Journal of Manufacturing Systems 63 (2022) 575–592

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Integrating process management and event processing in smart factories: A systems architecture and use cases

Ronny Seiger [a,*], Lukas Malburg [b,c], Barbara Weber [a], Ralph Bergmann [b,c]

[a] Institute of Computer Science, University of St.Gallen, Rosenbergstrasse 30, St.Gallen 9000, Switzerland
[b] Business Information Systems II, University of Trier, Behringstraße 21, Trier 54296, Germany
[c] German Research Center for Artificial Intelligence (DFKI), Branch University of Trier, Behringstraße 21, Trier 54296, Germany

ARTICLE INFO

Keywords:
Industrial internet of things
Process management
Systems architecture
Event processing

ABSTRACT

The developments of new concepts for an increased digitization of manufacturing industries in the context of Industry 4.0 have brought about novel system architectures and frameworks for smart production systems. These range from generic frameworks for Industry 4.0 to domain-specific architectures for Industrial Internet of Things (IIoT). While most of the approaches include a service-based architecture for selective integration with enterprise systems, a close two-way integration of the production control systems and IIoT sensors and actuators with Process-Aware Information Systems (PAIS) on the management level for automation and mining of production processes is rarely discussed. This fusion of Business Process Management (BPM) with IIoT can be mutually beneficial for both research areas, but is still in its infancy. We propose a systems architecture for IIoT that shows how to integrate the low-level hardware components–sensors and actuators–of a smart factory with BPM systems. We discuss the software components and their interactions to address challenges of device encapsulation, integration of sensor events, and interaction with existing BPM systems. This integration is demonstrated within several use cases regarding process modeling, automation and mining for a smart factory model, showing benefits of using BPM technologies to analyze, control, and adapt discrete production processes in IIoT.

1. Introduction

Business processes are widely used for representing and managing high-level organizational and digital processes in different types of domains and enterprises [1]. In an industrial context, these business processes are usually supported by Process-Aware Information Systems (PAIS)–mostly Enterprise Resource Planning (ERP) systems–at the top business-related level (Level 4) of the well-known ANSI/ISA-95 automation pyramid (cf. Fig. 1) [2]. While sophisticated systems and applications have been developed for monitoring and controlling processes on the individual horizontal levels L0–L4 of the pyramid, the vertical communication and integration of these systems between the levels from sensor and actuator to service is still rather rudimentary [3]. Especially towards the higher levels, interactions between the highly customized Manufacturing Execution Systems (MES) and the ERP systems are rather infrequent and systems are more likely to work in isolation [4–7]. Workflow Management Systems (WfMS) as one form of PAIS dedicated to managing business processes show many benefits here with respect to process and application integration as well as high-level modeling, automation, and mining of production processes [8] but they are very rarely used to support these processes along the automation pyramid. Sensor data streams from production machines and environmental sensors carry important context information relevant for the process executions. However, this data is usually not considered in higher level (business) processes–especially not at runtime [9]. With the ongoing developments in the context of the Industrial Internet of Things (IIoT) a stronger bidirectional integration of the business process/workflow management, process control levels and sensors on the shop floor is envisioned [10], which will be beneficial both for the IIoT and the Business Process Management (BPM) domains [11]. On the one hand, real-time data from the IIoT devices can be fed directly into the WfMS and services (e.g., for production process monitoring and analysis) and, on the other hand, the WfMS can directly influence the execution of the processes (e.g., in case of machine failures detected from sensor data) [12].

The goal of this work is to achieve an integration of IIoT systems with

* Corresponding author.
E-mail addresses: ronny.seiger@unisg.ch (R. Seiger), malburgl@uni-trier.de (L. Malburg), barbara.weber@unisg.ch (B. Weber), bergmann@uni-trier.de (R. Bergmann).

https://doi.org/10.1016/j.jmsy.2022.05.012
Received 21 February 2022; Received in revised form 16 May 2022; Accepted 17 May 2022
Available online 27 May 2022

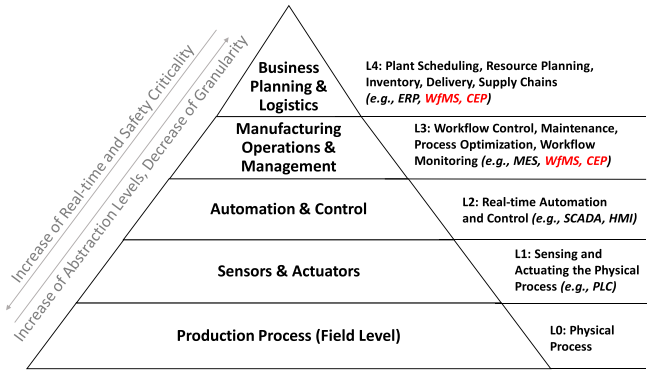


Fig. 1. ANSI/ISA-95 Automation Pyramid (adapted from [2]).

BPM technology along the BPM lifecycle including process modeling, process automation, and process mining. Following the ISO/IEC/IEEE 42010:2011 standard [13], we present a layered systems architecture as a reference framework, which aims at raising the abstraction level from the rather low-level machines and sensors of IIoT environments to a more business process-oriented viewpoint of the production control system and processes by enabling WfMS and stream processing platforms to be used at the top levels L3 and L4 of the pyramid as highlighted in Fig. 1. The architecture is based on abstraction and encapsulation of IIoT devices and their functionalities, which are integrated with WfMS and stream processing in a service-based way. The architecture combines components for actively invoking and orchestrating functionality from WfMS (*Process-oriented Architecture*), and for processing sensor data to enable monitoring and triggering of complex events on the business process level (*Event-driven Architecture*) [14]. We use a smart factory to showcase the implementation of the systems architecture and present several use cases to illustrate the benefits of integrating BPM with IIoT and using a process-oriented viewpoint for production control and analysis.

The paper is structured following the design science research methodology (DSRM) for information systems as proposed by Hevner et al. [15]: Section 2 introduces fundamental concepts (*Rigor–Knowledge Base*) and derives requirements from application scenarios (*Relevance–Environment*) that we address with our work. Section 3 discusses related work in the context of BPM and IIoT (*Rigor–Knowledge Base*). Section 4 presents the main artifact we developed–the systems architecture as framework and its components (*Develop/Build*). Section 5 showcases and discusses the BPM–IIoT integration from different viewpoints in several use cases within a smart factory model to show relevance and applicability of our framework in real-world production settings (*Justify/Evaluate*). Section 6 concludes the paper and shows starting points for future work.

2. Foundations & requirements for enabling BPM

The *Industrial Internet of Things (IIoT)* is "about connecting all the industrial assets, including machines and control systems, with the information systems and the business processes" to collect and analyze data with the goal of achieving optimized industrial operations [10]. In this context a *Smart Factory* acts as a "manufacturing solution that provides [.] flexible and adaptive production processes that will solve problems arising on a production facility with dynamic and rapidly changing boundary conditions" [16]. Typical hardware components of smart factories comprise sensors and actuators that are composed to more complex devices, machines, and stations of a production line [17]. Zuehlke coined the term "factory-of-things" [18], which fits nicely into our work as we assume that all sensors and actuators in a smart factory can be accessed and controlled via software and services–similar to other IoT systems [19].

BPM technologies can be **beneficial** in various application scenarios (AS) regarding the monitoring and control of IIoT environments (i.e., smart factories) and vice versa [11]: .

- AS1 To foster communication among stakeholders, production processes can be modeled and put into context of a company's business processes at an abstract level [6].
- AS2 The automation of these processes enables the seamless combination of automated tasks with manual steps (*human tasks*)–including exception and transaction handling–that are orchestrated by WfMS [17,20].
- AS3 BPM also facilitates the monitoring and mining of process executions within and across organizations in a more choreography style [1], thereby making the end-to-end workflow visible, which can be beneficial for IIoT environments as these are distributed systems and thus often lack a global state and history of executions [21].
- AS4 Being able to mine production processes and to put them into context of the corresponding business processes opens up opportunities for discovery, conformance checking, and optimization of a manufacturing company's processes [11,22] (cf. *Process Mining* [23]).
- AS5 In turn, data from IIoT devices can provide important context information for process executions in addition to data captured by the WfMS at runtime, and it can be used to trigger higher level events within processes and support decision making [14,24,8].

Despite these advantages, BPM technologies are rarely used in the context of smart factories. The **Goal** of our work is to establish a closer two-way link and integration between the low-level hardware components of a smart factory with the higher level information systems to manage production processes on a more abstract level via WfMS [11,12,6]. To be able to use BPM techniques and tools for process orchestration and analysis in IIoT, current Industry 3.0-style systems architectures have to be extended towards the integration of BPM technologies. In order to achieve the aforementioned goal, we address the following set of important Requirements that we encountered with creating the bidirectional link between BPM and IIoT for the application scenarios of controlling and monitoring a smart factory via PAIS (**AS1–AS5**). We base this set of requirements on relevant literature related to IIoT and on experiences with our own smart factory (*Environment*) following design science [15,25]:

- R1 *Encapsulation and Abstraction*: The hardware components in a smart factory may be accessible up to the level of individual motors and light barriers or switches on level L1. A production machine consists of a large number of these parts. When aiming at the integration with WfMS, considering the control of production machines on the level of these individual components–as e.g., proposed within the IoT reference model [19]–is not feasible as it would largely increase the complexity of processes regarding their modeling and automation, e.g., regarding real-time aspects which are usually not supported through WfMS [26]. Thus, encapsulation and abstraction of low-level machine components, data and functionality into higher level components and functions is required to support **AS1** and **AS2** [18,27,8].
- R2 *Remote Access*: The machines of a typical "Industry 3.0"-like production line execute low-level (e.g., G Code) programs that are locally deployed and controlled (e.g., via CNC) on levels L1 and L2 in isolation of other machines [5,28,29]. Thus, the interfaces to and interactions with other information systems and the *outside* world along the supply chains are rather few in number and, e.g., limited to selected interaction points between Manufacturing Execution System (MES) and ERP system [10]. The development of holistic smart factory control systems and their more sophisticated integration with PAIS/WfMS inherently

require open interfaces for remote access to production functionality and real-time data to support **AS2-AS5**. [17,18].

R3 *Handling of Concurrency*: With providing more open interfaces to access the smart factory's functionalities, there also emerges the necessity for handling concurrent access to the factory's physical resources [27]. The smart factory is a Cyber-Physical System (CPS) where resources are not easily scalable and operations are inherently concurrent and safety-critical, which is why concurrent and conflicting access from clients to the same CPS resources has to be handled to support **AS2** [30].

R4 *Integration of IIoT Data*: The increasing number of sensors and actuators of IIoT systems may act as new sources for large amounts of real-time context data regarding the smart factory and its environment. So far, specific sensors were mostly used in an isolated manner [31]–for local control loops within individual production machines. Data from these and additional sensors from the production environment should also be considered in combination on a more global process-oriented scale regarding the whole production line and supply chain on levels L3 and L4 [5,32]. Here, IIoT data can be used to enrich process execution information with additional context data and for triggering higher level business events to support **AS3-AS5**. This requires the production control systems to be extended and combined with the analysis of and reaction to complex business events from external sources [32–34,11].

3. Related Work

Related work with relevance for achieving the overall goal of integrating IIoT systems with PAIS/WfMS, and thereby addressing the requirements **R1-R4** can be categorized into two major aspects: 1) approaches regarding the combination of BPM and IIoT; and 2) system architectures for IIoT and CPS.

3.1. BPM and IIoT

Many architectures for CPS and IIoT do not explicitly address the integration with PAIS/(business) processes and only refer to an *application layer* on top of the hardware/software components (cf. Sect. 3.2). However, various approaches from the BPM community investigating aspects regarding the integration of BPM with IoT exist and are discussed in different surveys [35–38].

Chang et al. [35] present an overview of mobile cloud BPM systems for IoT. They suggest a service-based architecture as a middleware layer to communicate with the IoT devices via different protocols (**R2** ✓). WfMS are used as part of the management layer to control IoT devices at the edge. Baumgraß et al. discuss the integration of complex events from IoT with business processes in the context of smart logistics (**R4** ✓) [39]. Kammerer et al. put special focus on this integration of sensor events from industrial production machines and their processing in a BPM context [40] (**R4** ✓). A bidirectional communication architecture for IoT-aware process execution is presented by Schönig et al. [9]. Their work focuses on the data exchange and communication between IoT devices and WfMS at specific points. Thereby, they discuss the aspect of abstraction and encapsulation (**R1** **(✓)**) only concerning sensors and simple sensor events (**R4** **(✓)**). Koot et al. propose a reference architecture for IoT-enabled dynamic planning in the domain of smart logistics [41]. Their architecture consists of three layers *Technology*, *Application*, and *Business*, with software components for interacting with the IoT environment, for planning and optimization, and for handling of business level events (**R4** ✓). In the context of smart homes, Seiger et al. present the *PROtEUS* system to execute self-adaptive cyber-physical workflows in IoT environments, which assumes that services used to control IoT devices and to collect data for event processing exist [42,43] (**R2** ✓, **R4** ✓). Similarly, the *SmartPM* approach [44] supports adaptive processes based on a layered architecture consisting of Design, Enactment, Adaptation, Service and Cyber-Physical Layer. Based on this architecture, the authors in [45] describe an approach towards adaptive processes using semantic web services to represent and encapsulate functionality of a smart factory (**R1** ✓, **R2** ✓) as well as to handle concurrent access to resources (**R3** ✓). The architecture of the *SitOPT* system for situation-aware adaptive workflows in manufacturing is presented in [46]. Here complex failure situations can be recognized based on defined rules and compensating workflow templates are chosen and executed (**R4** ✓). In [47] the authors discuss an approach to model and enact end-to-end printing processes relying on WfMS on top of a middleware layer. This work has been extended to the *HORSE* framework as a comprehensive approach and reference architecture for CPS in manufacturing [48]. The authors discuss many relevant architectural aspects but do not go into details regarding encapsulation and abstraction of devices (**R1** **(✓)**) or sensor event integration (**R4** ×). Bordel Sánchez et al. [49] present an architecture to control processes in Industry 4.0 scenarios. The architecture consists of 8 layers starting from the Physical System layer up to Global Model and Decision-Making, and Domain Expert Environment layer. They also suggest a dedicated *Data Analytics* layer for sensor data processing (**R4** ✓). Valderas et al. discuss in [50] an approach for modeling and executing IoT-enhanced business process based on a microservice architecture. They also show how to integrate high-level events from IoT context using complex event processing (CEP) (**R4** ✓). All in all, for most approaches it remains unclear how the actual encapsulation and abstraction of devices is realized (**R1**), how concurrency is handled (**R3**), and how complex events are integrated together with the orchestration of processes (**R4**). A comparison of the aforementioned related work on BPM and IIoT regarding requirements fulfillment can be found in Table 1.

3.2. Systems architectures for IIoT

With the advancements in the development of IIoT systems in recent years, a significant number of system and software architectures for IIoT and CPS have been proposed. A survey on IoT architectures is presented in [51], on architectural styles for IoT in [52], and on Industry 4.0 reference architectures in [53]. An interesting catalog of architectural decision for designing IIoT systems with high importance for our work is proposed in [54]. In the following, we discuss selected concrete approaches relevant to our work.

Layered architectures are among the most widely implemented system architectures for IoT [52]. The *Industrial Internet Reference Architecture* (IIRA) and the *Reference Architecture Model for Industry 4.0* (RAMI 4.0) are general frameworks for architectures in the context of IIoT [28,55]. They both integrate aspects from several high-level viewpoints but can be simplified to six main layers: Business Layer, Functional Layer, Information Layer, Communication Layer, Integration Layer, and Asset Layer. Our work is addressing most of these layers and providing more implementation details with the goal of achieving a vertical integration of existing BPM technologies to interact with the assets of a smart

Table 1
Comparison of Requirements addressed by Related Work on BPM and IIoT. (✓=addressed, (✓)= partially addressed, ×= not addressed).

Rel. Work	R1	R2	R3	R4	Sys. Arch.
[35]	×	✓	×	×	×
[39]	×	×	×	✓	(✓)
[40]	×	×	×	✓	(✓)
[9]	(✓)	✓	×	✓	(✓)
[41]	×	(✓)	×	✓	(✓)
[42]	×	✓	×	✓	(✓)
[44]	×	✓	×	×	(✓)
[45]	✓	✓	✓	×	(✓)
[46]	×	×	×	✓	(✓)
[48]	(✓)	✓	×	×	(✓)
[49]	×	✓	×	✓	(✓)
[50]	×	✓	×	✓	(✓)

factory.

An architecture consisting of the 5 C layers *Configuration*, *Conversion*, *Cyber*, *Cognition*, and *Configure* is proposed by Lee et al. [56,57]. The architecture is rather abstract and it remains unclear how the information systems located at higher levels (e.g., ERP systems, MES, and WfMS) are connected to the devices of the shop floor (**R2 (✓)**), and how the functionality and data of machines can be integrated at reasonable abstraction levels (**R1 ×**, **R4 (✓)**). As in almost all available architectures for IIoT and CPS, this architecture relies on services for enabling remote access to the software components via external applications [17,18] (**R2 ✓**). An extension of the 5 C architecture with three additional layers including *Customer*, *Content* and *Coalition* is proposed in [58]. It provides more details on connectivity and means for interactive monitoring of machines but it does not cover additional requirements. In [14], the authors present a real-time event-based platform for developing digital twin applications. They focus on the event-driven architectural style and discuss event processing and event-driven services for digital twins (**R4 ✓**). An IIoT platform for context-aware information services is presented in [59]. Its architecture consists of five layers *Asset*, *Integration*, *Context & Information*, *Services*, and *Business* and is also an event-driven and context-aware approach for manufacturing based on a multi-layered, service-oriented architecture (**R4 ✓**). The authors provide a detailed description of integrating sensors but not actuators (**R1 (✓)**).

Kuhn et al. present in [60] a service-based production ecosystem architecture of Industry 4.0. The authors discuss the abstraction and integration of production functionality and device capabilities in detail (**R1 ✓**). These capabilities are used for retrieving suitable devices and an asset administration shell is provided for accessing actuators and sensors. The prototypical implementation of the architecture is based on the software components from the BaSys 4.0 framework [61]. In [62] the authors describe a high-level architecture of a smart factory including a case study for an automation cell showing a partial implementation using OPC-UA [63]. This architecture does not provide details on the abstraction of machine functionality nor does it relate to BPM aspects. Similarly, Luo et al. present in[64] an OPC-UA-based smart manufacturing system architecture that does not provide many implementation details regarding individual machines or data integration. A system architecture for Industry 4.0 applications with a focus on data integration on the lower levels via services is discussed in [65] (**R4 ✓**). Although there is no PAIS involved in the data integration and active control, the authors point out the importance of a component for service orchestration, which could be achieved by WfMS. Hehenberger et al. present in [17] an overview and approach of methods and applications for CPS. Among others the authors discuss the design, modeling, simulation, and integration of CPS partially addressing aspects regarding abstraction and encapsulation of sensors and actuators (**R1 (✓)**) as well as resilient data processing (**R4 ✓**). The work presented in [28] describes an architecture for Cyber-Physical Production Systems (CPPS) integrating the RAMI 4.0 framework and design patterns for multi-agent systems. Among four identified patterns, there is a *Resource Agent* representing the hardware and software components on L0–L2 of the ISA-95 pyramid (**R1 ✓**). The authors also note the usefulness of an orchestrator for scheduling, coordination, and monitoring of production processes.

A systematic discussion and classification of self-adaptive IoT architectures can be found in [66]. The authors discuss different types of layering and distribution of hardware and software components. In the scope of our work, we assume the IIoT devices and data sources to be distributed; data processing and service orchestration on the application level can be centralized or distributed (**R2 ✓**, **R4 ✓**). A specific example of a software architecture for IoT for the case of a building emergency evacuation is proposed in [67]. Here the authors discuss the self-optimization of people flows and building capacities, which could also be interesting for handling concurrency in the smart factory (**R3 ✓**).

Other works focus on different aspects regarding the use of component-based principles and models for representing resources in architectures for CPS [68] (**R1 ✓**), or on the abstraction and encapsulation of IIoT devices as domain objects and their integration as context providers [69] (**R1 ✓**, **R4 ✓**). Of special interest for the processing of external events (**R4 ✓**) is the work proposed in [32], which presents a framework for the monitoring of systems of systems–typical CPPS–on different abstraction levels, and providing means for specifying event–condition–action rules to react to higher level (business) events. An agent-based framework for workflow-based actuation and time-critical data processing in IIoT is proposed in [70]. Here the authors also discuss aspects regarding task prioritization and scheduling, which is very relevant for handling concurrency (**R3 ✓**). A systems engineering tool-chain for automated parameter value selection to facilitate verification in CPS simulations is presented in [71]. Here the authors partially rely on business process and workflow technologies to model and automate tasks related to verification. A comparison of the aforementioned related work on IIoT systems architectures regarding requirements fulfillment can be found in Table 2.

3.3. Summary and research gap

As shown in Table 1 the presented approaches discuss selected aspects related to the combination of IIoT with BPM but they only partially address the necessary software components, system architecture, and development steps in the form of a framework to enable the rather low-level IIoT devices to interact with WfMS–and vice versa (**R1–R4**). They all present high-level architectures to a certain level of detail but assume that the IIoT hardware components are already wrapped as services in a distributed system, i.e., only addressing remote access to machine functionality (**R2**). They usually do not show how to achieve this abstraction and encapsulation of IIoT devices as services (**R1**). They do not discuss the actuators in an IIoT environment and with that the handling of concurrency (**R3**), but they rather only focus on sensors. In contrast to using business processes to only specify and support manufacturing operations as shown in [4,7], we are aiming at using processes along the entire BPM lifecycle [72] to also actively control (automate) the operational production processes, to mine them, and to react to specific events derived from IoT data. In contrast to work discussing aspects regarding information systems integration and enterprise architecture [6], we focus on the detailed descriptions of software components and their interactions as part of a systems architecture framework for IIoT-enabled BPM. Moreover, we show a concrete instantiation of this architecture for a smart factory applied to real-world use cases.

The selected works on IIoT system architectures either present rather general and abstract frameworks or very specific implementations of systems architectures, only discussing a subset of the requirements regarding the integration of IIoT with BPM technology as shown in

Table 2
Comparison of Requirements addressed by Related Work on IIoT Systems Architectures. (✓=addressed, (✓)= partially addressed, ×= not addressed).

Rel. Work	R1	R2	R3	R4	BPM
[56]	×	(✓)	×	(✓)	×
[58]	×	✓	×	×	×
[14]	×	✓	×	✓	×
[59]	(✓)	✓	×	✓	×
[60]	✓	✓	×	(✓)	×
[62]	×	✓	×	×	×
[64]	×	✓	×	×	×
[65]	×	✓	×	✓	×
[17]	(✓)	✓	×	✓	×
[28]	✓	✓	×	×	×
[67]	×	(✓)	✓	×	×
[68]	✓	✓	×	×	×
[69]	✓	✓	×	✓	×
[32]	×	✓	×	✓	×
[70]	×	✓	✓	✓	×

Table 2. Here we also see various works addressing the integration of sensors, but not actuators and with that no discussion of handling concurrency (**R3**). Despite several works pointing out the need and close connections with WfMS for service orchestration and monitoring, none of the works explicitly address BPM-related aspects and how to integrate IIoT with WfMS. The goal of our work is to achieve this integration and discuss what is necessary to go from the IIoT devices on the shop floor to the level of WfMS to benefit from the work of the BPM domain [11].

All in all, we have identified a research gap when it comes to describing the necessary steps and software components in the form of an architectural framework to integrate the hardware components–both sensors and actuators–of IIoT environments with BPM systems not only to model processes, but also to automate and monitor processes in IIoT along the entire BPM lifecycle [72]. On the one hand, existing IIoT systems architectures do not explicitly address aspects related to BPM. On the other hand, existing approaches discussing the BPM–IIoT integration do not fully cover aspects and requirements regarding systems architectures and necessary development steps such as the abstraction of IIoT devices or the handling of concurrency. Additionally, they often only refer to the integration and communication of *sensors* as a subset of IIoT components with BPM systems.

4. Systems architecture for a smart factory

To address the goal of achieving an integration of BPM with IIoT and the requirements identified in Sect. 2, we designed and developed (following the design science research methodology [15,25] and ISO/-IEC/IEEE 42010:2011 [13]) a layered architecture for smart factories with a focus on the individual software components and their interactions. In this section we provide a description of the architecture as general framework from an integration viewpoint and elaborate on exemplary software components for implementation of the architecture in an IIoT setting. The layered architecture and software components as well as their connectors are depicted in Fig. 2. From bottom to top, the levels of abstraction and modularization of components increase. Communication is only allowed between two adjacent layers. With the individual layers, namely the *Hardware Layer*, *Control Layer*, *Domain Layer*, *Service Layer*, and *Process Layer*, we aim at separating concerns and responsibilities w.r.t. functionalities and typical hardware–software topologies in IIoT [52,51,18]. The layers are meant to be generic for IIoT and are partially following the ANSI/ISA-95 pyramid [2,7] but with a stronger focus on software components. When describing exemplary software components, we refer to our own smart factory setup and implementation (cf. Sect. 4.1).

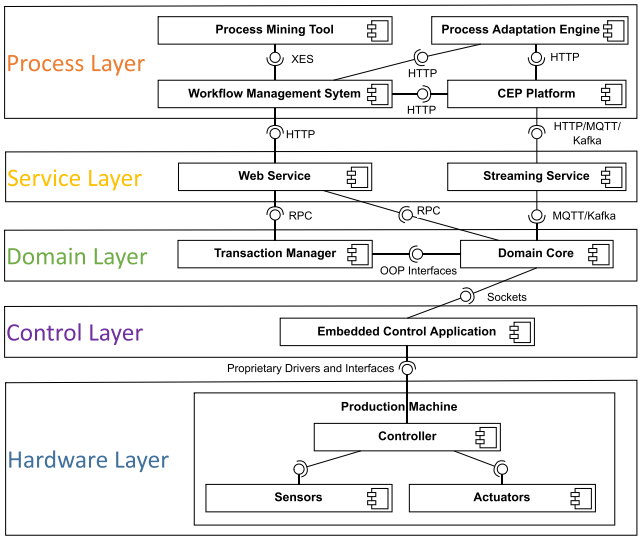


Fig. 2. Layers and components of the smart factory systems architecture.

4.1. Smart factory model

In this research we use a simulation model of a smart factory as physical structure that consists of hardware components provided by Fischertechnik [12]. This smart factory simulates a complex real-world production line spanning one or two shop floors that consist of various production stations, e.g., ovens (OV), milling machines (MM), punching machines (PM) and drilling machines (DM), high-bay warehouses (HBW), sorting machines (SM), vacuum gripper robots (VGR), and human workstations (HW), as shown in Fig. 3. Each of these stations is composed of numerous sensors, motors, switches, valves, LEDs, and compressors that are controlled by embedded computers (*TXT controllers*) similar to Programmable Logic Controllers (PLC) [5]. These controllers are connected to a standard computer network via Ethernet or WiFi. The factory simulates discrete manufacturing processes featuring the production of generic workpieces in the form of small cylindrical blocks of different colors.

This simulation model is a very suitable and rather inexpensive testbed for our research prototypes regarding the combination of BPM and IIoT technologies in a lab environment [12]. It represents a typical CPS of interconnected stations in the sense of *Machine Tool 4.0* [73,74] with feedback loops between computing and physical processes [30]. However, being a small-scale simulation model of a smart factory, the model also exhibits behavior and properties different from a real-world smart factory, e.g., imposing less strict real-time and safety requirements and having less reliable hardware components [12,75]. We elaborate on these limitations and trade-offs in Sect. 5.

4.2. Hardware layer

Starting from the bottom, the *Hardware Layer* contains all physical hardware components–abstracted as *sensors* and *actuators*–belonging to the production machines of the IIoT environment as physical structure [17]. These components are usually connected to and controlled by *controllers* (e.g., PLCs) that offer software-based interfaces via proprietary drivers. Through the controllers the production machine's components can be accessed to some extent and at varying levels of granularity from high-level production functionality down to individual sensors and actuators by other software processes. Sensors produce data coming from the real production system and its surroundings, actuators change the state of the physical world by executing physical actions [30].

Examples. Typical component instances on the hardware layer of our smart factory are shown in Fig. 8: motors, switches, LEDs belonging to

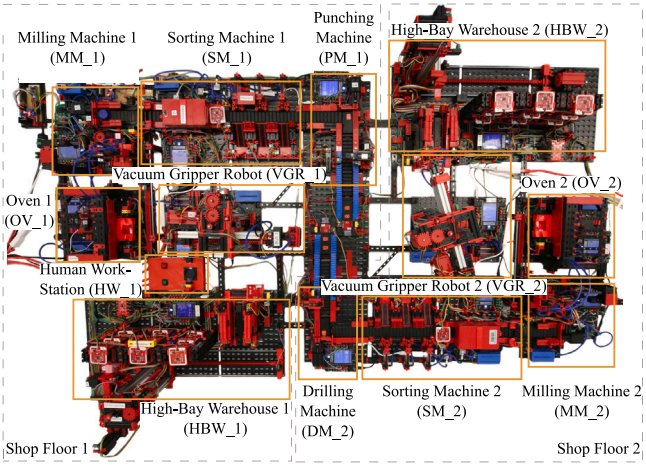


Fig. 3. Smart Factory Model used in this Work.

machines of the individual production stations, e.g., *Oven 1* (cf. Fig. 3). The controllers that these components are connected to are small embedded computers (*TXT controllers*).

4.3. Control layer

The *Control Layer* controls the components of the hardware layer via software on a very fine-grained level. This is usually realized by distributed *Embedded Control Applications* running on the controllers that use the interfaces of the hardware components via proprietary protocols, interfaces and drivers [5]. The control applications are hand-coded with proprietary, low-level libraries and machine code that are specific to the devices and controllers [29]. The Hardware Layer and Control Layer comprise the existing Layers L0–L2 from the ANSI/ISA-95 pyramid [2].

Implementation and examples. In our smart factory, the embedded control applications–realized via the *ftrobopy* module[1]–offer functions to, e.g., start/stop motors, check limit switches, or activate LEDs (cf. Fig. 8), using the Python programming language. They are distributed and provide interfaces for communication with other components and software processes (e.g., via TCP sockets).

4.4. Domain layer

Despite having the possibility of accessing each sensor and actuator via the control layer, we aim at the integration of the IIoT systems and their functionality with rather high-level processes in WfMS, which requires abstraction and encapsulation in order to enable reuse (**R1**). In the *Domain Layer*, this fine-grained access to low-level functionality of the IIoT devices via the control layer is encapsulated, modularized, and abstracted into higher level domain-specific, reusable entities based on principles from object-oriented programming (OOP) and domain-driven design (DDD) (**R1**) similar to the approach in [69].

4.4.1. Domain core

The domain layer is designed following the idea of having an isolated *Domain Core* within hexagonal architectures [76] that only contains core domain/business functionality without dependencies on additional external services or technologies. Here the concrete levels of abstraction and encapsulation for *domain entities*, *value objects* and *aggregates* [76] strongly depend on domain and use case specific requirements as well as properties of the IIoT environment (cf. Sect. 5.2). The information model of the IIoT environment (cf. *Machine Tool 4.0* [74]) thereby serves as basis for designing the domain core. We decided to have a domain-partitioned core, i.e., the stations of the production line as main domain classes with methods referring to higher level production-related business functionality [12,45,77]. These domain classes are the *aggregate roots* in the sense of DDD [76] within the domain core.

Implementation. The domain core is implemented using the Python programming language with the aggregate roots as individual modules for our smart factory. We assume the domain core to be instantiated for one physical instance of a smart factory, i.e., all the stations for one physical model are instantiated as objects (*process resources*) within the corresponding Python application. The UML class diagram in Fig. 4 shows the aggregate root classes for the different types of production stations in our smart factory (cf. Fig. 3) including their methods and input parameters that represent high-level production-related (business) functionality. Common functionality is imposed upon all aggregate roots via the *ProductionStation* interface. This interface includes methods to connect and disconnect from the Domain Layer to the controllers on the Control Layer, which are highly protocol and device specific and should be implemented by adapter classes outside the domain core following hexagonal architecture [76]. The interface also features methods for reading sensor values from individual machines and for the streaming of data, which can again be rather technology-specific and should therefore be implemented by an adapter class outside the domain core. Referring to the smart factory setup depicted in Fig. 3, we would instantiate for example two objects of the *Oven* class, two objects of the *SortingMachine* class, and one object of the *PunchingMachine* class.

4.4.2. Transaction manager

In the domain layer we find a clear separation of "reading" queries (i.e., retrieving sensor data) and "writing" (i.e., manipulation) commands. Reading data, i.e., sensor events, from the production stations is possible directly via the methods referring to reading sensors and streaming data of the aggregates (cf. *ProductionStation* interface in Fig. 4) as these are non-critical operations that require high throughput (**R4**). In contrast, invoking *active* domain-specific functionality of IIoT devices for actuation in the physical world is only possible via a dedicated *Transaction Manager* component for each aggregate root in accordance with **R3**. These transaction managers are needed to ensure that during the execution of a physical action, no conflicting action can be executed in parallel at one production station [45]. Fig. 4 shows the transaction manager classes acting as a proxies to clients calling the individual production station's functionality and delegating it to the actual production station instance (i.e., aggregate root). Figs. 8 and 9 show these invocation for the example of one instance of the *Oven* production station.

Implementation. In our implementation we rely on simple transaction mechanisms and queues that store and process requests based on their timestamp and optional starting time as well as priority attributes to realize parts of the concurrency handling [45]. Fig. 8 and Fig. 9 show that calls to active domain core functionality always go through a transaction manager, which then invokes the methods of the domain entities. The transaction mechanisms and queues provide simple means to limit concurrent access via sequential, non-interruptable calls to the aggregate roots (**R3**). We are currently working on integrating more sophisticated approaches for planning and task scheduling to prevent conflicts and deadlocks [78]. The handling of concurrency and safety-critical aspects with real-time requirements is usually implemented within the PLC system on Level L1 of the ANSI/ISA-95 pyramid (the *Control Layer* in our proposed architecture) to control the machines within real-world production settings. In our simplified smart factory model we do not have strict real-time and safety requirements, which is why we rely on the aforementioned, more sophisticated but slower means of handling concurrency (**R3**) in the domain layer.

4.5. Service layer

We propose to use Web services in a dedicated *Service Layer* to enable remote access to the domain-specific functionality of the domain layer (**R2**) [17,18].

4.5.1. Web service

The methods (API) of the domain entities, i.e., production stations (cf. Fig. 4), are wrapped and exposed one-to-one via a RESTful *Web Service* application as web resources. The size of the web service corresponds to the entities of the domain core, i.e., one web service enables access to one physical instance of one smart factory as an independently deployable component (*Architectural Quantum* [79]). Due to the physically limited resources and lack of scalability as driving architectural characteristics in CPS [30,79], a smaller service size towards microservices is not feasible, which is why we have decided for the service-based architectural style with one service per physical instance of the smart factory [79]. Each call from the service layer to invoke actuation functionality is handled by a proxy method of the corresponding transaction manager that delegates the call to the aggregate roots in the core of the domain layer for the physical instances of the factory's production stations.

[1] https://github.com/ftrobopy/ftrobopy

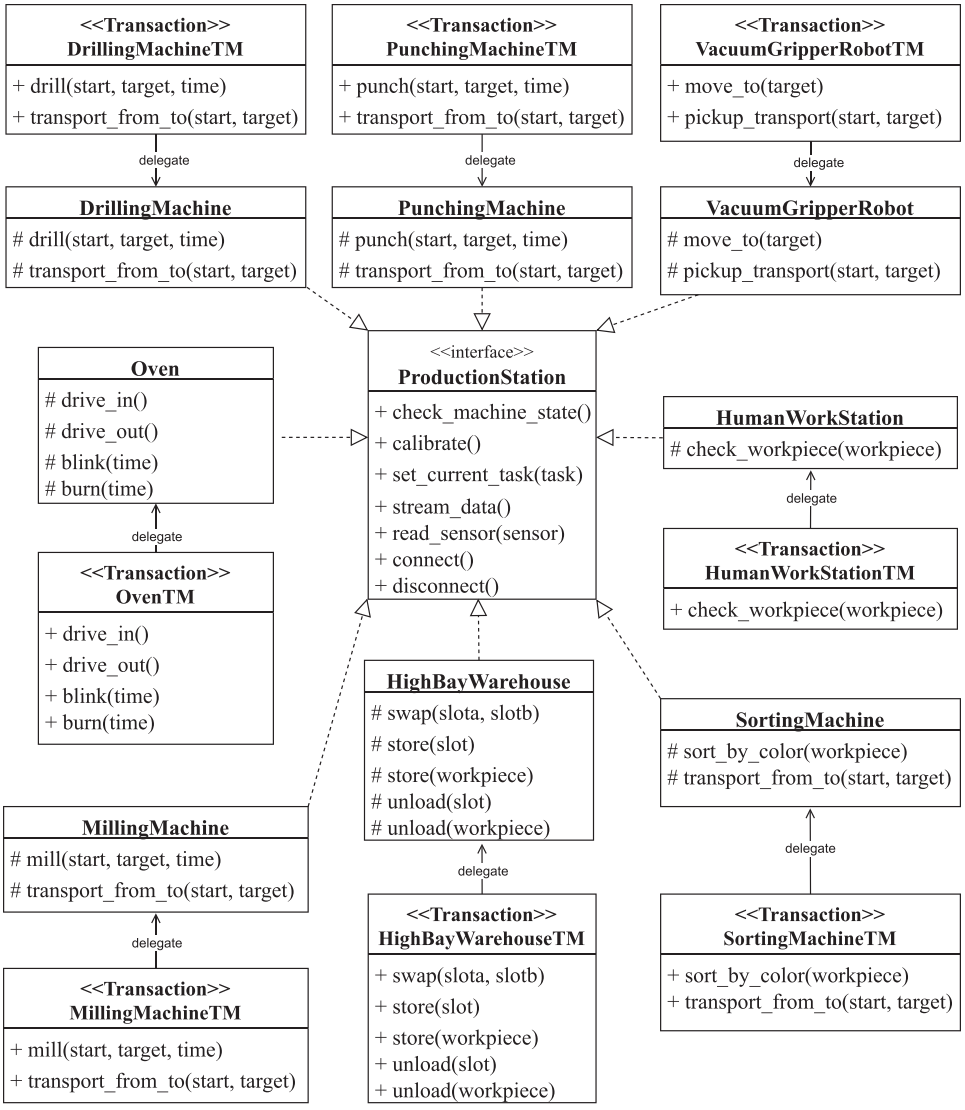


Fig. 4. Excerpt from the class diagram of the domain layer.

Implementation. To be consistent with the separation of commands and queries realized in the domain core and to follow standard naming conventions for the design of the web service API, we propose to use the *HTTP GET* method for queries to retrieve sensor and status data from the production stations (*read-sensor*, *check-machine-state*), and to use the *HTTP PUT* method for commands to invoke active production machine functionality via the transaction managers in the domain core. Necessary input data is provided as payload as part of the requests' body. Here, the Uniform Resources Identifiers (URIs) follow the naming scheme (derived from the domain classes and their instances, cf. Fig. 3 and Fig. 4):

baseuri/station-instancenumber/production-method

Examples are:

- */baseuri/ov-1/burn*
- */baseuri/hw-1/check-workpiece*
- */baseuri/mm-2/transport-from-to*

In our implementation, the web service communicates with the Transaction Managers on the domain layer via synchronous Remote Procedure Calls (RPCs) for *actuation requests*. We use synchronous RPC here to ensure that the execution of a service request is not interrupted by other actuation requests to the same resources (**R3**). Queues are used to avoid loosing incoming requests during this time [45]. The web service can also communicate directly with the Domain Core via RPC for actively retrieving sensor data (*read requests*). In our smart factory, the RESTful web service is implemented using the Python *Flask*[2] framework. Fig. 8 and Fig. 9 show the resources on the service layer that map to the production stations and their domain-specific methods. On the service layer, we have the option to use semantic annotations to check whether the preconditions for executing service requests are met and whether the service execution has produced the desired outcome as further described in [45,80]. These annotations can also be used for automatic service discovery and composition as well as for AI-based planning [45]. We are currently working on making the service implementations compatible with the OPC-UA information model [63] to improve communication among components and assets [3].

4.5.2. Streaming service

The service layer also incorporates a *Streaming Service* to enable message-based read access to more fine-grained real-time sensor (event) data streams from the IIoT environment (**R4**). Here we propose to use asynchronous publish-subscribe communication [14] to emit events

[2] https://palletsprojects.com/p/flask/

from objects in the domain core without the need for going through the transaction manager.

4.5.2.1. Implementation. In our setup, we use Apache Kafka[3] and an MQTT broker[4] as two alternative streaming platforms that provide application layer interfaces. We publish the states of all sensors and actuators of the production machines, of environmental sensors and cameras, and higher level events (e.g., currently executing production tasks) via the streaming services in customizable intervals.

4.6. Process layer

The main goal of our work is the integration of IIoT systems with BPM technology along the BPM lifecycle [72] regarding modeling, automation, and mining of processes in IIoT [11] (cf. **AS1-AS5** in Sect. 2). Hence, on top of the service layer, we add a novel *Process Layer* with applications dedicated to (business) process management and processing of complex (business) events (**R4**). On the process layer, we find: a *Workflow Management System* (WfMS) for process management, modeling and automation; tools for process monitoring and mining; a *CEP Platform* for stream/complex event processing; and a *Process Adaptation Engine* to react to exceptions occurring during process execution as further described in [12,43,81]. The workflow among the main BPM components is depicted in Fig. 5 and will be described in the following sections.

4.6.1. Workflow management system

The WfMS is responsible for aspects related to BPM including the support of the process designers with modeling of high-level operational production processes combined with human and organizational activities and aspects in a dedicated process modeling tool as shown in Fig. 5 (**AS1**).

The WfMS also makes it possible to automate these processes by creating process instances from the modeled production processes and then orchestrating the execution of modeled activities per process instance. These activities may include requests to the resources of the web service on the service layer via dedicated service calls (*service/external tasks*) that are part of the modeled processes (**AS2**). Using a WfMS for orchestration here provides many additional benefits such as built-in exception and distributed transaction handling and handling of state. Moreover, the monitoring of process executions among the distributed IIoT components and software systems in a more choreography style is possible to make the often hidden end-to-end workflows visible [1] (**AS3**). The WfMS keeps a record of all events related to the execution of process instances (e.g., start and finish of a process instance or an activity instance) in an *Event Log* (cf. Fig. 5).

In case more than one web service is involved in the control of the IIoT environment (i.e., other IIoT devices interact with the smart factory or multiple smart factories with each other) and conflicting or erroneous interactions may occur across web services, the concurrency and distributed transaction handling is also within the responsibilities of the WfMS as specified by the process designers in the respective process models. This is in contrast to the transaction managers on the domain layer, which are responsible for handling errors and limiting access to the physical resources of the smart factory (**R3**).

Implementation. In our setup, we use the *Camunda BPM Platform*[5] as WfMS. Fig. 6 shows a screenshot of the corresponding process modeling tool Camunda Modeler[6] with a process in BPMN 2.0 for our smart factory consisting of a start and end event and one production activity "OV_1_Burn" as *Service Task*. On the right side the configuration of this task is shown. It is configured to execute an HTTP PUT request to the web service on the service layer specifically addressing the "/ov-1/burn" resource and passing the production time as request payload (cf. *Input Parameters* section in Fig. 6). Although the process engineer is free to choose the labels of activities and events in the process model, we suggest to follow the naming convention *Station_Instancenumber_Production_Method* for defining the labels of automated service tasks that interact with the smart factory. This way we can automatically generate pre-configured activities in BPMN 2.0 to be used for modeling derived from the domain core and web service resources. To support the process engineer with these configurations, we have developed templates for the Camunda Modeler that can be easily (re-) used and configured in a more *low-code* fashion. Figs. 8 and 9 show exemplary process models in BPMN 2.0 on the process layer, which contain service tasks as activities referring to the specific functionality of the domain entities exposed via the web service.

4.6.2. Process mining tool

The *Process Mining Tool* receives the output from the WfMS–usually an *Event Log* in XES (eXtensible Event Stream) format [82]–that can be used for process mining tasks supported by the tool as shown in Fig. 5 (**AS3, AS4**). In our architecture process mining can also be done on process event streams from the WfMS [83] or even from the CEP platform published via the streaming services [84].

Implementation. As we assume that the WfMS is producing an event log following the XES standard, an arbitrary process mining tool supporting this standard can be chosen for process analysis tasks. In our tool-chain we use *Fluxicon Disco*[7] (cf. Sect. 5.4).

4.6.3. CEP platform

The CEP platform processes and evaluates events from the IIoT environment and additional external sources received via the streaming services on the service layer in order to derive higher level business events and context data with relevance for the process execution (**AS5, R4**). In case of exceptions during production that may require interventions or adaptations of the process execution, the *Process Adaptation Engine* is invoked by the CEP platform, it retrieves data from the WfMS about the state of the process, and then adapts the process according to the type of exception and state as described in more detail in [12,45,43,34].

Implementation. We use *Siddhi*[8] as standalone CEP platform. Both main components on the process layer, the WfMS and CEP platform, interact with each other via HTTP requests. An exemplary use case illustrating the interactions between the WfMS and CEP platform during execution of a manufacturing process for quality control and deriving high-level business events is described in Section 5.3.

5. Use cases: BPM in a smart factory

To demonstrate and discuss the developed systems architecture framework following the DSRM (Justify/Evaluate phase, cf. Sect. 1) [15,25], we implemented a vertical prototype of the software architecture using the software components described in Section 4 for two different configurations of the smart factory described in Sect. 4.1. One configuration basically consists of all the 6 stations belonging to *Shop Floor 2* (cf. Fig. 3) and the other configuration consists of both shop floors with 12 stations as depicted in Fig. 3. The effort of implementing and instantiating the whole systems architecture for the two different configurations was very low since only the objects to be instantiated from the classes (aggregate roots) in the domain core (cf. class diagram in Fig. 4) on the domain layer had to be adjusted with respect to the number and different types of available production stations. For

[3] https://kafka.apache.org/
[4] https://mqtt.org/
[5] https://camunda.com/
[6] https://camunda.com/products/camunda-platform/modeler
[7] https://fluxicon.com/disco/
[8] https://siddhi.io/

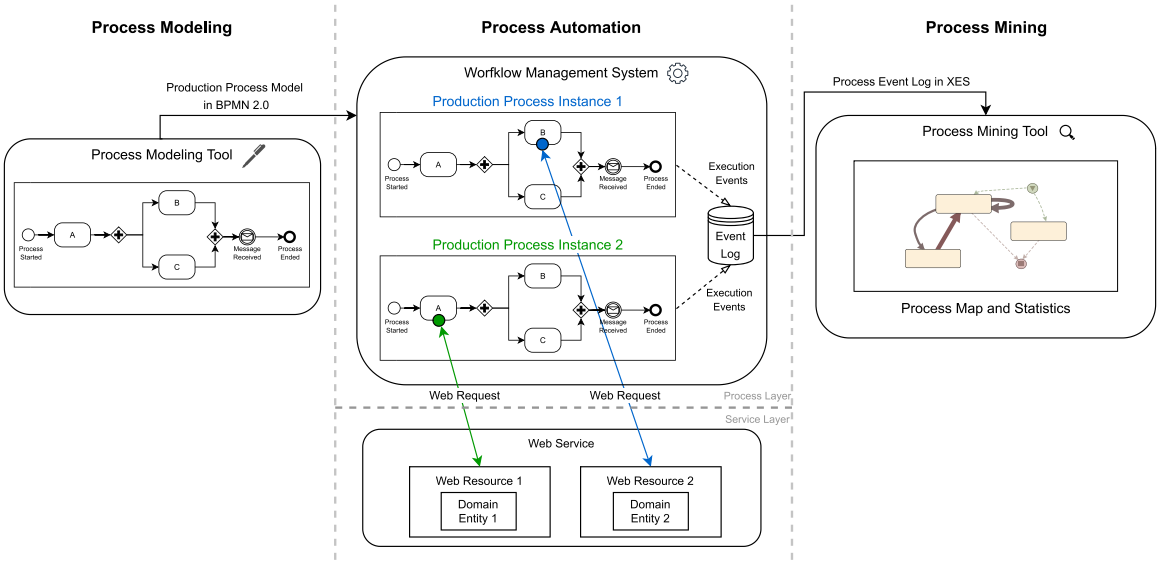


Fig. 5. Workflow among the BPM-related components on the process layer.

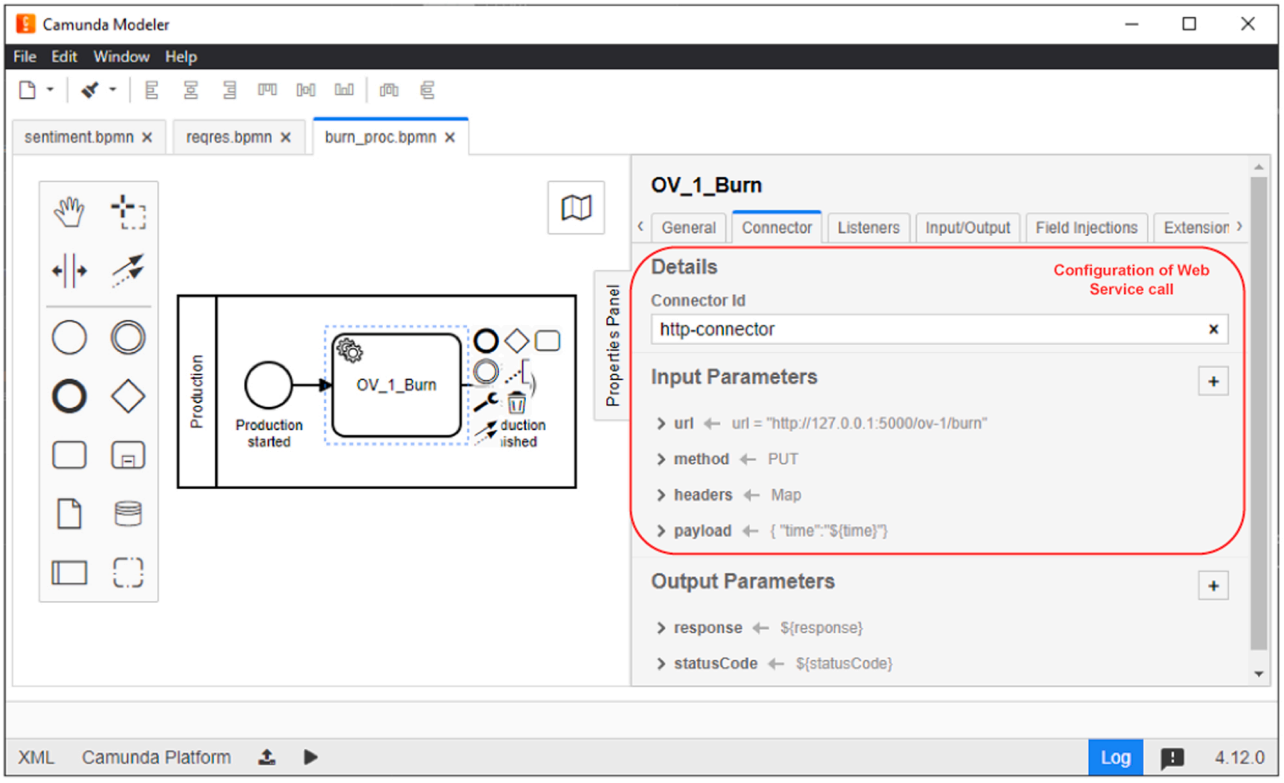


Fig. 6. Screenshot of a process in process modeling Tool.

configuration one this resulted in 6 instances of 6 different classes; for configuration two, 11 instances of 8 different classes. This already provides a good indication of the general suitability of our framework to be implemented in different IIoT setups.

In the following, we describe four selected use cases referring to the application scenarios AS1–AS5 mentioned in Sect. 2 that emerged with the integration of BPM and IIoT: 1) modeling and automation of discrete production processes (cf. AS1, AS2), 2) choosing the granularity-levels of process activities (cf. AS1, AS2), 3) IIoT event processing in production processes cf. AS5), and 4) mining of production processes (cf. AS3, AS4). These use cases cover more process-oriented viewpoints focusing on the actual production control via modeled processes and process analysis from the perspective of process engineers and business process analysts (cf. ISO/IEC/IEEE 42010:2011 and RAMI 4.0 [85]).

5.1. Modeling and automation of discrete production processes

Production processes are rarely modeled in the same level of detail as other business processes in a company. Usually, the production processes are represented as black box subprocesses in end-to-end business processes of a manufacturing company, or not at all [4]. These business processes are often modeled in BPMN 2.0 and executed in

state-of-the-art WfMS [1]. In this section, we demonstrate how discrete production processes can also be modeled in more detail and put into context of a company's business processes (**AS1**). Moreover, we show how these operational production processes can be automated and executed by WfMS that orchestrate the execution of process activities representing the functionalities of the smart factory machines (**AS2**).

5.1.1. Scenario and Implementation

A typical end-to-end business process integrating enterprise and production functionality is shown in Fig. 7 [86]. Here we see an order-to-product process in BPMN 2.0: Orders from customers may be entered into an ERP system manually, which creates a new order for the production system. The production system then plans the production of the new order and the automated production control system–instructed via the WfMS based on our systems architecture–starts executing the highlighted "Produce Order" sub-process as scheduled. This sub-process consists of several automated activities related to storage, transport, and production (burning) of a workpiece. These activities are implemented via service tasks in BPMN 2.0 that are used by the WfMS (here: Camunda BPM Platform) to call the specific web service resources of the smart factory in correspondence with the components of the layered systems architecture. The figure shows an overlay of this sub-process over the production stations of our smart factory illustrating the resources on Shop Floor 1 that execute the individual tasks. When the workpiece was produced, the ERP system is informed and the product is picked up for shipping. A video of the smart factory executing a more complex production process controlled by a WfMS can be found in [87].

5.1.2. Results and discussion

With the successful implementation of the systems architecture, we have reached the goal of integrating the functionality and data of the smart factory on a business process-oriented level with other information systems [6]. The example shows that we not only support the seamless end-to-end process modeling of human tasks together with automated manufacturing steps in a process model relying on the BPMN 2.0 standard for process design or communication purposes [6], but we are also able to automatically control and orchestrate the modeled production steps of operational real-world processes via an industry-grade WfMS from the low-level sensors and actuators to the (remote) services that control them in a vertical proof-of-concept prototype (**AS1**, **AS2**; **R1** ✓, **R2** ✓). With that we have achieved a vertical integration as well as a horizontal integration of Cloud manufacturing along the supply chains via business processes and (Cloud) service orchestration in accordance with [88]. This integration now opens up the whole body of BPM technologies and research to be used for (business) process and supply chain management in IIoT [89] in the context of process orchestration and choreography [1]. For both, the modeling and execution of the production steps as part of business processes, reasonable abstraction levels for the individual operations and processes have to be found as they simplify the rather complex physical processes in and behavior of production machines to a large degree. Here several trade-offs must be considered as discussed in more detail in Sect. 5.2.

Limitations. The remote control of the production machines via WfMS naturally raises concerns regarding security and safety. For our setup we assume that state-of-the-art security mechanisms for securing communication and access to the individual software components and data are in place. Real-time and safety-critical operations referring to the actuation of the production machines in the physical world should be executed on lower layers of the proposed systems architecture, which is usually the responsibility of a PLC system within the control layer. As shown in Fig. 1 real-time and safety-requirements increase with getting closer to the hardware and physical processes. Thus, the WfMS should only be used to call rather coarse-grained, encapsulated functions, and operations within these functions should be executed on lower levels as discussed in the next section. The smart factory model used for the implementation of the use cases (cf. Sect. 4.1) exhibits many properties and behavior of real-scale production machines in the context of IIoT. However, with it being a small scale simulation environment there are also relevant differences that need to be considered when implementing our proposed systems architecture, e.g., regarding different requirements related to safety and real-time critical operations that are not as strict in the simulation model [12].

5.2. Choosing the granularity-levels of process activities

One of the challenges that arose when designing the class model of the domain core following OOP and DDD principles and developing the associated web services is choosing the right levels of granularity and encapsulation for the methods representing the functionality of the smart factory stations to be exposed via the web service for invocation from the WfMS (**R1**) [90]. Here we need to consider trade-offs between the model complexity, communication effort, and performance on the process layer at the cost of reusability, flexibility, and control of functionality.

5.2.1. Scenario and implementation

In this scenario we compare two different process granularity levels for the manufacturing operation of burning a workpiece in the smart factory's oven. Fig. 8 shows the *Burn* activity on the process layer as one activity that calls the corresponding resource on the service layer, which then passes the call on to the domain layer where the *burn()* method consisting of three subroutines to drive the oven's slide into the oven (*drive_in()*), to blink the oven's LEDs for simulation of the burn process (*blink()*), and to drive the slide out (*drive_out()*), are executed. In comparison, Fig. 9 shows on the process layer the burn activity from the previous figure decomposed as a sequence of three more fine-grained activities representing the aforementioned subroutines that are executed by the WfMS.

5.2.1.1. Experiments. To investigate the impact of the two different granularity levels on the overall process execution time, we conducted an experiment to execute and compare both process variants from Figs. 8 and 9 using the proposed systems architecture and smart factory model presented in Sect. 4.1. Table 3 shows a comparison of the processing times on the individual layers for both processes. These times are the averages of three repeated runs for both processes. We used a state-of-the-art desktop computer for running the software components. One instance each of the web service, the transaction managers and the domain core run in the same virtual machine. An instance of the WfMS runs in another virtual machine on the same host. The control applications for the factory run on distributed embedded computers (controllers), which are connected to the same network via LAN. The software components on the individual layers–Control Layer, Domain Layer, and Service Layer–were instructed via code to log the start and end times of the individual operations. For deriving the execution times on the Process Layer, we used the event log generated by the WfMS.

5.2.2. Results and discussion

The results show that the communication and processing overhead when executing smaller parts of a complex process activity as single activities also naturally leads to an increased execution time of the overall composite process activity. For the *Burn* activity we observed significant time differences between the atomic execution and decomposed execution introduced by the additional communication steps and WfMS. Being a rather heavy-weight application deployed in a dedicated virtual machine in the Process layer, the WfMS needs some time to instantiate a new activity instance upon receiving the request to execute an activity and propagate the request to the Service layer. Processing overheads on the Service and Domain layer are rather minimal as these are tightly coupled with the operating system processes. The Control layer is the direct interface with the hardware and thus monitors and

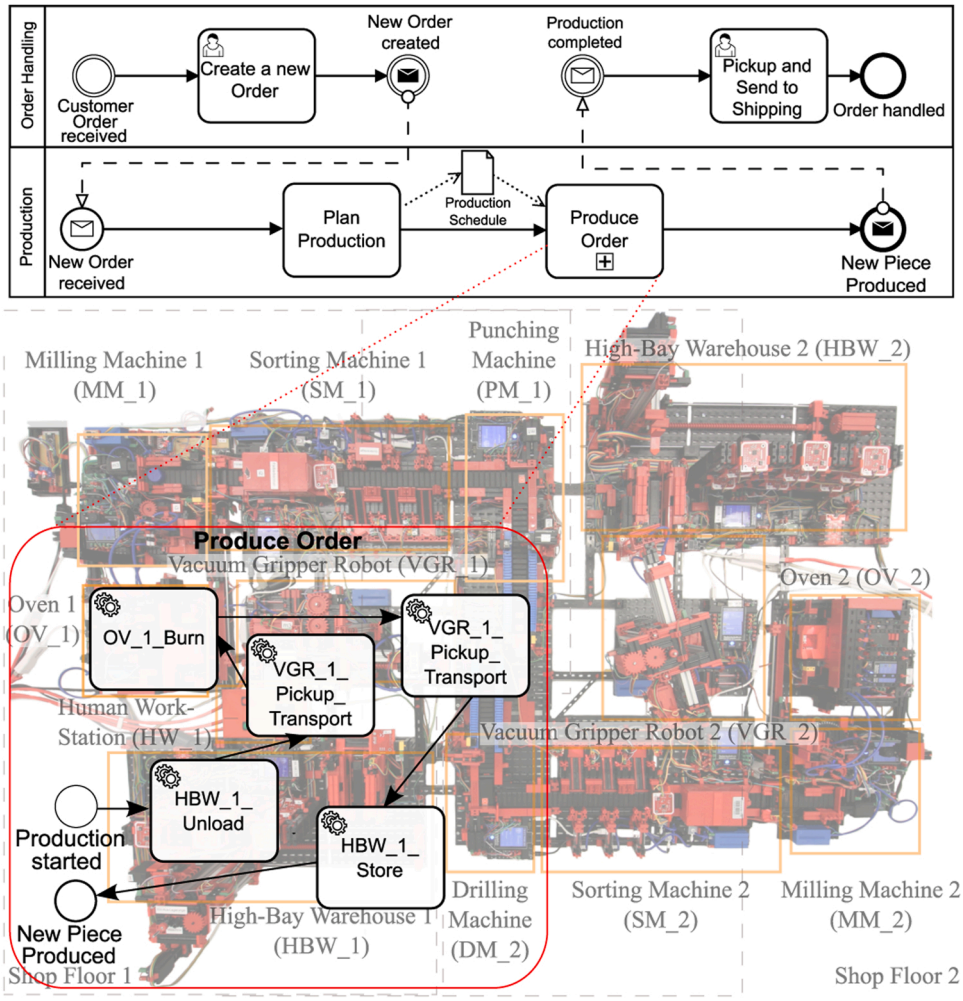


Fig. 7. Order-to-product process with production in our factory.

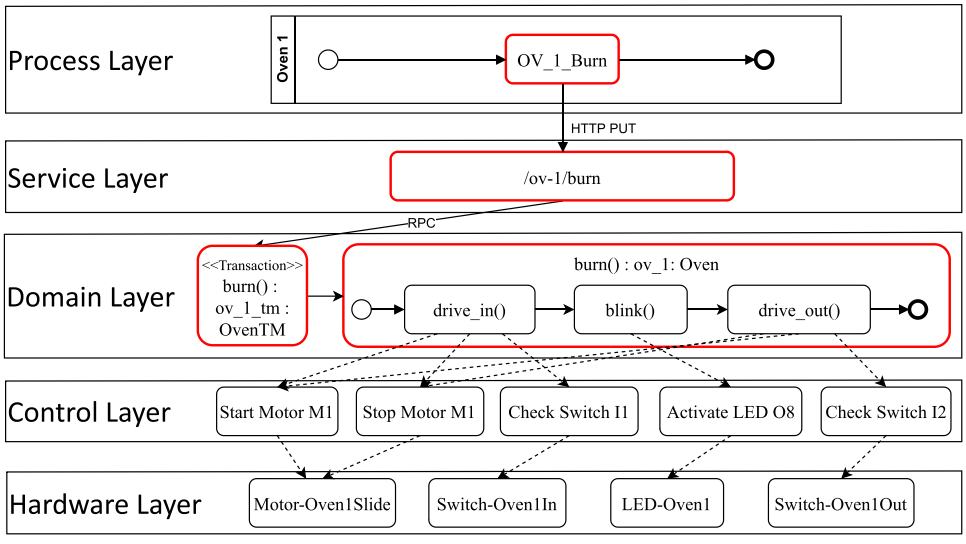


Fig. 8. Execution of *Burn* Encapsulated as one Atomic Process Activity.

controls the execution of the activities in the physical world. Therefore, processing times on this layer are much higher as they represent the interactions in the physical system (i.e., the actual production steps), which are usually significantly slower than the computations in the digital systems on the layers above.

Having a relatively coarse-grained encapsulation of domain functionality to be called from the WfMS (cf. the *Burn* service task on the process layer in Fig. 8) simplifies the process modeling and reduces

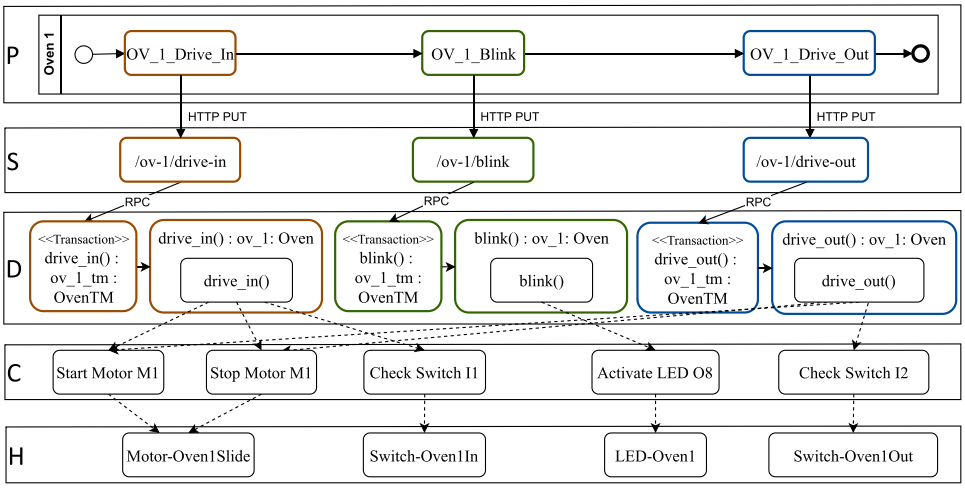


Fig. 9. Execution of *Burn* decomposed into three smaller process activities.

Table 3
Processing times (in ms) on each layer for the *Burn* Activity as Atomic Step (cf. Fig. 8) and Decomposed into three Activities (cf. Fig. 9).

Layers/Activity	Burn (atomic) in ms	Burn (decomposed) in ms		
		Drive_In	Blink	Drive_Out
Process	731	1.512	1.441	1.272
Service	72	59	35	45
Domain	44	27	10	23
Control	23.149	6.586	10.019	6.495
Total Duration	23.996	27.524		

model and transaction complexity. However, these service methods can only be reused in other processes at the same coarse-grained level without any means for controlling intermediate execution steps. Decomposing the service methods into smaller atomic parts (cf. the *Burn* activity decomposed into *Drive_In*, *Blink*, *Drive_Out* as separate transactional tasks of the same web service instance on the process layer in Fig. 9) increases modularity and reuse but also the complexity of the model as well as processing and communication times among the process layer, service layer, and domain layer due to each service call needing to go through all these layers. This is in contrast to the more coarse-grained *Burn* service task requiring only one pass as illustrated in Fig. 8.

Our proposed systems architecture supports the handling of concurrency and transactions on the process layer via the WfMS for inter-service coordination (cf. Sect. 4.6) and on the domain layer via the transaction managers (cf. Sect. 4.4) for the individual production stations for intra-service coordination and limiting access to the physical resources (**R3** ✓). Smaller web services and service methods lead to an increased transaction management effort when running multiple processes simultaneously [91]. On the other hand, the smaller activities enable a more fine-grained reuse and control of functionality in other processes, which results in a higher degree of flexibility when modeling processes. Additionally, smaller units of functionality may lead to a better degree of capacity utilization due to less waiting times during blocking activities and increased parallelism (e.g., a new workpiece can already be driven into the oven for another process instance while the current workpiece is driven out if the oven features two slides, cf. Fig. 9).

All in all, the discussed trade-offs have to be considered when deciding about suitable granularity integrators and disintegrators as well as abstraction levels of process activities within the smart factory. In our systems architecture we propose to apply patterns and principles from object-oriented programming and domain-driven design to find suitable levels of abstraction and encapsulation for domain-specific functionality/capabilities of the IIoT devices (**R1** ✓). In general, the representation of smart factory functionality as process activities of varying granularity enables us to abstract from potentially very complex physical processes and low-code implementations to control the hardware of the production machines. Thus we are able to hide or zoom-in on process execution details as needed [90].

Limitations. In our topology, the software components of the process layer, service layer, and domain layer are running on the same physical computer. These components can also be more distributed, which would result in even longer execution times. This implies that the more time and safety-critical operations (e.g., real-time control loops) in the IIoT system must be implemented closer to the hardware on lower levels of the software stack to achieve faster non-interruptable execution cycles (cf. Fig. 1 [2]). These operations should then be provided as encapsulated functionality to the upper layers–at the cost of flexibility, reusability, and degree of parallelization [92]. However, computing resources close to the hardware, i.e., on the control layer, are more constraint and the respective applications have to be optimized with respect to their runtime environment, which also has to be considered when deciding about suitable levels of encapsulation and deployment locations of functionality. Moreover, the additional resource costs of having rather complex streaming, WfMS, and CEP applications on the higher service and process layers have to be taken into account when deciding about deployment locations of the proposed systems architecture's components. These more heavy-weight components are usually distributed and situated on dedicated servers in the Cloud with more computational resources available than on the Edge in the typical IIoT environment, which in turn requires stronger security mechanisms to be implemented as the smart factory can easily be controlled remotely.

5.3. IIoT event processing in production processes

With **R4** we introduced the requirement of considering events from the IIoT environment and external sources in a more global context related to the process execution [93]. Therefore, the *Process Layer* also contains a CEP platform to derive complex events from the low-level IIoT data with relevance for the business processes. CEP hereby serves as a suitable general framework to derive more complex events from different event sources [93,33,34]. These complex events can be used as basis to initiate specific actions, e.g., to adapt the processes in case of exceptions using the *Process Adaptation Engine* [12,43]. In the following we discuss two scenarios of using IIoT event processing in production processes as classified in [24]: 1) to derive business-level events relevant for the execution, and 2) to check additional context data during the execution of a production activity (**AS5**).

5.3.1. Scenario 1: deriving business-level events

Fig. 10 shows the BPMN 2.0 model of a multi-stage production process containing a sequence of three service tasks as core process activities on Shop Floor 1 of our smart factory (*OV_1_Burn*, *MM_1_Mill*, *PM_1_Punch*). In parallel to the execution of these activities, the WfMS may expect to receive and to react to an external message event (*Machine error occurred*) as modeled. Receiving this message triggers an escalation event *Machine error* on the process level, which leads to the service tasks of checking the workpiece in production (*HW_1_Check_Workpiece*) and performing machine maintenance.

```
@App:name("ShopFloor1Errors")                                          1
                                                                       2
define stream Oven1Stream(status string,                               3
    timestamp string);
define stream Mill1Stream(status string,                               4
    timestamp string);
define stream Punch1Stream(status string,                              5
    timestamp string);
                                                                       6
define stream AlertStream(msg string, station                          7
    string, timestamp string);
                                                                       8
@info(name='OvenQuery')                                                9
from Oven1Stream[status == 'error']                                    10
select 'Machine error occurred' as msg, 'oven1'                        11
    as station, timestamp
insert into AlertStream;                                               12
                                                                       13
@info(name='MillQuery')                                                14
from Mill1Stream[status == 'error']                                    15
select 'Machine error occurred' as msg, 'mill1'                        16
    as station, timestamp
insert into AlertStream;                                               17
                                                                       18
@info(name='PunchQuery')                                               19
from Punch1Stream[status == 'error']                                   20
select 'Machine error occurred' as msg, 'punch1'                       21
    as station, timestamp
insert into AlertStream;                                               22
```

Listing 1:. CEP App for Raising a Machine Error Event as Message in the Production Process (Scenario 1).

Listing 1 presents the corresponding CEP app for the Siddhi platform that is used to monitor the states of the three machines involved and emits a new event as message to the WfMS in case of an error within one of the machines. Each machine emits event data on a dedicated event stream (Lines 3, 4, 5) with event attributes comprising a *status* and *timestamp*. As shown in Fig. 11 these streams are connected to the machine instances (*Domain Entities*) on the Domain Layer via the Streaming Service on the Service Layer using the MQTT protocol. A machine error is indicated via the status attribute of an event having the value “error”. A dedicated SQL-like CEP query (Lines 9, 14, 19) for each machine event stream filters incoming events regarding this value (Lines 10, 15, 20) and emits a new event with the machine error occurrence as message, station name, and timestamp as event attributes (Lines 11, 16, 21) to a stream of alerts (Lines 12, 17, 22) defined in Line 7. With every event on this alert stream a message to the WfMS is sent via HTTP (cf. Fig. 11). This event is then matched to the affected process instances based on the name of the message event (*Machine error occurred*) in the BPMN 2.0 model depicted in Fig. 10. This way, the process level event of a machine error is raised from the raw sensor events of the production machines.

5.3.2. Scenario 2: checking context data

Fig. 12 shows a production process with a typical production activity in the smart factory (*MM_1_Mill*), which is extended with a check of environmental conditions from two different IIoT sensors as there usually may be some constraints on environmental context during the milling of a workpiece. In parallel to starting the milling operation via a service task, the CEP platform receives a service call from the WfMS via a service task (*Check Environmental Conditions*) to start checking temperature and light levels during the milling task. Once the milling is done, the CEP platform emits a message to the WfMS in the form of an HTTP call regarding the environment check, which is evaluated in an event-based gateway based on the message name. If the environmental conditions were met (*Environment OK*), the workpiece is forwarded for sorting; otherwise (*Environment Not OK*) it needs to undergo a manual control (*HW_1_Check_Workpiece*) and may be rejected.

The corresponding CEP application evaluates the incoming event streams from the WfMS, milling machine, temperature sensor, and light sensor. Listing 2 shows the corresponding CEP query *EnvCheckOk*. Here the averages of temperature and light level are calculated for a 10 s-duration of the milling activity (Line 2). The average is only calculated (Line 6) when new events arrive on the environment stream (Line 2) and the milling activity is active (Lines 3–4). When the averages of temperature and light levels are within the allowed ranges (Lines 7–8), then a new event message “Environment OK” is emitted (Lines 5, 9).

```
@info(name='EnvCheckOk')                                               1
from EnvAnalysisStream#window.timeBatch(10sec)                         2
  left outer join MillingActivityState                                 3
  on MillingActivityState.state == "active"                            4
select "Environment OK" as msgname,                                    5
  avg(temp) as avgtemp, avg(light) as avglight                         6
having (avgtemp >= 10 and avgtemp<=15) and                             7
  (avglight >= 150 and avglight <=200)                                 8
insert into EnvCheckStream;                                            9
```

Listing 2:. CEP Query for Checking Environment Conditions during Activity Execution (Scenario 2).

5.3.3. Results and discussion

The two scenarios show that the combination of BPM with CEP and

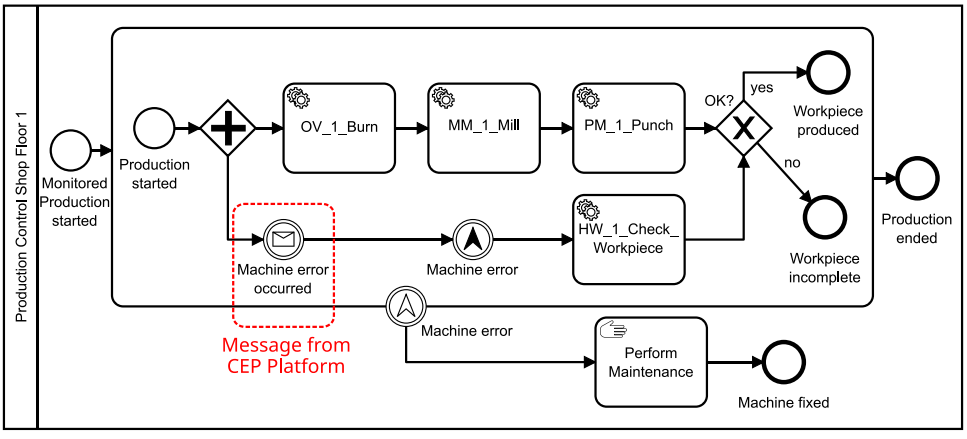


Fig. 10. Integration of a derived high-level process event in a production process (Scenario 1).

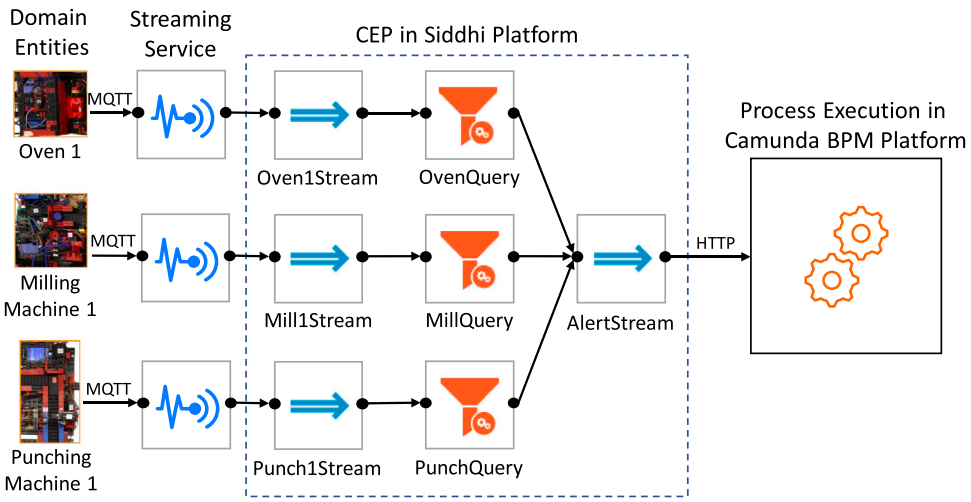


Fig. 11. Integration of IIoT event processing with the execution of a production process (Scenario 1).

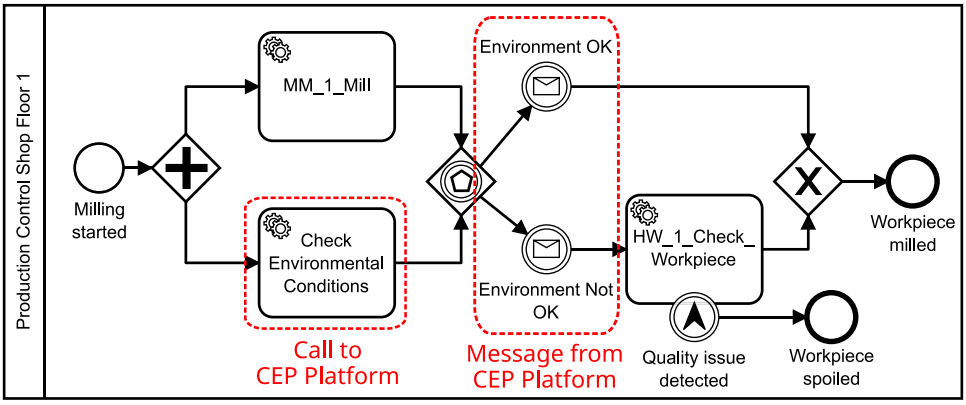


Fig. 12. Extension of a production activity with a check of environmental conditions (Scenario 2).

the extension of "normal" process-based systems with event-driven behavior on the process layer is rather easy to achieve and very useful. The streaming service and CEP platform allow for an integration and processing of real-time data from arbitrary distributed sources to derive complex events with relevance for the processes [32,33]–either as business level events or as contextual data [24] (**R4** ✓). Both the modeling of the processes (e.g., in BPMN 2.0 supported by templates in Camunda Modeler) and writing of the SQL-like CEP queries (supported by drag & drop features of the CEP platform Siddhi's graphical editor) do not require deep programming skills and can also be achieved in a low-code fashion by process engineers. Especially the second scenario shows that it can be very beneficial to link data from sensors to the corresponding process activities, i.e., to also put sensor data in a process context. This new information helps with analysis and segmentation of the potentially large data streams from IIoT, which are now enriched with process data [8] to for example enable traceability of process activity executions where certain environmental conditions were not met (e.g., in cold chain logistics [89]).

5.4. Mining of production processes

Among the advantages of combining WfMS with IIoT for modeling and executing production processes is that the corpus of tools and concepts of BPM, in particular for process monitoring and mining, becomes available for IIoT at rather low cost [11] (**AS4**). The WfMS is able to monitor and create digital traces of the process executions in an *event log*. This log can be exploited by process mining techniques to discover processes and obtain statistics, to check for conformance, and to optimize processes, both offline [22,23] and even online at runtime with the IIoT environments constantly streaming process-related data [83,84].

5.4.1. Scenario and implementation

Fig. 13 shows a process map and statistics obtained using the process mining tool *Fluxicon Disco* from a given event log for the executions of an exemplary, more complex production process in the smart factory. The process was executed based on the presented systems architecture with the *Camunda BPM Platform* as WfMS on the process layer, which also created the event log. The map shows the execution of activities as directly-follows graph with mean execution times and absolute frequencies of process activities and transitions for two cases of one process variant. The labels of the activities follow the proposed naming schemes (cf. Sect. 4.6) indicating the instance of production machine (i.e., process *resource*) and production functionality that was executed. Using the process mining tool we are able to derive that the process consists of 16 production-related activities (*Burn*, *Drill*, *Mill*, *Punch*), storage and transport operations, and *Check_Workpiece* for quality control. The event log contains 23 process events in total per case with a mean case duration of 15.7 min. The event log not only contains data regarding the *control flow*, but also additional attributes allowing further analysis, e.g. to analyze the performance of the resources (here: 12 production stations) that executed the process activities (cf. Fig. 14). We were able to derive all this information via the process mining algorithms for *process discovery* based on data stored in the event log. It was not necessary to have the underlying BPMN 2.0 process model available.

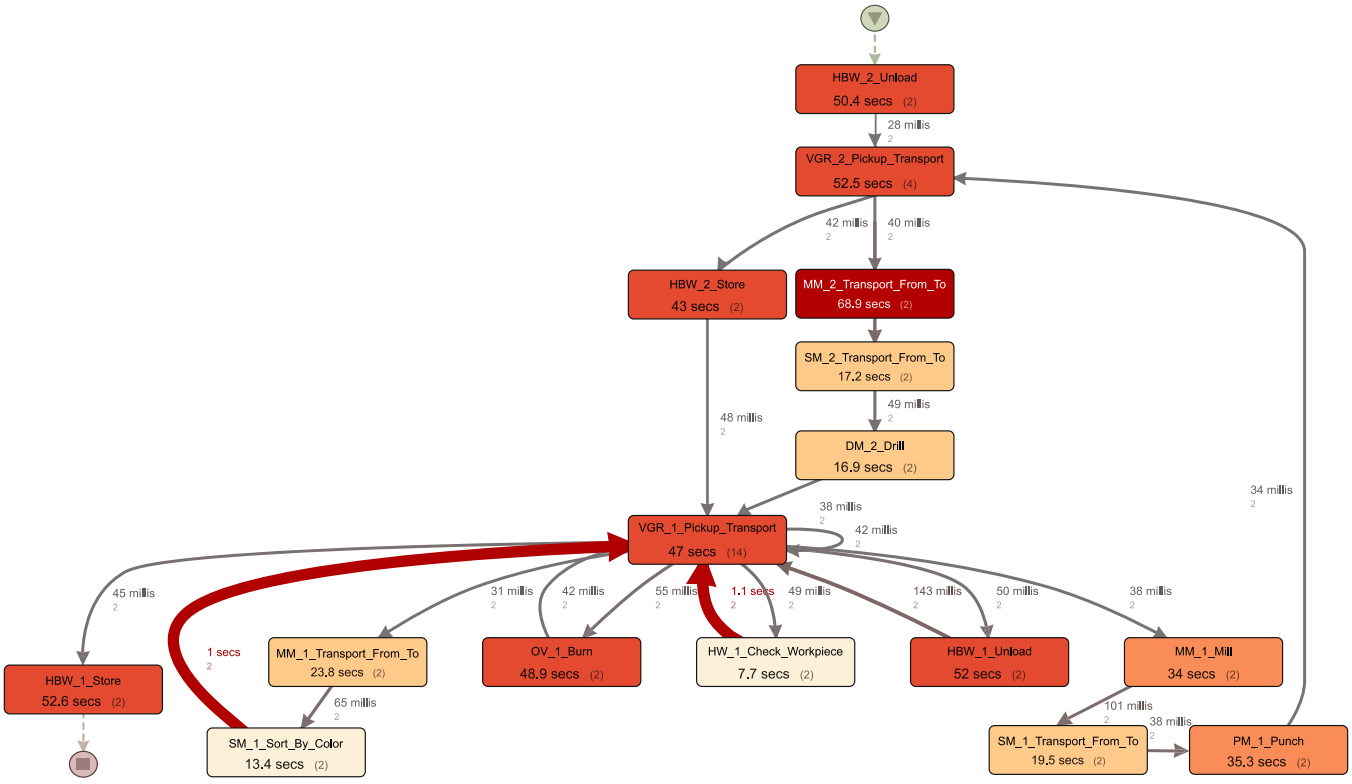
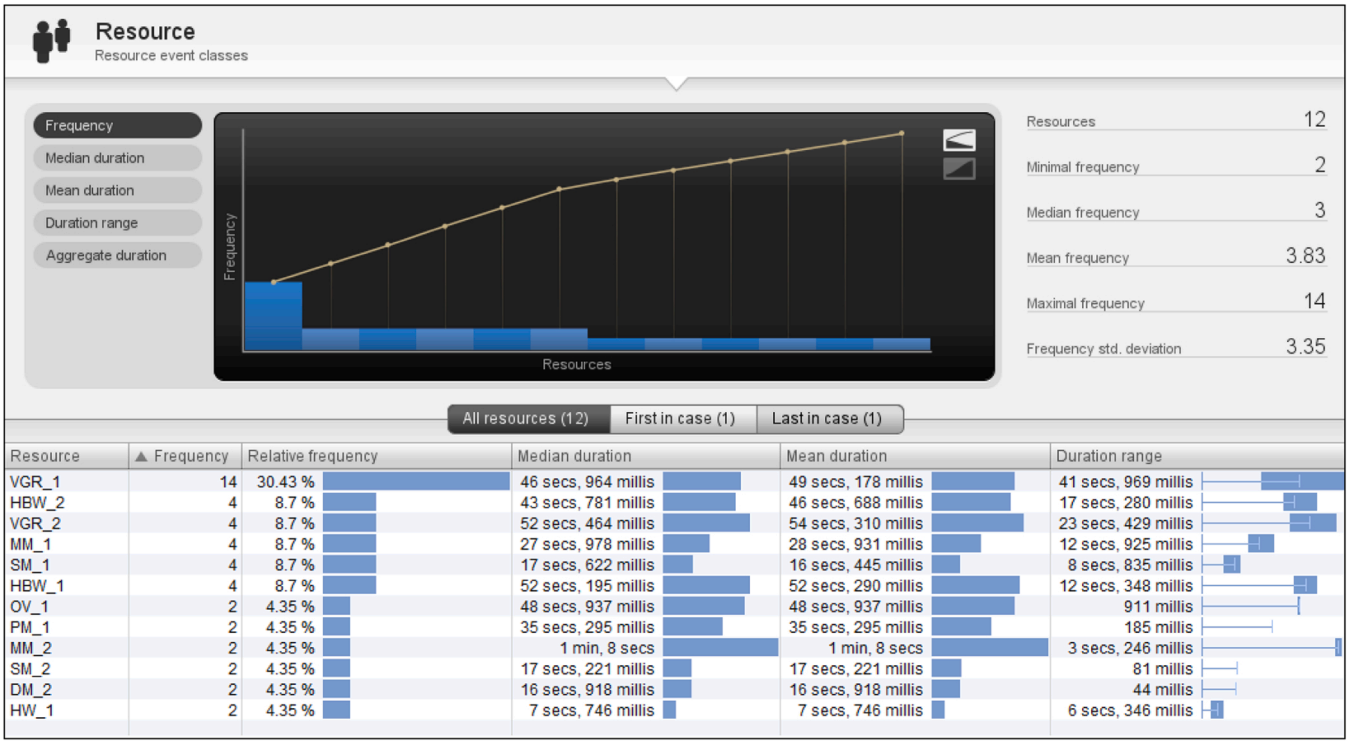


Fig. 13. Performance map of a production process obtained by a process mining tool.



Resource	▲ Frequency	Relative frequency	Median duration	Mean duration	Duration range
VGR_1	14	30.43 %	46 secs, 964 millis	49 secs, 178 millis	41 secs, 969 millis
HBW_2	4	8.7 %	43 secs, 781 millis	46 secs, 688 millis	17 secs, 280 millis
VGR_2	4	8.7 %	52 secs, 464 millis	54 secs, 310 millis	23 secs, 429 millis
MM_1	4	8.7 %	27 secs, 978 millis	28 secs, 931 millis	12 secs, 925 millis
SM_1	4	8.7 %	17 secs, 622 millis	16 secs, 445 millis	8 secs, 835 millis
HBW_1	4	8.7 %	52 secs, 195 millis	52 secs, 290 millis	12 secs, 348 millis
OV_1	2	4.35 %	48 secs, 937 millis	48 secs, 937 millis	911 millis
PM_1	2	4.35 %	35 secs, 295 millis	35 secs, 295 millis	185 millis
MM_2	2	4.35 %	1 min, 8 secs	1 min, 8 secs	3 secs, 246 millis
SM_2	2	4.35 %	17 secs, 221 millis	17 secs, 221 millis	81 millis
DM_2	2	4.35 %	16 secs, 918 millis	16 secs, 918 millis	44 millis
HW_1	2	4.35 %	7 secs, 746 millis	7 secs, 746 millis	6 secs, 346 millis

Fig. 14. Resource analysis obtained by a process mining tool.

5.4.2. Results and discussion

The lack of visibility and traceability regarding interactions among devices and components is identified as a negative side effect of the loose coupling and high flexibility in event-driven, distributed systems such as IIoT [21], which may often also feature autonomous behavior of software components (*agents*). Using WfMS for monitoring of these interactions in a more choreography-style and for partial orchestration already opens up opportunities for applying process mining on the event logs or event streams generated by the WfMS and thus for making the often hidden end-to-end workflows visible and explicit (**AS3**, **AS4**). With using a WfMS for the (business) process-based control and monitoring of the smart factory, we can rely on the WfMS as central entity to track the status of all process executions, which can be beneficial for online and offline analysis of processes, e.g., for process optimizations and adaptations [12,94]. As shown in [84], event logs and streams suitable for offline and online process mining can also be partially derived directly from the raw IIoT data using CEP (**R4** ✓)–thus making the existence of a WfMS in the IIoT environment optional in case the focus is only on monitoring and not on orchestration, e.g., when computing resources are scarce and running a dedicated WfMS for coordination is not feasible or necessary.

Process mining algorithms can then be applied on the available event logs and event streams to discover the interactions of IIoT devices in the form of process models as-is and process statistics (end-to-end workflows). As we also know the underlying BPMN 2.0 model of the production process displayed in Fig. 13, we could now start analyzing the executions for conformance to determine if the execution of an instance deviated from the normative model e.g., using the ProM[9] framework. Moreover, the statistics obtained from the event logs may provide valuable insights into the performance of resources and potential bottlenecks within the production processes and thus open up opportunities for process enhancements and optimizations [23].

5.5. Summary of contributions

In Sect. 3.3 we identified a research gap concerning the integration of IIoT components with BPM systems and vice versa. Proposed systems architectures either do not sufficiently address aspects related to the BPM lifecycle; or BPM-related approaches do not sufficiently elaborate on the individual software systems architectures and development steps.

Our proposed layered systems architecture provides a detailed reference framework of necessary development steps, architectural decisions, and software components that enable the integration of IIoT with BPM technology. Moreover, we demonstrate an instantiation of the framework for two different smart factory configurations. All the aspects of our solution involve best practices and design patters from software engineering and software architecture. We discuss the communication between industry-grade process-oriented information systems with sensors and actuators of an IIoT environment along the entire BPM lifecycle with exemplary use cases related to process modeling, process automation and process mining in a smart factory model. Additionally, we show how to integrate context data from IIoT devices on a BPM level via complex event processing.

6. Conclusion and future work

The goal of our work was to establish a closer communication and interaction between the rather low-level hardware components of an IIoT environment with PAIS on a process management level to actively monitor and control production processes as part of an enterprise's business processes. Related research has focused on the selective integration of IIoT-related aspects into business processes without providing detailed descriptions of the underlying architectures. On the other hand, proposed software and systems architectures for IIoT do not explicitly address the interactions with existing PAIS and BPM in general. Addressing requirements of abstraction and encapsulation, remote access, handling of concurrency, and integration of IIoT data (**R1**–**R4**), we presented a systems architecture for process-based management of IIoT, which fits into the general RAMI 4.0 framework [28,55]. This systems architecture was developed following the design science research methodology (DSRM) [15,25]. We provided descriptions of the individual software components and their interactions to close the gap between the shop floor and PAIS in both directions. We were able to show the feasibility and many benefits of our approach to use BPM technology in IIoT environments with a proof-of-concept implementation of the architecture for a smart factory model. The systems architecture and its components have already proven to be rather generic, as we have successfully instantiated it for another smart factory with a different topology without much effort [84]. To what degree the developed concepts can be transferred to larger scale and more realistic smart factories remains open for future investigations. All in all, the systems architecture makes the corpus of BPM technologies available to be used in IIoT environments for process modeling, automation, and mining [11,12] (**AS1**–**AS5**), which opens up many new opportunities for research on BPM and IIoT [11].

Future research includes the instantiation of the systems architecture in other IoT environments that have similar properties and requirements (e.g., smart homes [43], smart health [35], smart emergency management [44]) to enable BPM and IoT research [11]. Here we expect the modifications to be mostly necessary in creating the specific domain core classes. We will also use the systems architecture as basis to investigate more advanced BPM and software engineering concepts regarding flexible and adaptive processes [43,12,45] as well as process mining techniques for IIoT [12,84]. Moreover, we will investigate the use of model-driven software engineering for describing the software components of the domain core, the event processing applications and the operational processes with the goal of synthesizing the corresponding software artifacts as well as *Digital Twins* of the smart factory for simulation [95,96,22,97].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Dumas M, LaRosa M, Mendling J, Reijers HA. Fundamentals of Business Process Management. Springer; 2018.
[2] Ismail A, Truong H-L, Kastner W. Manufacturing process data analysis pipelines: a requirements analysis and survey. J Big Data 2019;6(1).
[3] Diedrich C, et al. Semantic Interoperability for Asset Communication within Smart Factories. 22nd Conf. on Emerg. Technol. & Fact. Autom. IEEE,; 2017. p. 1–8.
[4] Erasmus J, Vanderfeesten I, Traganos K, Grefen PWPJ. Using business process models for the specification of manufacturing operations. Comput Ind 2020;123.
[5] Monostori L. Cyber-physical production systems: roots, expectations and R&D challenges. Procedia CIRP 2014;17:9–13.
[6] Erasmus J, Vanderfeesten I, Traganos K, Grefen PWPJ. The Case for Unified Process Management in Smart Manufacturing. 22nd EDOC. IEEE; 2018. p. 218–27.
[7] Prades L, Romero F, Estruch A, García-Dominguez A, Serrano J. Defining a methodology to design and implement business process models in BPMN according to the standard ANSI/ISA-95 in a manufacturing enterprise. Procedia Eng 2013;63:115–22.
[8] Rinderle-Ma S, Mangler J. Process Automation and Process Mining in Manufacturing. In: BPM, 12875. Springer; 2021. p. 3–14.
[9] Schönig S, Ackermann L, Jablonski S, Ermer A. IoT meets BPM: a bidirectional communication architecture for IoT-aware process execution. Softw Syst Model 2020;19(6):1443–59.
[10] Sisinni E, Saifullah A, Han S, Jennehag U, Gidlund M. Industrial internet of things: challenges, opportunities, and directions. IEEE Trans Ind Inf 2018;14(11):4724–34.
[11] Janiesch C, et al. The internet of things meets business process management: a manifesto. IEEE Syst Man Cybern Mag 2020;6(4):34–44.

[9] https://www.promtools.org/

[12] Malburg L, Seiger R, Bergmann R, Weber B. Using Physical Factory Simulation Models for Business Process Management Research. In: BPM Workshops, vol. 397 of LNBIP. Springer; 2020. p. 95–107.
[13] International Organization for Standardization (ISO), ISO/IEC/IEEE 42010:2011, Systems and Software Engineering–Architecture Description (2011).
[14] López CEB. Real-time event-based platform for the development of digital twin applications. Int J Adv Manuf Technol 2021;116(3–4):835–45.
[15] Hevner AR, March ST, Park J, Ram S. Design science in information systems research. MIS Q 2004:75–105.
[16] Radziwon A, Bilberg A, Bogers M, Madsen ES. The smart factory: exploring adaptive and flexible manufacturing solutions. 24th DAAAM Int Symp Intell Manuf Autom 2014;69:1184–90.
[17] Hehenberger P, Vogel-Heuser B, Bradley D, Eynard B, Tomiyama T, Achiche S. Design, modelling, simulation and integration of cyber physical systems: Methods and applications. Comput Ind 2016;82:273–89.
[18] Zuehlke D. Smartfactory–towards a factory-of-things. Annu Rev Control 2010;34 (1):129–38.
[19] Bauer M, et al. IoT Reference Model. In: Enabling things to talk, SpringerOpen. Springer; 2013. p. 113–62.
[20] Seiger R, Aßmann U, Huber S, Case A. Study for Workflow-Based Automation in the Internet of Things. In: IEEE Int. Conf. on Softw. Archit. Companion. IEEE; 2018. p. 11–8.
[21] Cugola G, Nitto ED, Fuggetta A. Exploiting an Event-Based Infrastructure to Develop Complex Distributed Systems. In: Proc. of the Int. Conf. on Softw. Eng. IEEE; 1998. p. 261–70.
[22] Friederich J, Francis DP, Lazarova-Molnar S, Mohamed N. A framework for data-driven digitial twins of smart manufacturing systems. Comput Ind 2022;136: 103586.
[23] van der Aalst WMP, et al. Process Mining Manifesto. In: BPM Workshops. Springer; 2012. p. 169–94.
[24] Bertrand Y, de Weerdt J, SerralAsensio E, Bridging A. Model for Process Mining and IoT. In: ICPM Workshops, vol. 433 of LNBIP. Springer; 2022. p. 98–110.
[25] Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S. A design science research methodology for information systems research. J Manag Inf Syst 2008;24(3): 45–77.
[26] Grefen PWPJ, Ludwig H, Tata S, Dijkman R, Baracaldo N, Wilbik A, D'Hondt T. Complex Collaborative Physical Process Management: A Position on the Trinity of BPM, IoT and DA. In: Collaborative Networks of Cognitive Systems. Springer; 2018. p. 244–53.
[27] Derler P, Lee EA, Vincentelli AS. Modeling cyber-physical systems. Proc IEEE 2012; 100(1):13–28.
[28] CruzSalazar LA, Ryashentseva D, Lüder A, Vogel-Heuser B. Cyber-physical production systems architecture based on multi-agent's design pattern–comparison of selected approaches mapping four agent patterns. Int J Adv Manuf Technol 2019;8107:39.
[29] Sanchis R, García-Perales Ó, Fraile F, Poler R. Low-code as enabler of digital transformation in manufacturing industry. Appl Sci 2020;10(1):12.
[30] Lee EA. Cyber Physical Systems: Design Challenges. In: 11th IEEE Int. Symp. on Object-Oriented Real-Time Dist. Comp. IEEE; 2008. p. 363–9.
[31] Ehrendorfer M, Mangler J, Rinderle-Ma S. Sensor Data Stream Selection and Aggregation for the Ex Post Discovery of Impact Factors on Process Outcomes. In: Intelligent Information Systems. Springer; 2021. p. 29–37.
[32] Vierhauser M, Rabiser R, Grünbacher P, Seyerlehner K, Wallner S, Zeisel H. ReMinds : a flexible runtime monitoring framework for systems of systems. J Syst Softw 2016;112:123–36.
[33] Baumgrass A, et al. GET Controller and UNICORN: event-driven process execution and monitoring in logistics. Proc Demo Sess 13th Int Conf BPM 2015;1418:75–9.
[34] Malburg L, Rieder M-P, Seiger R, Klein P, Bergmann R. Object detection for smart factory processes by mach learning. Procedia Comput Sci 2021;184:581–8.
[35] Chang C, Srirama SN, Buyya R. Mobile cloud business process management system for the internet of things: a survey. ACM Comput Surv 2017;49(4):70:1–70:42.
[36] Torres V, SerralAsensio E, Valderas P, Pelechano V, Grefen PWPJ. Modeling of IoT devices in Business Processes: A Systematic Mapping Study. In: 22nd Conf. on Bus. Inform. IEEE; 2020. p. 221–30.
[37] Bazan P, Estevez E. Industry 4.0 and business process management: state of the art and new challenges. BPMJ 2021.
[38] Fattouch N, BenLahmar I, Boukadi K. IoT-aware Business Process: comprehensive survey, discussion and challenges. In: 29th Int. Conf. on Enabling Technol.: Infrastruct. for Collab. Enterp. IEEE; 2020. p. 100–5.
[39] Baumgraß A, Botezatu M, Di Ciccio C, Dijkman R, Grefen PWPJ, Hewelt M, Mendling J, Meyer A, Pourmirza S, Völzer H. Towards a methodology for the engineering of event-driven process applications. In: BPM. Springer; 2016. p. 501–14.
[40] Kammerer K, Pryss R, Hoppenstedt B, Sommer K, Reichert M. Process-driven and flow-based processing of industrial sensor data. Sensors 2020;20(18):5245.
[41] M. Koot, M.-E. Iacob, M. R. K. Mes, A Reference Architecture for IoT-Enabled Dynamic Planning in Smart Logistics, in: Adv. Inf. Syst. Eng., vol. 12751 of LNCS, Springer Nature, 2021, pp. 551–565.
[42] Seiger R, Huber S, Schlegel T. Toward an execution system for self-healing workflows in cyber-physical systems. Softw Syst Model 2018;17(2):551–72.
[43] Seiger R, Huber S, Heisig P, Aßmann U. Toward a framework for self-adaptive workflows in cyber-physical systems. Softw Syst Model 2019;18(2):1117–34.
[44] Marrella A, Mecella M, Sardiña S. Intelligent process adaptation in the smartPM system. ACM Trans Intell Syst Technol 2017;8(2):25:1–25:43.
[45] Malburg L, Klein P, Bergmann R. Semantic web services for AI-research with physical factory simulation models in industry 4.0. Proc Int Conf Innov Intell Ind Prod Logist 2020:32–43.
[46] Wieland M, Schwarz H, Breitenbucher U, Leymann F. Towards situation-aware adaptive workflows: SitOPT - A general purpose situation-aware workflow management system. In: Int. Conf. on Pervasive Comput. and Commun. Workshops. IEEE; 2015. p. 32–7.
[47] Traganos K, Vanderfeesten I, Grefen PWPJ, Erasmus J, Gerrits T, Verhofstad W. End-to-End Production Process Orchestration for Smart Printing Factories: An Application in Industry. 24th EDOC. IEEE; 2020. p. 155–64.
[48] Traganos K, Grefen PWPJ, Vanderfeesten I, Erasmus J, Boultadakis G, Bouklis P. The HORSE framework: a reference architecture for cyber-physical systems in hybrid smart manufacturing. J Manuf Syst 2021;61:461–94.
[49] Bordel Sánchez B, Alcarria R, Rivera DS d, Sánchez-Picot Á. Enhancing process control in industry 4.0 scenarios using cyber-physical systems. Dependable Appl 2016;7(4):41–64.
[50] P. Valderas, V. Torres, E. Serral Asensio, Modelling and executing IoT-enhanced business processes through BPMN and microservices, 2021, 111139.
[51] Ray PP. A survey on Internet of Things architectures. J King Saud Univ Comput Infom Sci 2018;30(3):291–319.
[52] Muccini H, Moghaddam MT. IoT Architectural Styles - A Systematic Mapping Study. In: 12th Europ. Conf. on Softw. Archit., vol. 11048. Springer; 2018. p. 68–85.
[53] Nakagawa EY, Antonino PO, Schnicke F, Capilla R, Kuhn T, Liggesmeyer P. Industry 4.0 reference architectures: state of the art and future trends. Comput Ind Eng 2021;156:107241.
[54] Malakuti S, Goldschmidt T, Koziolek H. A Catalogue of Architectural Decisions for Designing IIoT Systems. In: 12th Europ. Conf. on Softw. Archit., vol. 11048 of LNCS. Springer; 2018. p. 103–11.
[55] S.-W. Lin et al., Architecture Alignment and Interoperability: An Industrial Internet Consortium and Plattform Industrie 4.0 Joint Whitepaper, 2018.
[56] Lee J, Bagheri B, Kao H-A, Cyber-PhysicalSystems A. architecture for Industry 4.0-based manufacturing systems. Manuf Lett 2015;3:18–23.
[57] Ahmadi A, Cherifi C, Cheutet V, Ouzrout Y. A review of CPS 5 components architecture for manufacturing based on standards. 11th Int Conf Softw, Knowl, Inf Manag Appl (SKIMA 2017:1–6.
[58] Jiang J-R. An improved cyber-physical systems architecture for Industry 4.0 smart factories. Adv Mech Eng 2018;10(6).
[59] Alexopoulos K, Sipsas K, Xanthakis E, Makris S, Mourtzis D. An industrial Internet of things based platform for context-aware information services in manufacturing. Int J Comput Integr Manuf 2018;31(11):1111–23.
[60] Kuhn T, Sadikow S, Antonino PO, Service-Based A. Production Ecosystem Architecture for Industrie 4.0. KI 2019;33(2):163–9.
[61] Perzylo A, Grothoff J, Lucio L, Weser M, Malakuti S, Venet P, et al. Capability-based semantic interoperability of manufacturing resources: a BaSys 4.0 perspective. IFAC-Pap 2019;52(13):1590–6. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM.
[62] Pérez F, Irisarri E, Orive D, Marcos M, Estevez E. A cpps architecture approach for industry 4.0. 2015 IEEE 20th Conf Emerg Technol Fact Autom (ETFA) 2015:1–4.
[63] Mahnke W, Leitner S-H, Damm M. OPC Unified Architecture. Springer Science & Business Media; 2009.
[64] Luo Z, Hong S, Lu R, Li Y, Zhang X, Kim J, et al. OPC UA-based smart manufacturing: system architecture, implementation, and execution. 5th Int Conf Enterp Syst 2017:281–6.
[65] Trunzer E, et al. System architectures for Industrie 4.0 applications. Prod Eng Res Dev 2019;13(3):247–57.
[66] Muccini H, Spalazzese R, Moghaddam MT, Sharaf M. Self-Adaptive IoT Architectures: an Emergency Handling Case Study. In: 12th Europ. Conf. on Softw. Archit. ACM; 2018. 19:1–19:6.
[67] Muccini H, Arbib C, Davidsson P, Moghaddam MT. An IoT software architecture for an evacuable building architecture. 52nd HICSS Sch 2019:1–10.
[68] Crnkovic I, Malavolta I, Muccini H, Sharaf M. On the Use of Component-Based Principles and Practices for Architecting Cyber-Physical Systems. In: 19th Int. Symp. on Comp.-Based Softw. Eng. IEEE; 2016. p. 23–32.
[69] Alkhabbas F, De Sanctis M, Spalazzese R, Bucchiarone A, Davidsson P, Marconi A. Enacting Emergent Configurations in the IoT Through Domain Objects. In: 16th Int. Conf. on Service-Oriented Comput., vol. 11236 of LNCS. Springer; 2018. p. 279–94.
[70] Gharbi I, Barkaoui K, Samir BA. An Intelligent Agent-Based Industrial IoT Framework for Time-Critical Data Stream Processing. In: Mobile, Secure, and Programmable Networking. Springer; 2021. p. 195–208.
[71] Lu J, Chen D, Wang G, Kiritsis D, Törngren M. Model-Based Systems Engineering Tool-Chain for Automated Parameter Value Selection. Trans Syst Man Cyber Syst 2022;52(4):2333–47.
[72] van der Aalst WMP. Business process management: a comprehensive survey. Int Sch Res Not 2013 2013.
[73] Mourtzis D. Machine tool 4.0 in the era of digital manufacturing. 32nd Eur Model Simul Symp 2020:416–29.
[74] Armendia M, Alzaga A, Peysson F, Fuertjes T, Cugnon F, Ozturk E, Flum D. Machine tool: from the digital twin to the cyber-physical systems. In: Twin-Control. Springer; 2019. p. 3–21.
[75] Klein P, Bergmann R. Generation of complex data for AI-based predictive maintenance research with a physical factory model. 16th Int Conf Inform Control Autom Rob SciTePress 2019:40–50.
[76] Evans E, Evans EJ. Domain-driven design: tackling complexity in the heart of software. Addison Wesley Prof 2004.

[77] Klein P, Malburg L, Bergmann R. FTOnto: a domain ontology for a fischertechnik simulation production factory by reusing existing ontologies. Proc Conf LWDA 2019;2454:253–64.
[78] Rossit DA, Tohmé F, Frutos M. Industry 4.0: smart scheduling. Int J Prod Res 2019; 57(12):3802–13.
[79] Ford N, Richards M. Fundamentals of software architecture. OaReilly Media Inc 2020.
[80] Seiger R, Aßmann U. Consistency and Synchronization for Workflows in Cyber-physical Systems. In: Proc. of the 10th ACM/IEEE Int. Conf. on Cyber-Phys. Syst. ACM; 2019. p. 312–3.
[81] Seiger R, Heisig P, Aßmann U. Retrofitting of Workflow Management Systems with Self-x Capabilities for Internet of Things. In: Daniel F, Sheng QZ, Motahari H, editors. Business Process Management Workshops. Springer; 2019. p. 433–44.
[82] C. W. Günther, E. Verbeek, XEX Standard Definition - Version 2.0, 2014.
[83] Burattin A, Eigenmann M, Seiger R, Weber B. MQTT-XES: Real-time telemetry for process event data. Proc Best Diss Award, Dr Consort, Demonstr Resour Track BPM 2020, Vol 2673 CEUR Workshop Proc 2020:97–101.
[84] Seiger R, Zerbato F, Burattin A, Garcia-Banuelos L. B. Weber, Towards IoT-driven Process Event Log Generation for Conformance Checking in Smart Factories. 24th EDOC Workshop. IEEE; 2020. p. 20–6.
[85] Kannengiesser U, Müller H. Towards viewpoint-oriented engineering for Industry 4.0: a standards-based approach. Ind Cyber-Phys Syst (ICPS) 2018:51–6.
[86] Mourtzis D, Gargallis A, Zogopoulos V. Modelling of customer oriented applications in product lifecycle using RAMI 4.0. Procedia Manuf 2019;28:31–6. 7th International conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2018).
[87] Demo Video: Object Detection for Smart Factory Processes by Machine Learning, 10.6084/m9.figshare.13240784 (2020).
[88] Lanza G, Peukert S, Steier GL. Chapter 3- Latest advances in cloud manufacturing and global production networks enabling the shift to the mass personalization paradigm. In: Mourtzis D, editor. Design and Operation of Production Networks for Mass Personalization in the Era of Cloud Technology. Elsevier; 2022. p. 39–77.
[89] Javaid M, Haleem Abid, Pratap Singh R, Rab S, Suman R. Upgrading the manufacturing sector via applications of industrial internet of things (IIoT). Sens Int 2021;2:100129.
[90] Zerbato F, Seiger R, Di Federico G, Burattin A, Weber B. Granularity in process mining: can we fix it? CEUR Workshop Proc 2021;2938:40–4.
[91] Richards M. Microservices vs. Service-Oriented Architecture. O'Reilly; 2015.
[92] Ribeiro L, Björkman M. Transitioning from standard automation solutions to cyber-physical production systems: an assessment of critical conceptual and technical challenges. IEEE Syst J 2018;12(4):3816–27.
[93] Soffer P, et al. From event streams to process models and back: challenges and opportunities. Inf Syst 2019;81:181–200.
[94] Hoffmann M, Malburg L, Bergmann R. ProGAN: Toward a Framework for Process Monitoring and Flexibility by Change via Generative Adversarial Networks. In: Marrella A, Weber B, editors. BPM Workshops. Springer; 2022. p. 43–55.
[95] Kirchhof JC, Michael J, Rumpe B, Varga S, Wortmann A. Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: 23rd Int. Conf. on Model Driven Eng. Lang. and Syst. ACM; 2020. p. 90–101.
[96] B. Ashtari Talkhestani, T. Jung, B. Lindemann, N. Sahlab, N. Jazdi, W. Schloegl, M. Weyrich, An architecture of an Intelligent Digital Twin in a Cyber-Physical Production System, at - Automatisierungstechnik 67, 9, 2019, pp.762–782.
[97] Zhang J, Deng C, Zheng P, Xu X, Ma Z. Development of an edge computing-based cyber-physical machine tool. Robot Comput Integr Manuf 2021;67:102042.