process (Fig. 6(b)). The earthquake-signal process includes the user task used to control the earthquake condition (Fig. 6(c)), whose associated form allows the corresponding value to be injected during the simulation (Fig. 6(d)).

The execution setup allows controlled environmental conditions to be injected and their effects to be observed on the interacting BPMN models. When an earthquake condition is injected, the event is propagated to the stationary gateway, which can produce the command that causes the SAFE PIR process to change its operating mode. Similarly, presence-related values can be injected and consumed by the SAFE PIR process to activate the corresponding behavioral branches.

The SAFE scenario illustrates how the proposed approach can support the analysis of DTP behavior by making interactions among IoT entities explicit and executable. Execution traces and message exchanges provide evidence for checking the behavior represented by the BPMN models. The resulting models and execution artifacts can support subsequent design and implementation activities, such as refining device firmware or gateway software modules.

6. Discussion

The proposed approach focuses on the behavioral layer of IoT DTPs, using BPMN to represent the behavior of IoT entities as executable and inspectable models. In the SAFE scenario, this made it possible to decompose the behavior of the system into multiple interacting models, representing the SAFE PIR device, the stationary gateway, and the event sources used to control the simulation. Although the SAFE system was originally developed independently of the proposed approach and the modeled scenario covers only a subset of it, the case provides a suitable basis for assessing the feasibility of the proposed modeling and execution approach. Within this scope, the prototype provides preliminary evidence that these models can be executed as coordinated artifacts, with the middleware supporting process instantiation, configuration injection, and message routing among process instances.

The current implementation should be interpreted as a prototype focused on the BPMN execution and message-correlation layer, rather than as a complete Digital Twin platform. The IoT platform is not yet integrated in the execution loop. As a consequence, simulated telemetry, device registration, and entity management are not yet handled through a dedicated IoT platform. Integrating an open-source

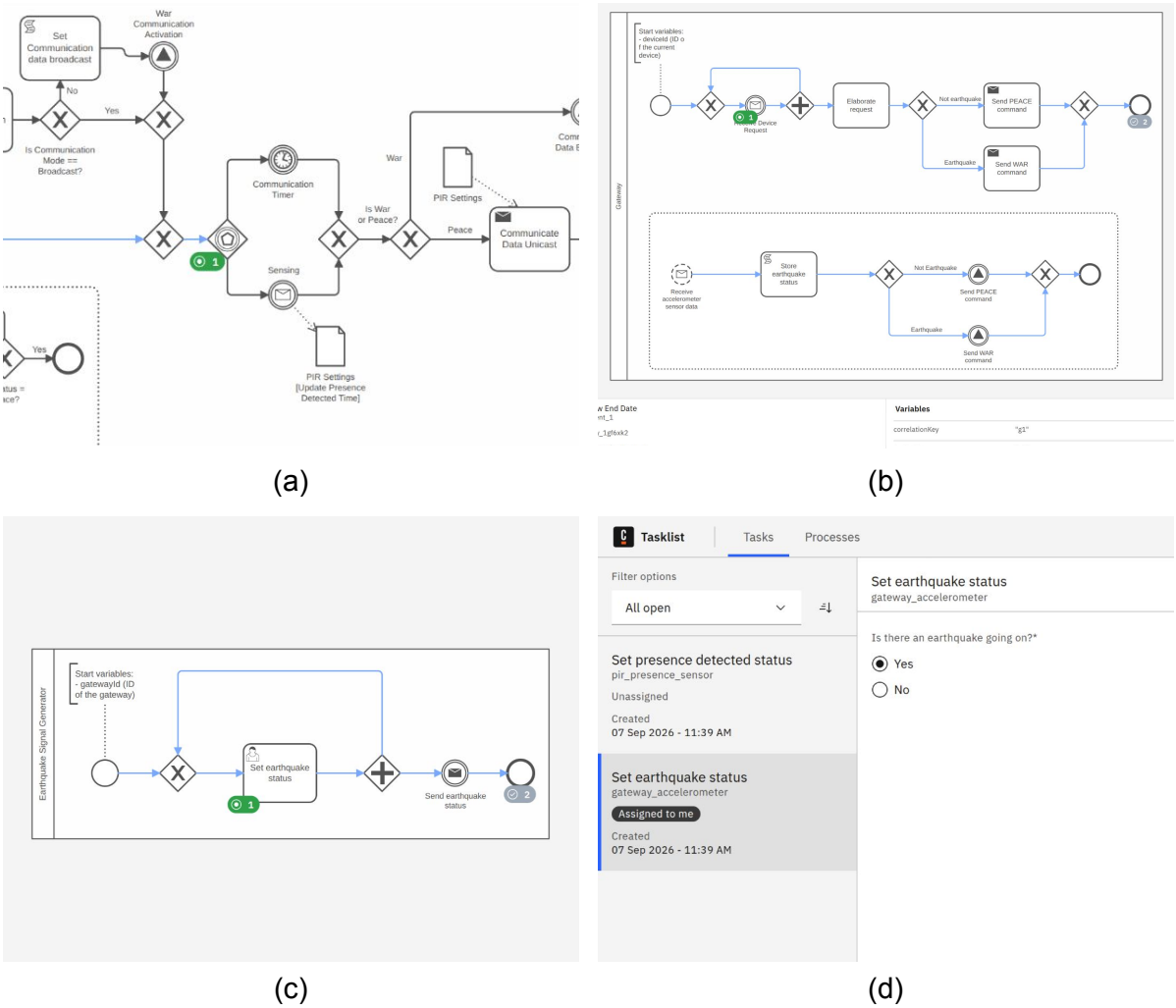

Figure 6: Controlled execution of the SAFE simulation in Camunda: (a) fragment of the SAFE PIR process waiting for polling or presence-related events; (b) stationary gateway process with message reception; (c) earthquake-signal process with the user task for controlling the earthquake condition; and (d) form used to inject the earthquake-related event.

IoT platform, such as the DThingsBoard extension introduced in previous work [21], would allow the BPMN behavioral models to be connected with the digital representation of the involved IoT entities and would support the replacement of platform-specific rule-based behavior with BPMN-based execution.

Scalability was not an objective of the current prototype and has not yet been evaluated. Future work should investigate execution scalability, in terms of interacting entities, event rates, throughput, and latency, as well as modeling scalability, considering the effort required to define, maintain, and coordinate an increasing number of behavioral models and interactions.

Another aspect that remains preliminary is the analysis of simulation results. At the current stage, the behavior of the DTP is inspected through the Camunda interface and service logs. This is suitable for technical validation, but can be less accessible for non-expert stakeholders. A dedicated Simulation Manager is therefore needed to centralize simulation configuration, control, and analysis, providing higher-level views of process states, exchanged messages, injected events, and observed behavioral traces.

Overall, the results indicate that BPMN can make the behavioral assumptions of an IoT DTP explicit and executable. However, integration with an IoT platform and centralized simulation-analysis facilities are required to move from the current prototype toward a more complete and usable DTP environment.

7. Conclusion and Future Work

This paper presented an approach for supporting the design and early behavioral analysis of IoT DTPs through executable BPMN models. The approach makes the expected behavior of IoT entities explicit, executable, and inspectable before the complete physical system is available. A reference architecture was introduced, together with a prototype implementation of its core execution and coordination mechanisms. The prototype, based on Camunda, Kafka, and a custom middleware, was applied to an earthquake emergency scenario derived from the SAFE project to execute and inspect interacting behavioral models of the SAFE PIR device, the stationary gateway, and simulation-related event sources.

Future work will focus on integrating the prototype with an IoT platform and implementing a dedicated Simulation Manager for scenario configuration, execution control, and centralized analysis. The approach will also be evaluated on IoT scenarios beyond SAFE, involving different device behaviors and interaction patterns, to better characterize its applicability and the suitability of BPMN in different IoT settings. Finally, further work will investigate the integration of BPMN-based behavioral models with 3D and Virtual Reality technologies to support more interactive ways to experience and analyze IoT DTPs during design-time validation.

Acknowledgments

This work has been funded by the European Union - NextGenerationEU, Mission 4 Component 1 CUP J11J24001890006 and supported by the UPRAISE project (Virtual Worlds Innovation Masters: Shaping Future Digital Skills for Europe), funded by the European Union under the Digital Europe Programme (DIGITAL), Grant Agreement No. 101225920.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and DeepL for grammar and spelling checks and rewording. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: Transdisciplinary perspectives on complex systems: New findings and approaches, Springer, 2016, pp. 85–113.

[2] D. Jones, C. Snider, A. Nassehi, J. Yon, B. Hicks, Characterising the digital twin: A systematic literature review, CIRP journal of manufacturing science and technology 29 (2020) 36–52.

[3] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, F. Sui, Digital twin-driven product design, manufacturing and service with big data, The International Journal of Advanced Manufacturing Technology 94 (2018) 3563–3576.

[4] A. Fuller, Z. Fan, C. Day, C. Barlow, Digital twin: Enabling technologies, challenges and open research, IEEE access 8 (2020) 108952–108971.

[5] M. Callisto De Donato, F. Corradini, F. Fornari, B. Re, Safe: An ICT platform for supporting monitoring, localization and rescue operations in case of earthquake, Internet of Things 27 (2024).

[6] M. Callisto De Donato, F. Corradini, F. Fornari, B. Re, M. Romagnoli, Design and development of a digital twin prototype for the SAFE project, in: Enterprise Design, Operations, and Computing. EDOC 2023 Workshops, volume 498 of *Lecture Notes in Business Information Processing*, Springer, 2023, pp. 107–122.

[7] P. Valderas, V. Torres, E. Serral, Towards an interdisciplinary development of iot-enhanced business processes, Bus. Inf. Syst. Eng. 65 (2023) 25–48.

[8] P. Valderas, Supporting the implementation of digital twins for iot-enhanced bps, in: International Conference on Research Challenges in Information Science, Springer, 2023, pp. 222–238.
[9] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, G. Fortino, A. Gal, U. Kannengiesser, F. Leotta, F. Mannhardt, A. Marrella, J. Mendling, A. Oberweis, M. Reichert, S. Rinderle-Ma, E. Serral, W. Song, J. Su, V. Torres, M. Weidlich, M. Weske, L. Zhang, The internet of things meets business process management: A manifesto, IEEE Systems, Man, and Cybernetics Magazine 6 (2020) 34–44.
[10] F. Fornari, I. Compagnucci, M. Callisto De Donato, Y. Bertrand, H. H. Beyel, E. Carrión, M. Franceschetti, W. Groher, J. Grüger, E. Kilic, A. Koschmider, F. Leotta, C.-Y. Li, G. Lugaresi, L. Malburg, J. Mangler, M. Mecella, O. Pastor, U. Riss, R. Seiger, E. Serral, V. Torres, P. Valderas, Digital twins of business processes: A research manifesto, Internet of Things 30 (2025) 101477.
[11] V. Torres, E. Serral, P. Valderas, V. Pelechano, P. Grefen, Modeling of iot devices in business processes: a systematic mapping study, in: 2020 IEEE 22nd conference on business informatics (CBI), volume 1, IEEE, 2020, pp. 221–230.
[12] P. Valderas, V. Torres, E. Serral, Modelling and executing iot-enhanced business processes through bpmn and microservices, Journal of Systems and Software 184 (2022) 111139.
[13] A. Fedeli, F. Fornari, A. Polini, B. Re, V. Torres, P. Valderas, et al., Flobp: a model-driven approach for developing and executing iot-enhanced business processes, Software and Systems Modeling 23 (2024) 1217–1246.
[14] Y. Bertrand, A. Stevens, B. Deforce, J. De Smedt, J. De Weerdt, E. Serral, Approaches for iot-enhanced predictive process monitoring, Process Science 2 (2025) 7.
[15] I. Compagnucci, F. Corradini, F. Fornari, A. Polini, B. Re, F. Tiezzi, A systematic literature review on iot-aware business process modeling views, requirements and notations, Software and Systems Modeling 22 (2023) 969–1004. URL: https://doi.org/10.1007/s10270-022-01049-2.
[16] J. Sleuters, Y. Li, J. Verriet, M. Velikova, R. Doornbos, A digital twin method for automated behavior analysis of large-scale distributed iot systems, in: 2019 14th Annual Conference System of Systems Engineering (SoSE), IEEE, 2019, pp. 7–12.
[17] L. Nguyen, M. Segovia, W. Mallouli, E. M. d. Oca, A. R. Cavalli, Digital twin for iot environments: a testing and simulation tool, in: International Conference on the Quality of Information and Communications Technology, Springer, 2022, pp. 205–219.
[18] G. Georgiev, R. Trifonov, D. Gotseva, G. Kotov, Digital twin-driven testing of iot systems: A scalable simulation-based approach, in: 2025 13th International Scientific Conference on Computer Science (COMSCI), IEEE, 2025, pp. 1–6.
[19] L. Maillet-Contoz, E. Michel, M. D. Nava, P.-E. Brun, K. Lepretre, G. Massot, End-to-end security validation of iot systems based on digital twins of end-devices, in: 2020 Global Internet of Things Summit (GIoTS), IEEE, 2020, pp. 1–6.
[20] L. Pietroni, J. Mascitti, D. Galloppo, Life-saving furniture during an earthquake. intelligent, interconnected and interacting, AGATHÓN| International Journal of Architecture, Art and Design 10 (2021) 218–229.
[21] M. Callisto De Donato, F. Corradini, F. Fornari, B. Re, M. Romagnoli, Enabling 3d simulation in thingsboard: A first step towards a digital twin platform, in: T. P. Sales, S. de Kinderen, H. A. Proper, L. Pufahl, D. Karastoyanova, M. van Sinderen (Eds.), Enterprise Design, Operations, and Computing. EDOC 2023 Workshops, Springer Nature Switzerland, Cham, 2024, pp. 325–330.