

Granularity Patterns for Programming Cyber-physical Systems with Business Processes

Ronny Seiger¹[0000–0003–1675–2592], Olaf Zimmermann², and Daniel
Lübke³[0000–0002–1557–8804]

¹ RWTH Aachen University, Aachen, Germany ronny.seiger@rwth-aachen.de

² Institute of Computer Science, University of St.Gallen, St.Gallen, Switzerland
olaf.zimmermann@unisg.ch

³ Digital Solution Architecture, Hannover, Germany
daniel.luebke@digital-solution-architecture.com

Abstract. Business process management (BPM) systems and corresponding notations have shown their applicability and flexibility in modeling, automating, and mining business processes in enterprise applications and service-oriented architectures for many decades. Becoming increasingly open and accessible via software-based interfaces, cyber-physical systems (CPS)—systems interacting with the physical world via sensors and actuators—benefit from BPM-based automation too, as they are inherently process-driven and software-intensive. We present four patterns discussing the choice of appropriate levels of detail (*granularities*) and corresponding decision forces regarding what to model in executable business processes and what to implement in traditional programming languages in CPS software. Two patterns give advice on when to model *sensing* in CPS at a more fine-grained or on a more coarse-grained level in business processes. Similarly, two patterns discuss these granularity choices for *actuation* in CPS. These four patterns guide the modeling and implementation of executable business processes that can be used for combining and automating sensing and actuation of CPS.

Keywords: Business Process Management · Granularity · Cyber-physical Systems · Design Pattern · Sensing and Actuation.

1 Introduction

Business Process Management (BPM) systems and notations have been used for modeling, automating, executing, and analyzing business processes in enterprise contexts for many years [17,46]. Modeling languages such as the industry standard BPMN 2.0 simplify the graphical modeling and representation of business process for domain and business experts. They facilitate communication and analysis, but they also feature execution semantics allowing for automation. BPM systems instantiate and execute business processes with support of 1) model versioning and migration, 2) state management for process instances, 3) connectors to external systems and human tasks, and 4) process monitoring and creation of event logs for process mining.

Business processes and BPM systems have been playing an essential role in enterprise applications in business domains such as retail [30], insurance, finance, telecommunications [46] as well as in software architecture methods and tools [47]. Their application for automating processes in cyber-physical systems (CPS) and Internet of Things (IoT) has only recently attracted attention of researchers and practitioners (e.g., within the BPM-IoT community [11]). Processes are ubiquitous in these domains too, and they benefit from automation based on BPM systems and explicitly and graphically modeled business processes. However, CPS and IoT introduce new requirements and decision forces related to their close interactions and feedback loops with the physical world as one of their main new characteristics [21]. Safety, sustainability, and operational resilience are among these new domain-specific forces [8,42]. CPS might produce high volumes of streams of sensor data and they require fast reactions to sensor and system events in control (feedback) loops to ensure safety when interacting with the physical world. BPM notations and systems are not designed for representing and supporting this behavior at the required fine-grained implementation levels. Some parts of the sensing and actuation can be modeled at the business process level, but other parts have to be left to the code-based implementation of the CPS due to safety requirements. Nevertheless, CPS benefit from BPM-based automation as processes become more explicit, traceable, and analyzable, which is especially helpful when multiple components and systems are involved.

In this work we report how standard business process modeling notations and BPM systems are applied to *program* CPS (i.e., to automate them via executable business processes). Based on more than a decade of research and development experience in this field, we discuss patterns that focus on the modeling and execution of novel, CPS-specific aspects—namely *sensing* (i.e., the perception of CPS and their surroundings via sensors, e.g., temperature and movement sensors) and *actuation* (i.e., the manipulation of the physical world via actuators, for example those controlling motors)—as part of business processes in CPS. These patterns revolve around choosing the appropriate level of detail—*granularity*—of actuation and sensing tasks in the business process models that describe executable processes in CPS. The known uses for the presented patterns stem from collaborations and information exchanges with industry, from literature search, and from own research in CPS lab design, implementation, evolution, and maintenance. This work targets software developers and system engineers tasked with automating processes in CPS that span multiple software components and interactions with the physical world via sensors and actuators.

2 Motivating Example and Problem Statement

To illustrate the application of business process technologies for “programming” CPS, we first introduce a smart manufacturing laboratory setup that we have developed as a testbed for our research. Then, we demonstrate the BPM-based automation using an example process that involves multiple CPS components

and systems. We then introduce the general problem statement which is addressed by the patterns presented in this work.

The smart manufacturing example illustrates and emphasizes the characteristics of modern CPS that integrate software-controlled hardware to interact with the physical world with feedback loops consisting of sensing, processing, and actuation (sense, analyze, act) between the physical and digital worlds [21]. Among the main building blocks of CPS are 1) *actuators* (e.g., the motors of the mobile robots) that are able to manipulate the physical world and its properties; and 2) *sensors* (e.g., the camera on the mobile robot) that are able to measure physical world properties [1]. We focus on these two important classes of components and their combinations within the scope of this work as they are the enablers of software-based physical world interactions.

2.1 Smart Manufacturing CPS Laboratory

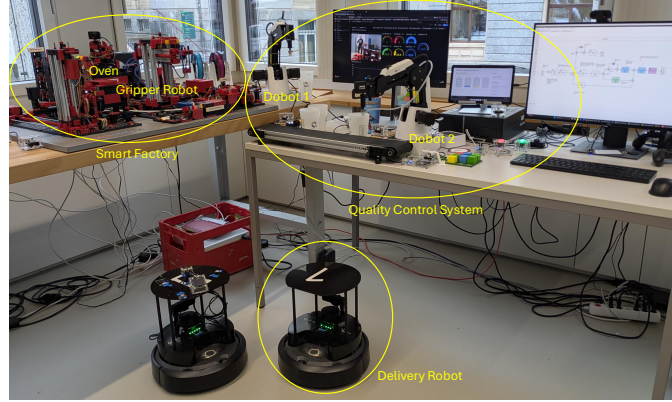


Fig. 1. Laboratory with several CPS for our scenario process.

In our smart manufacturing laboratory simulating an intelligent, interconnected and software-intensive manufacturing environment, we use several CPS to investigate realistic production and logistics processes. As depicted in Fig. 1, this setup includes a smart factory with several production stations that are controlled by a custom Python-based software stack [38]. Furthermore, there is a quality control station with two robot arms, a conveyor belt, and various sensors which are all also controlled by custom Python-based software [33]. A mobile delivery robot with autonomous navigation capabilities based on the ROS2 operating system [22] complements the setup.

2.2 Example Process to Orchestrate CPS in Smart Manufacturing

The focus of our work is the BPM-based orchestration of CPS, i.e., automating CPS primarily using business processes, but also traditional programming lan-

guages. Fig. 2 shows a representative, executable business process model following the Business Process Model and Notation (BPMN 2.0). The model comprises multiple CPS as self-contained systems. The individual CPS are represented using *pools* and *swimlanes* to show their responsibilities in the business process.

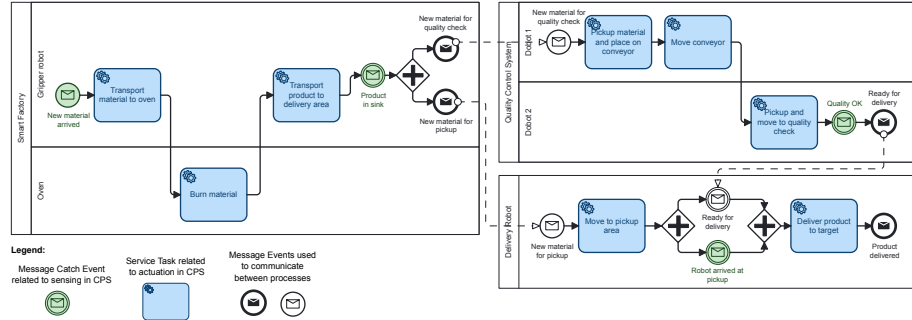


Fig. 2. Example of an executable process model in BPMN 2.0, which orchestrates the different CPS, including sensing and actuation modeled at a specific granularity.

The business process starts with the gripper robot of the smart factory registering the new arrival of material via sensing. The robot then transports the material to the oven, where it is burnt next. Afterwards, the gripper robot transports the product to a pickup station. A *Dobot* robot arm of the Quality Control System and the Delivery Robot receive notifications of the product being ready. The Delivery Robot is sent by the BPM system to the pickup area, and the Dobot 1 picks up the product. It places the product on the conveyor belt, which is then activated, and the product picked up and placed at quality control by Dobot 2. When quality control notifies that the product is OK, it can be picked up by the Delivery Robot upon its arrival.

The example process in Fig. 2 highlights CPS-related sensing, represented as *Message Catch Events* in BPMN and CPS-related actuation, represented as *Service Tasks* in BPMN. These elements are modeled at a specific level of detail (granularity) in the process as chosen by the process/systems engineer who traded off the relevant decision forces to be discussed next. The more fine-grained details are left to the source code-based implementation of the sensing and actuation outside the business processes. With this work, we provide guidance in choosing the appropriate levels of modeling sensing and actuation in CPS in business processes and leaving the remaining parts to other software components.

2.3 Problem Statement (Decision Forces)

The aspect of choosing appropriate granularities for BPM-related tasks has been identified as one of the main challenges in BPM [4]. We focus our investigations on the granularity of *sensing* and *actuation* in process controlled CPS.

Sensing in the business process refers to message-based events received from an external component (e.g., *Robot arrived at pickup* from the delivery robot control system, cf. Fig. 2), outside the business process (model). Here the execution engine waits for an event trigger to continue the execution. CPS architects and developers have to decide about the granularity of these sensor-related events inside the business process model.

Actuation invokes the CPS to execute tasks, i.e., to perform work, involving its actuators with processing logic (e.g., *Move to pickup area* to be executed by the delivery robot, cf. Fig. 2). Using a BPMN service task, we assume that the execution engine calls an external system—the *CPS controller*—via its API. This CPS controller contains the specific implementation of the functionality in source code of a programming language such as Python, C++ or C#.

Reflecting on our previous work and research in developing process controlled CPS [33,34,35,36,38,39], we encountered and resolved the following *forces* that shape the granularity decisions for sensing and actuation in business processes:

- *Process model complexity* refers to the number of elements in the process model, their names and interrelations.
- *Process understandability* refer to how easy the logic of sensing and actuation in a process model can be understood by humans.
- *Reusability* refers to the ease of using existing modeling elements for sensing and actuation in other process models.
- *Process observability* refers to the degree of detail that the execution of sensing and actuation can be monitored at the process level.
- *Ownership of CPS code and APIs* refers to the degree of which access and modification to the CPS control code implementing the actual sensing and actuation is possible and how comprehensive the APIs of the CPS are [49].
- *Communication overhead* refers to the amount and frequency of information exchange between the BPM system and the CPS that is needed to execute the sensing and actuations modeled in a business process.
- *Real-time and safety* refers to requirements imposed by the CPS to realize and guarantee safety and real-time constraints during the execution of sensing and actuations. These are novel decision forces emphasized by CPS compared to more traditional BPM domains.

In general, we can observe a tension between the forces when shifting more fine-grained process details towards the business process models. Process observability, understandability, and reusability might be facilitated when more details are part of the process model, at the cost of an increasing process model complexity. The last three, more CPS-related forces address the processing of complex sensor data streams and decision making for actuation in near real-time, which go beyond common BPM system capabilities. They argue for moving these aspects outside the business processes into other software components.

The conflicts between these forces go back to the granularity decisions. They lead us to formulate the following concrete questions (recurring problems) that we address with our proposed patterns:

- Q1: At what granularity level do we represent and integrate sensing of CPS in executable business processes?
- Q2: At what granularity level do we represent actuation of CPS in executable business processes and what is left to their code-based implementation?

3 Granularity Patterns for Process-based Control of CPS

We present a pair of patterns related to *Sensing*, namely 1) *Fine-grained Sensing in Processes* and 2) *Coarse-grained Sensing via Transformation Services*, and a pair of patterns related to *Actuation*, namely 1) *Fine-grained Actuation in Processes* and 2) *Coarse-grained Actuation via Controllers* for process-based control of CPS. Both pattern pairs discuss the effects of different levels between more coarse-grained and more fine-grained process representation⁴.

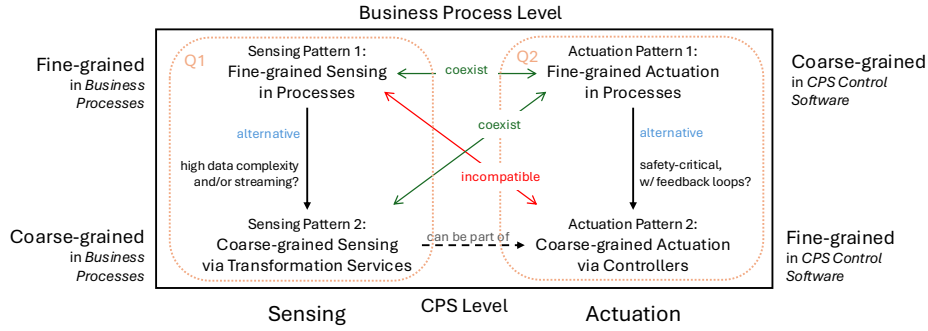


Fig. 3. Overview of the two granularity pattern pairs, each for sensing (question Q1) and for actuation (question Q2), along with their relationships. All patterns can be used standalone in different parts of a business process to implement different actions in the CPS. The relationship arrows indicate how they can be used together in a business process to implement *one action*.

Fig. 3 provides an overview of the patterns related to the granularity spectrum and their relations. The different granularity patterns for sensing and for actuation are alternatives when used for implementing a sensing or actuation action in a business process. The fine-grained sensing and actuation patterns and also the coarse-grained sensing and fine-grained actuation patterns might coexist. The coarse-grained sensing pattern can be part of the coarse-grained actuation pattern. The fine-grained sensing and coarse-grained actuation pattern are incompatible. In Fig. 3 we can also see that the main decision drivers for choosing the granularity level for a) sensing (left in Fig. 3) refer to characteristics

⁴ Note that there is a spectrum of multiple levels between fine-grained and coarse-grained sensing and actuations. Our discussions focus on the effects of tending towards a more fine-grained vs. a more coarse-grained business process implementation.

of the sensor data, and b) actuation (right in Fig. 3) refer to safety requirements in the CPS. We will elaborate on these aspects further in the next sections.

This section is structured as follows: Subsection 3.1 presents the two sensing granularity patterns; Subsection 3.2 covers the two actuation alternatives. We apply and adjust the EuroPLoP pattern template as this: Context and Problem are covered on the subsection level (i.e., Sensing and Actuation); additional fine-grained Context, the Solution with Examples, and Known Uses are presented on the pattern level (i.e., individually for all patterns). Decision Forces Resolution and Related Patterns again are covered on the subsection level (i.e., jointly for the two sensing patterns and for the two actuation patterns) for comparison.

3.1 Sensing

Context. In typical CPS, a multitude of different types of sensors continuously measuring and reporting physical data⁵ (e.g., humidity, pressure, power levels) are installed. Data often is provided as *streams*. The streams emitted from software controlling the sensors can be used to trigger behavior in a business process reactively, e.g., in the form of a *Message Catch Event* modeled in the process.

Problem. When using a BPM-based approach to model and execute reactive behavior derived from sensing in CPS, one has to decide about the representation and granularity of sensing in a business process model, including the processing/evaluation of received sensor data.

Sensing Pattern 1: Fine-grained Sensing in Processes

Context This pattern is eligible if/when the sensor data is rather simple (e.g., discrete values), non-critical, and emitted only once or at a low frequency.

Solution Raw sensor data from the CPS controller is integrated and evaluated directly inside a business process (e.g., in a subprocess) at a fine-grained level by the BPMS (cf. Fig. 4).

Example In Fig. 5 we show the decomposition of the business-level events related to the “Quality OK” check from the example process depicted in Fig. 2 at a more fine grained level in a subprocess, e.g., in the form of multiple business/domain events with their raw data being evaluated in a decision table. Data from the different sensors (e.g., weight and temperature) is rather simple, only received once, and can be evaluated directly by the BPMS.

⁵ We denote the unprocessed data emitted directly from a sensor as *raw sensor data*.

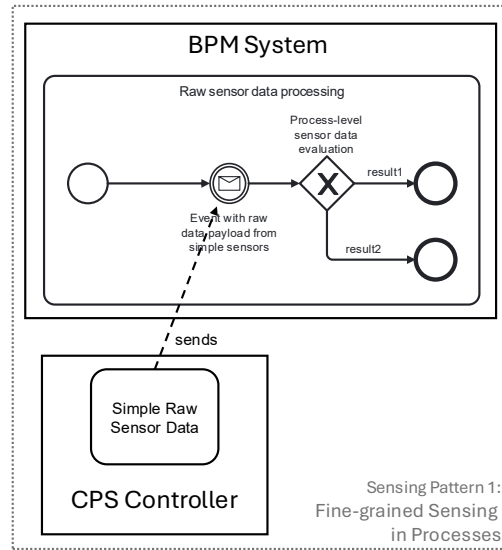


Fig. 4. Simple raw sensor data events are sent from the CPS Controller for direct processing in the business process at a fine-grained level.

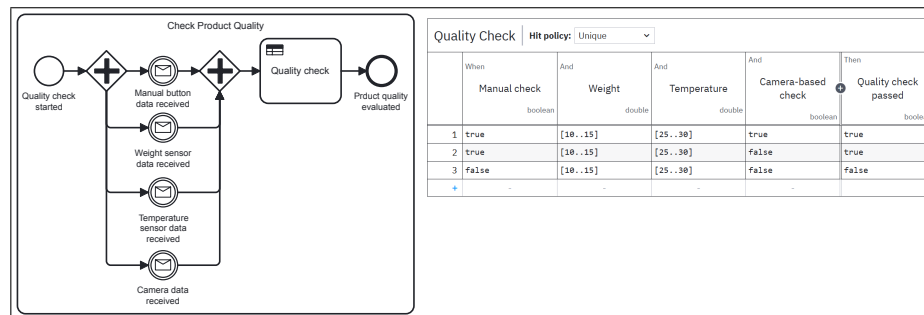


Fig. 5. Sensing based on raw sensor data evaluated in a separate decision model. This example is more fine-grained in the business process than the sensing depicted in Fig. 7.

Known Uses We identified this pattern from examples in our CPS laboratory where we employ simple sensors to trigger non-critical event-based behavior. Sensors such as buttons, light barriers, NFC readers, and motion sensors emit simple, discrete data—often only one specific value—when changes happen or they are actively used. These sensor values can be directly injected into a process variable and evaluated by the BPMS⁶. The processing of raw sensor data in business processes has been proposed and analyzed in several related works: Meyer et al. discuss the use of dedicated, fine-grained sensing tasks in BPMN models as new class of activity [26]. In [40] and [19], the authors use raw sensor data from a small-scale smart factory to represent and trigger behavior at the business process level. An example referring to the air quality of a meeting room measured from the raw sensors values of CO₂ sensors in business processes is shown in [27]. Furthermore, Suri et al. represent fine-grained temperature sensing as active tasks of a business process in [41]. Schönig et al. describe an architecture and formalism to integrate IoT data into business processes in [31]. They also discuss abstractions and argue that only relatively simple, low-frequency data should be processed by the BPM systems, other processing should happen outside of the business processes.

Sensing Pattern 2: Coarse-grained Sensing via Transformation Services

Context This pattern is eligible if/when sensor data is complex (e.g., multiple values, continuous) and/or streamed from the CPS at high frequencies.

Solution A Transformation Service serving as an additional component receives data from the CPS controller. It abstracts the sensor streams into more coarse-grained events that are triggered in the business process (cf. Fig. 6).

Example In Fig. 7 we show the events related to the product quality check at a more coarse grained level in the business process (e.g., in the form of multiple business/domain events) than the raw sensor data evaluation in the business process (cf. Fig. 5). These events have been derived—abstracted from the raw sensor data—by external transformation services that for example use advanced means to evaluate the temperature and weight of the product.

Known Uses We identified this pattern from working with an extended setup of our CPS laboratory described in [37] and from discussions within the *Internet of Processes and Things (IoPT)*⁷ community [24]. In the extended setup, we use a higher number of sensors (e.g., distance sensors, accelerometers, light sensors) producing streams of sensor readings as time-series to monitor human activities. Sensor data streams contain multiple values and need to be combined and

⁶ Note that time/safety-critical behavior such as an emergency stop from the pressing of an emergency stop button would be excluded from this rule of thumb.

⁷ <https://zenodo.org/communities/iopt>

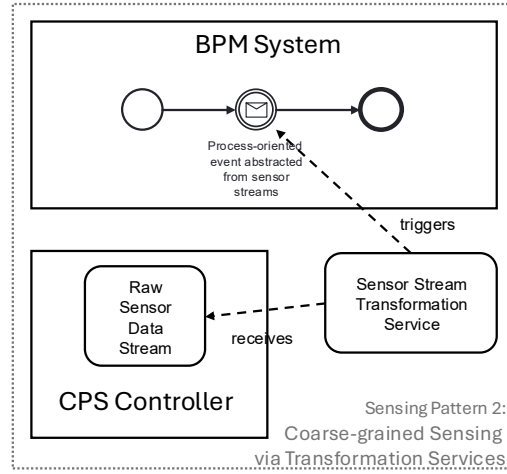


Fig. 6. Abstraction of complex raw sensor data streams from the CPS by an additional Sensor Stream Transformation Service component that processes received sensor data and triggers coarse-grained events in the business process.

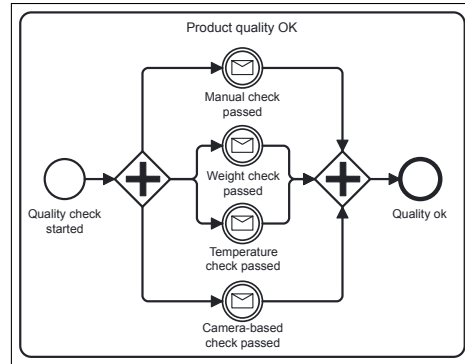


Fig. 7. Sensing-related events derived via external transformation services are represented with message catch events (e.g., “Weight check passed”) in a subprocess. This example shows the “Check Product Quality” subprocess in Fig. 5 at a more coarse-grained level as it does not evaluate raw sensor data in the business process events.

analyzed, often also regarding temporal or logical dependencies between one or multiple events. This analysis is not within the capabilities of a typical BPM system. The authors of [2,38,36,37] propose architectures that use dedicated event processing components to derive more coarse-grained events relevant for the business process from raw data such as GPS positions, environmental sensors, or in a clinical context [3]. The authors of [15] show examples from a healthcare scenario and the automated control of an HVAC system, partially working with values already abstracted by an external component from raw sensor data.

Decision Forces Resolution and Pattern Guidance. In Fig. 8 we present a *force field analysis* of how the relevant decision forces (cf. Section 2.3) are resolved by the two sensing patterns for CPS⁸. This analysis can be used to guide decisions regarding the choice of the sensing granularity levels by evaluating the decision forces resolution against project-specific non-functional requirements and constraints. We can only provide indications/trends of how forces are influenced based on the granularity decisions. A quantification is strongly dependent on the specific decision cases.

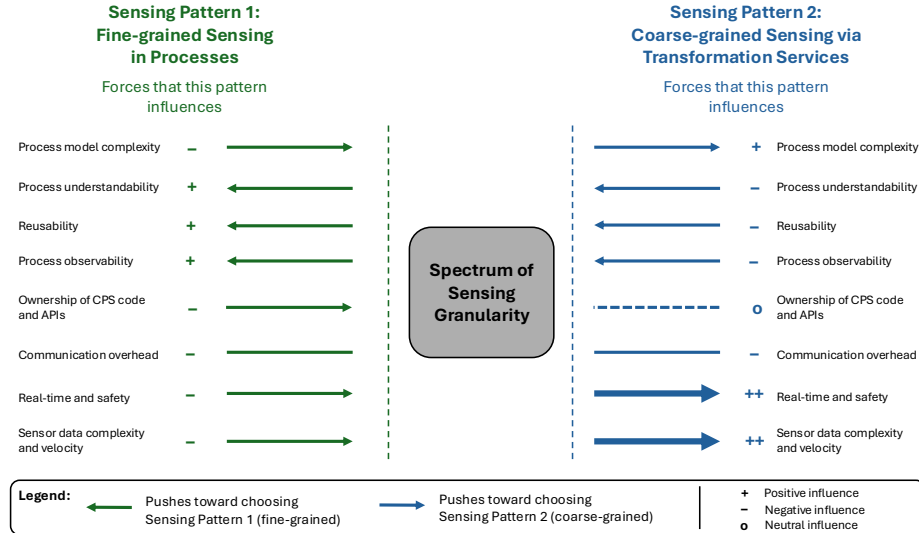


Fig. 8. Decision force field analysis for the two *sensing* granularity patterns.

Model complexity Process model complexity increases with more fine-grained modeling of sensing. Additional logic regarding the sensing might be in the

⁸ Note that while our discussion focus on sensors in CPS as main sources of raw data, the discussions can be generalized towards data/stream processing in business processes from arbitrary sources.

model (e.g., using gateways/decision tables) as permitted by the modeling language. However, this will further increase the process model complexity. More coarse-grained sensing in the process model shifts the complexity towards the implementation in the CPS Controller and/or Transformation Service.

Reusability Reusability (e.g., message events related to sensors) is facilitated when modeling sensing at a more fine-grained level as the corresponding BPMN elements can be reused in a more fine-grained manner. A more coarse-grained event which might be the result of processing different raw sensor data in the transformation component (Sensing Pattern 2) can only be reused as is.

Process understandability Process understandability is facilitated when modeling sensing and associated logic at a more fine-grained level (Sensing Pattern 1) as this makes the sensing data and processing logic more explicit, until the model becomes too complex to understand [32]. In the more coarse-grained Sensing Pattern 2 the processing logic is “hidden” in the sensor Transformation Service, which negatively influences process understandability.

Process observability A more fine-grained representation of sensing in a process model (Sensing Pattern 1) leads to a more fine-grained event log of the executions created by the BPM system, which facilitates process monitoring and analysis using process mining. The analysis of sensor processing at a more coarse-grained process level (Sensing Pattern 2) hinders process observability and requires the Transformation Service to log the sensing in more detail.

Ownership of CPS code and APIs The level of granularity to model sensing from external systems in the process is limited by their offered APIs. If complete ownership of the CPS control software is given, then developers can decide about the granularity of accessing individual CPS components, their sensors and the corresponding APIs. If the design of the CPS APIs is out of control (e.g., in legacy/brownfield CPS), the granularity-level of modeling sensing might be constrained, or additional software components have to be implemented on top of these APIs to enable more fine-grained sensing [34].

Communication overhead We assume that sensing is represented via one message catch event triggered by an external, potentially remote, system that is part of the CPS. A more fine-grained decomposition of business-level events into messages (Sensing Pattern 1) that each relate to the processing of sensor data (e.g., in a subprocess) requires these messages to be activated via distinct requests, which increase communication overhead. Sensing Pattern 2 introduces an additional software component, which also requires additional network communication, but triggers the business process level event only via one request.

Real-time and safety Sensing is critical when it is part of the internal control/feedback loops of the CPS. Real-time and safety-critical behavior related to

sensing is better supported by the more coarse-grained Sensing Pattern 2 as processing of the sensor data/streams can happen internally in the Transformation Service, not in the BPM systems. Other business process level sensing events might not have these constraints, arguing for a more fine-grained processing in the process (Sensing Pattern 1).

Sensor data complexity and velocity In CPS, sensor data are often emitted in the form of continuous data streams (e.g., as time-series data containing the raw sensor values), at varying—potentially very high—frequencies and complexities. Temporal and logical dependencies among sensor values might exist. A fine-grained representation of these sensor streams and their processing as suggested by Sensing Pattern 1 might not be feasible or even possible using standard process modeling notations and BPM systems. Thus, the more coarse-grained Sensing Pattern 2 should be preferred when sensor data exhibits these properties—leaving the processing to the external transformation services and only triggering coarse-grained events in the business process.

Related Patterns. If sensing is part of actuation (cf. pattern map in Fig. 3) that needs to provide real-time and safety-critical behavior, it should be part of the same task/activity that calls the lower level CPS control software to execute the actuation as described in *Actuation Pattern 2: Coarse-grained Actuation via Controllers* (cf. Section 3.2). The *Publisher* pattern in [6] addresses sensor data publication in the IoT in a more general way, without relation to business processes. In the uBPMN language, dedicated modeling elements for Sensor, Reader, Image, Audio, and Collector are proposed as BPMN 2.0 extensions to represent sensing at a more specific, fine-grained level [45]. These pattern can complement ours, as our sensing patterns do not address the specific sources of sensor data. The Cloud Computing Patterns include a *Data Abtractor* for processing—similar to the proposed transformation service (Sensing Pattern 2) [10]. The *Aggregator* pattern from the Enterprise Integration Patterns may be applied when realizing the sensor transformation service [18]; the BPM system acts as *Event-driven Consumer* [18] in both granularity patterns. The *Rules Engine* pattern from [29] can be applied to represent the evaluation of the sensor data at the BPM level or in the Sensor Transformation Service component.

3.2 Actuation

Context. Actuators in CPS involve combinations of sensors and actuators with processing logic and feedback loops to control devices and manipulate physical objects and the physical world. A feedback loop comprises a sensing, controlling, and actuating cycle. *Service Tasks* can be used to represent these actuators in a business process. These service tasks invoke functionality implemented by the specific CPS controller and exposed via service-based interfaces.

Problem. When using a BPM-based approach to automate processes in CPS, one has to decide about the representation and granularity of actuation in a process model while keeping the model executable. CPS-specific decision forces, namely safety and real-time critical behavior have to be considered.

Actuation Pattern 1: Fine-grained Actuation in Processes

Context This pattern is eligible in case of simple, non-safety critical actuation.

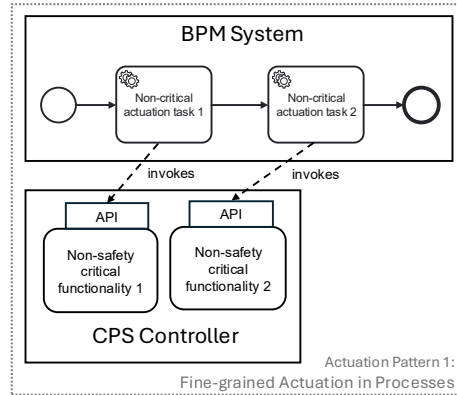


Fig. 9. Fine-grained Actuation in Processes with the BPM System invoking non-critical CPS Controller functionality at a fine-grained level.

Solution The actuation is modeled and enacted directly in the form of tasks/activities at the business process level in a fine-grained manner (cf. Fig. 9). The BPMS executes these tasks and invokes the corresponding functionality of the CPS.

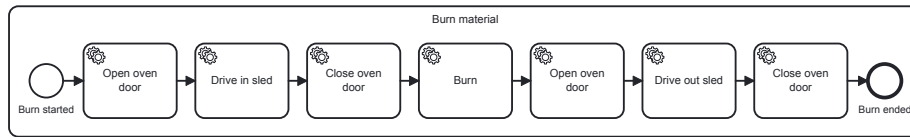


Fig. 10. Actuations represented by a sequence of service tasks invoking the smart factory API in a subprocess at a more fine-grained level as the “Burn material” activity of the scenario process in Fig. 2.

Example In the example process in Fig. 2 the activity *Burn material* is modeled as one activity without further details at a coarse granularity. In Fig. 10 we see

this activity decomposed into a subprocess that contains service tasks controlling the CPS at a more fine-grained level. The actuations are non-safety critical and allow for fine-grained execution at the business process level.

Known Uses We identified this pattern from use cases in our CPS laboratory and previous work in the context of smart homes [35] where simple fine-grained actuations (e.g., activating a motor, switching lights on, or activating a door opener) were sufficient to be triggered from a business process. These actuations are usually short-running/instantaneous, non-safety critical, or do not require additional sensing/control loops. We found further known uses of this pattern in the literature: The integration of IoT devices as performers executing fine-grained actuation tasks in business processes is presented in [7] and in [26]. Examples of modeling fine-grained production-related actuations in business processes can be found in [19,23,38]. Further approaches use fine-grained actuations in processes in a disaster management scenario [25] and to control, e.g., an irrigation system [9] or a cooling system [44].

Actuation Pattern 2: Coarse-grained Actuation via Controllers

Context This pattern is eligible if/when feedback (control) loops are part of the actuation to ensure safety-critical behavior.

Solution Safety guides the choice of appropriate granularity levels of actuation in favor of more coarse-grained service tasks in the business process models. A service task in the business process only invokes the functionality in the CPS controller via its API and leave the rest to the CPS controller (cf. Fig. 11).

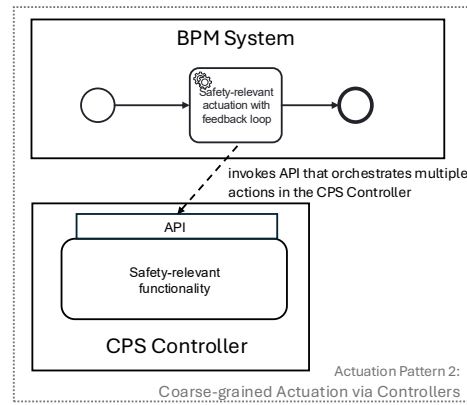


Fig. 11. The BPM system invokes the actuation in the CPS controller via its API at the business process level. The execution of the actuation is left to the CPS Controller.

Example An example of a safety-critical actuation task modeled at a coarse-grained level can be found in Fig. 2 with the “Move to pickup area” activity to just be triggered by the BPMS and executed by the system controlling the delivery robot. This ROS2-based CPS controller encapsulates all navigation aspects and safety-relevant functionality including obstacle avoidance using sensors in closed feedback loops as part of the long-running activity execution.

Known Uses We identified this pattern from working with different types of robots in our CPS laboratory. These robots perform complex continuous movements that involve the activation of different motors, control, and associated sensing. Their capabilities should be encapsulated and exposed at more coarse-grained levels [39], leaving the lower level actuations to the CPS controllers. The need for encapsulating safety-critical actuation in CPS that relies on internal feedback loops at a more coarse-grained level in business processes is discussed in [12,5]. The authors support our finding that the implementation of critical behavior should be left to lower level code in the CPS controller. Specific examples include the work in [14] for control of aerial drones, for anti-lock breaking systems of cars in [12], or for the control of an elevator in [20].

Decision Forces Resolution and Pattern Guidance. Most of the decision forces are shared between sensing and actuation in CPS when choosing appropriate granularity levels. Therefore, we refer to the discussions around Fig. 8 for Sensing for the common aspects and only highlight novel aspects that become more prominent for actuations. Nevertheless, in Fig. 12 we present a summary of how *all* the forces are addressed by the two actuation patterns. The first table entries are the ones already covered in Sensing (cf. Section 3.1). The last three are explained in text here. This analysis can be used to guide decisions regarding the choice of the actuation granularity levels by evaluating the decision forces resolution against project-specific non-functional requirements and constraints.

Ownership of CPS code and APIs The level of granularity to model actuation as service tasks calling external software systems is limited by the APIs offered by these external systems. This strongly influences the granularity at which actuation can be modeled in the business process. Very fine-grained modeling (Actuation Pattern 1) might be limited in this respect; more coarse-grained modeling can use or build upon existing APIs (Actuation Pattern 2).

Communication overhead The communication overhead is negatively affected by having fine-grained invocations of the CPS from the BPM system (Actuation Pattern 1) as it has to call the external API every time it executes the service task. A more coarse-grained actuation (Actuation Pattern 2) reduces these calls, but still requires communication between BPM system and the CPS.

Real-time and safety with feedback loops Many CPS must adhere to strict safety and real-time constraints to avoid causing harm to people and other entities.

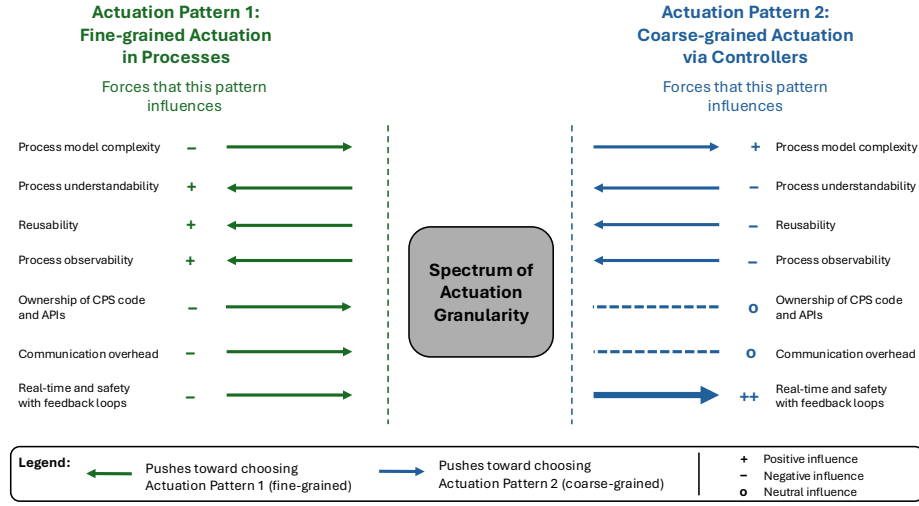


Fig. 12. Decision force field analysis for the two **actuation** granularity patterns.

Safety mechanism are implemented in closed control (feedback) loops that involve continuous actuation, control and sensing to maintain a desired operational state. If the feedback loop is closed inside the BPMN model, every iteration incurs a BPM engine round trip, which is an issue for high-frequency control as BPM systems are not designed for real-time control. If the CPS exhibits these constraints, then the actuations have to be modeled at a more coarse-grained level (Actuation Pattern 2). This moves the implementation of the actuation closer to the hardware and allows for verification outside the business process.

Related Patterns. Actuation Pattern 2 discussing actuation with feedback loops relies on continuous streams of sensor data emitted from within the CPS. This relates to Sensing Pattern 2 (cf. Section 3.1) where the processing of complex and high-frequency sensor streams is left to a transformation service—potentially part of the CPS Controller—outside the BPM system. Actuation Pattern 2 suggests an abstraction of functionality, which involves the modeling and invocation of complex, domain-specific logic combining sensing and actuation-related requests; hence, it can be seen as a *Facade* simplifying access to safety-critical functionality [13]. The two patterns *Closed-Loop* and *Open-Loop* process control in [6] correspond to our Actuation Pattern 2 and Actuation Pattern 1, respectively, but without contextualization in business processes and granularity concerns. The *Synthesize-Utilize* and *Synthesize-Command* patterns presented in [28] address feedback loop-based self-adaptive control of CPS, which can be part of the safety-relevant functionality in the CPS control software as described in Actuation Pattern 2. More generally, the invocation of actuation in CPS via an API call from the business process relates to the *Processing Resource* pattern [48]. The *AS2AS Flow-based Service Composition* and *AS2AS Service Or-*

chestration patterns in [43] address process-like service invocations in IoT. The work in [16] present patterns for process execution and integration design, also considering different granularity levels with *Microflow* and *Macroflow*.

4 Conclusion and Future Work

In this paper, we presented four patterns for process-driven CPS design and programming. When introducing BPM technologies to the realm of CPS to facilitate automation, decisions about abstraction and granularity of CPS functionality at the business process level become essential. In contrast to the forces influencing these granularity decisions in “traditional” BPM domains (e.g., enterprise applications), decision forces revolving around real-time, safety, data complexity and velocity, and closed control loops become more pronounced in the CPS domain. We presented two patterns focusing on *sensing* in CPS, namely *Fine-grained Sensing in Processes* and *Coarse-grained Sensing via Transformation Services*. Similarly, we discuss two patterns for *actuation* in CPS, *Fine-grained Actuation in Processes* and *Coarse-grained Actuation via Controllers*. Fig. 3 classifies these four patterns and shows their relations. Known uses and examples of the patterns can be found in different domains, including smart manufacturing, smart homes, and smart logistics.

Future work concerns the inclusion of the presented patterns in our emerging, formal model of architectural requirements and decisions, which we develop to support software architects with decision making in the new realm of CPS. In this paper, we focus on CPS-related aspects in business processes; in the future, we will also investigate further business process-related aspects that are relevant in CPS and influenced by the granularity choices (e.g., error handling, timeouts/retries, and compensations). More technical aspects to be elaborated include: 1) orchestrator deployment locations (cloud vs. edge BPM systems) influenced by the communication overhead resulting from different granularities; 2) the configuration of specific connectors to the CPS (e.g., MQTT, CoAP, HTTP) in relation with the granularity choices; and 3) process mining tasks influenced by the choice of functionality realized at the business process level.

Acknowledgments. This work has received funding from the Swiss National Science Foundation under Grant No. 10002384 (*TaSSAreCt* project). We want to thank our shepherd Uwe Zdun and all participants at the writer’s workshop at the EuroPLoP 2026 for their feedback and enhancements made to this paper.

References

1. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Van Kranenburg, R., Lange, S., Meissner, S.: Enabling Things to Talk: Designing IoT Solutions with TheIoT Architectural Reference Model. Springer Nature (2013)
2. Baumgrass, A., Di Ciccio, C., Dijkman, R.M., Hewelt, M., Mendling, J., Meyer, A., Pourmirza, S., Weske, M., Wong, T.Y.: Get controller and unicorn: Event-driven process execution and monitoring in logistics. In: BPM (Demos). pp. 75–79 (2015)

3. Bazan-Munoz, A., Pautasso, C., Ortiz, G., Garcia-de Prado, A.: BPSmart-CARE: a framework for managing contextualized actions in IoT systems through the integration of business process modelling and complex event processing. *Internet of Things* **36**, 101887 (2026)
4. Beerepoot, I., Di Ciccio, C., Reijers, H.A., Rinderle-Ma, S., Bandara, W., Burattin, A., Calvanese, D., Chen, T., Cohen, I., Depaire, B., et al.: The biggest business process management problems to solve before we die. *Computers in Industry* **146**, 103837 (2023)
5. Bi, Z., Zhang, C.W., Chi, J., Luo, C.: Internet of things (IoT) for sustainable mechatronic systems and cyber-physical systems (cps). In: 10th Int. Conference on Information and Network Technologies (ICINT). pp. 102–106. IEEE (2025)
6. Bloom, G., Alsulami, B., Nwafor, E., Bertolotti, I.C.: Design patterns for the industrial internet of things. In: 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS). pp. 1–10. IEEE (2018)
7. Bocciarelli, P., D’Ambrogio, A., Giglio, A., Paglia, E.: A bpmn extension for modeling cyber-physical-production-systems in the context of industry 4.0. In: 2017 IEEE 14th international conference on networking, sensing and control (ICNSC). pp. 599–604. IEEE (2017)
8. Chen, H.: Applications of cyber-physical system: A literature review. *Journal of Industrial Integration and Management* **02**, 1750012 (10 2017)
9. Domingos, D., Martins, F.: Using bpmn to model internet of things behavior within business process. *International Journal of Information Systems and Project Management* **5**(4), 39–51 (2017)
10. Fehling, C., Leymann, F., Retter, R., Schupeck, W., Arbitter, P.: Cloud computing patterns: fundamentals to design, build, and manage cloud applications. Springer (2014)
11. Fortino, G., Mecella, M.: Internet of Things Meets Business Process Management: A Synergistic Integration. Springer (2026)
12. Furrer, F.J.: Safe software and secure software. In: Safety and Security of Cyber-Physical Systems: Engineering dependable Software using Principle-based Development, pp. 187–262. Springer (2022)
13. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: Elements of reusable object oriented software (1995)
14. Graja, I., Kallel, S., Guermouche, N., Kacem, A.H.: Bpmn4cps: A bpmn extension for modeling cyber-physical systems. In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). pp. 152–157. IEEE (2016)
15. Hasić, F., Serral, E., Snoeck, M.: Comparing BPMN to BPMN + DMN for IoT process modelling: a case-based inquiry. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing. pp. 53–60 (2020)
16. Hentrich, C., Zdun, U.: A pattern language for process execution and integration design in service-oriented architectures. In: Transactions on Pattern Languages of Programming I, pp. 136–191. Springer (2009)
17. Hentrich, C., Zdun, U.: Process-Driven SOA - Proven Patterns for Business-IT Alignment. CRC Press, Taylor and Francis, Boca Raton (2012)
18. Hohpe, G., Woolf, B.: Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional (2004)
19. Kirikkayis, Y., Gallik, F., Seiger, R., Reichert, M.: Integrating IoT-driven events into business processes. In: International Conference on Advanced Information Systems Engineering. pp. 86–94. Springer (2023)

20. Klein, R., Xie, J., Usov, A.: Complex events and actions to control cyber-physical systems. In: Proceedings of the 5th ACM international conference on Distributed event-based system. pp. 29–38 (2011)
21. Lee, E.A.: Cyber physical systems: Design challenges. In: 2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC). pp. 363–369. IEEE (2008)
22. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics* **7**(66), eabm6074 (2022)
23. Malburg, L., Hoffmann, M., Bergmann, R.: Applying mape-k control loops for adaptive workflow management in smart factories. *Journal of Intelligent Information Systems* **61**(1), 83–111 (2023)
24. Mangler, J., Seiger, R., Benzin, J.V., Grüger, J., Kirikkayis, Y., Gallik, F., Malburg, L., Ehrendorfer, M., Bertrand, Y., Franceschetti, M., et al.: From internet of things data to business processes: Challenges and a framework. In: Internet of Things Meets Business Process Management: A Synergistic Integration, pp. 25–61. Springer (2026)
25. Marrella, A., Mecella, M., Sardina, S.: Intelligent process adaptation in the smartpm system. *ACM Transactions on Intelligent Systems and Technology (TIST)* **8**(2), 1–43 (2016)
26. Meyer, S., Ruppen, A., Magerkurth, C.: Internet of things-aware process modeling: integrating IoT devices as business process resources. In: International conference on advanced information systems engineering. pp. 84–98. Springer (2013)
27. Mottola, L., Picco, G.P., Oppermann, F.J., Eriksson, J., Finne, N., Fuchs, H., Gaglione, A., Karnouskos, S., Montero, P.M., Oertel, N., et al.: Make sense: simplifying the integration of wireless sensor networks into business processes. *IEEE Transactions on Software Engineering* **45**(6), 576–596 (2017)
28. Musil, A., Musil, J., Weyns, D., Bures, T., Muccini, H., Sharaf, M.: Patterns for self-adaptation in cyber-physical systems. In: Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects, pp. 331–368. Springer (2017)
29. Reinfurt, L., Breitenbücher, U., Falkenthal, M., Leymann, F., Riegg, A.: Internet of things patterns. In: Proceedings of the 21st European Conference on Pattern Languages of Programs. EuroPlop '16, Association for Computing Machinery, New York, NY, USA (2016)
30. Ruecker, B.: Practical Process Automation. " O'Reilly Media, Inc." (2021)
31. Schöning, S., Ackermann, L., Jablonski, S., Ermer, A.: IoT meets bpm: a bidirectional communication architecture for IoT-aware process execution. *Software and systems modeling* **19**(6), 1443–1459 (2020)
32. Schreiber, C., Abbad-Andaloussi, A., Weber, B.: On the cognitive and behavioral effects of abstraction and fragmentation in modularized process models. *Information Systems* **125**, 102424 (2024)
33. Seiger, R.: Towards flexible and adaptive human-robot collaborations via event-driven microservices. In: 2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA). pp. 1–4. IEEE (2025)
34. Seiger, R., Franceschetti, M., Abbad-Andaloussi, A.: A process to non-invasively augment legacy IoT systems using business processes and microservices. In: Proc. of the 14th International Conference on the Internet of Things. pp. 1–9 (2024)
35. Seiger, R., Huber, S., Heisig, P., Assmann, U.: Enabling self-adaptive workflows for cyber-physical systems. In: International Conference on Business Process Modeling, Development and Support. pp. 3–17. Springer (2016)

36. Seiger, R., Huber, S., Schlegel, T.: Toward an execution system for self-healing workflows in cyber-physical systems. *Software & Systems Modeling* **17**(2), 551–572 (2018)
37. Seiger, R., Kurz, A.F., Franceschetti, M.: Online detection of process activity executions from IoT sensors using generated event processing services. *Future Generation Computer Systems* **174**, 107987 (2026)
38. Seiger, R., Malburg, L., Weber, B., Bergmann, R.: Integrating process management and event processing in smart factories: A systems architecture and use cases. *Journal of Manufacturing systems* **63**, 575–592 (2022)
39. Seiger, R., Seidl, C., Akmann, U., Schlegel, T.: A capability-based framework for programming small domestic service robots. In: *Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-based Software-Engineering*. pp. 49–54 (2015)
40. Skouti, T., Seiger, R., Furrer, F.J., Strahringer, S.: RBPMN: the value of roles for business process modeling. *Software and Systems Modeling* **23**(6), 1375–1406 (2024)
41. Suri, K., Gaaloul, W., Cuccuru, A., Gerard, S.: 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 (2017)
42. Tabassum, M.R., Mustafiz, S.: Advances in requirements engineering for IoT and CPS: A survey on model-driven research and practices. In: *2025 IEEE/ACM 7th International Workshop on Software Engineering Research & Practices for the IoT (SERP4IoT)*. pp. 40–47 (2025)
43. Tkaczyk, R., Wasielewska, K., Ganzha, M., Paprzycki, M., Pawlowski, W., Szmaja, P., Fortino, G.: Cataloging design patterns for internet of things artifact integration. In: *Int. Conf. on Communications Workshops*. pp. 1–6. IEEE (2018)
44. Valderas, P., Torres, V., Serral, E.: Modelling and executing IoT-enhanced business processes through bpmn and microservices. *Journal of Systems and Software* **184**, 111139 (2022)
45. Yousfi, A., Hewelt, M., Bauer, C., Weske, M.: Toward uBPMN-based patterns for modeling ubiquitous business processes. *IEEE Transactions on Industrial Informatics* **14**(8), 3358–3367 (2017)
46. Zimmermann, O., Doubrovski, V., Grundler, J., Hogg, K.: Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned. In: *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. pp. 301–312 (2005)
47. Zimmermann, O., Luban, K., Stocker, M., Bernard, G.: Continuous process model refinement from business vision to event simulation and software automation: bridging gaps between stakeholder communities, practices, notations, and tools. In: *Proceedings of the 5th International Workshop on Software-Intensive Business: Towards Sustainable Software Business*. pp. 59–66 (2022)
48. Zimmermann, O., Lübke, D., Zdun, U., Pautasso, C., Stocker, M.: Interface responsibility patterns: Processing resources and operation responsibilities. In: *Proceedings of the European Conference on Pattern Languages of Programs 2020. EuroPLOP '20*, Association for Computing Machinery, New York, NY, USA (2020)
49. Zimmermann, O., Stocker, M., Lübke, D., Zdun, U., Pautasso, C.: *Patterns for API design: simplifying integration with loosely coupled message exchanges*. Addison-Wesley Professional (2022)