A Process to Non-invasively Augment Legacy IoT Systems Using Business Processes and Microservices

Ronny Seiger
ronny.seiger@unisg.ch
University of St.Gallen
St.Gallen, Switzerland

Marco Franceschetti
marco.franceschetti@unisg.ch
University of St.Gallen
St.Gallen, Switzerland

Amine Abbad-Andaloussi
amine.abbad-andaloussi@unisg.ch
University of St.Gallen
St.Gallen, Switzerland

ABSTRACT

Nowadays, Internet of Things (IoT) systems can be found everywhere. Thereby, the IoT systems' software greatly varies in terms of openness and extensibility. Especially older *legacy* IoT systems only provide limited means of interacting and extending their functionality due to proprietary and closed software interfaces. Nevertheless, IoT systems evolve constantly based on new functional requirements and the need for integration with new systems and IoT devices. In this work we investigate the use of *Business Process Management* (BPM) technologies in combination with *Microservices* to augment IoT systems in a non-invasive, lightweight manner focusing on integration, data augmentations, and human-in-the-loop. We start by analyzing the interfaces and processes of a smart factory as a representative legacy system to identify missing functionality, which we generalize into functional requirements for IoT systems. We show how these requirements can be addressed using BPM and microservices based on a discussion of suitable BPM concepts and software architecture, which is accompanied by a prototypical implementation. These results inform our proposal and discussion of a generic development process to non-invasively augment process-oriented legacy IoT systems using BPM and microservices.

CCS CONCEPTS

• **Information systems** → **Process control systems**; • **Software and its engineering** → *Publish-subscribe / event-based architectures*; *Software as a service orchestration system*; • **Computer systems organization** → **Sensors and actuators**; • **Applied computing** → *Business process management systems*; *Service-oriented architectures.*

KEYWORDS

Smart factory, business process management, microservices, software architecture, retrofitting, domain-driven design

ACM Reference Format:
Ronny Seiger, Marco Franceschetti, and Amine Abbad-Andaloussi. 2024. A Process to Non-invasively Augment Legacy IoT Systems Using Business Processes and Microservices. In *14th International Conference on the Internet of Things (IoT 2024), November 19–22, 2024, Oulu, Finland*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3703790.3703791

IoT 2024, November 19–22, 2024, Oulu, Finland

ACM ISBN 979-8-4007-1285-2/24/11
https://doi.org/10.1145/3703790.3703791

1 INTRODUCTION

The Internet of Things (IoT) enables new means of interaction with the physical world via connectivity, sensing and actuation. A plethora of software-controlled *smart* devices equipped with sensors, actuators, and processors exist nowadays that enable the creation of complex IoT systems [2]. However, the software controlling these smart devices is often limited in its functionality, interfaces, and extensibility due to being closed source and proprietary [41]. We regard these as *legacy IoT systems* as they are rather inflexible and infeasible to modify [10]. Nevertheless, IoT systems are constantly evolving with new functionality being added, and new software and hardware being integrated into the systems, which raises problems with these inflexible legacy IoT systems [26].

In this work we investigate the application of *Business Process Management* (BPM) technologies in combination with *Microservices* to augment legacy IoT systems with new functionality in a non-invasive, lightweight process-oriented manner. BPM has proven its feasibility in scenarios related to enterprise application integration and service orchestration based on the modeling, automation, monitoring, and analysis of business processes for many decades [7], which is why we deem it suitable to be also applied in IoT-related integration and extension scenarios in combination with microservices [15]. The microservices architectural style has shown its flexibility and ability to extend software systems with small-footprint, functionally cohesive services in the software industry [30].

We start our discussions from a smart factory as legacy IoT system. We identify shortcomings based on our experience and an analysis of the factory's interfaces and capabilities. We generalize this missing functionality into functional requirements related to process monitoring and logging, error detection, and human-in-the-loop that an IoT system should fulfill. Thereby, the process-focused view proves to be a natural fit in IoT systems with automated, repetitive tasks, where human interventions are mostly needed in exceptional cases [15]. The new functional requirements guide us in augmenting the existing processes of the legacy IoT system. We discuss the application of suitable BPM concepts and, where necessary, the introduction of new microservices and software architectural aspects to showcase exemplary augmentations of the IoT system's processes. From these discussions, we propose a general process to augment the functionality of legacy IoT systems based on BPM technologies and microservices by adding a new process-oriented layer on top of the legacy system and microservices acting as intermediates between the BPM and legacy systems.

In this work we follow the design science research methodology proposed by Peffers et al. [29], which defines a structured approach to solve a recognized problem in Information System engineering

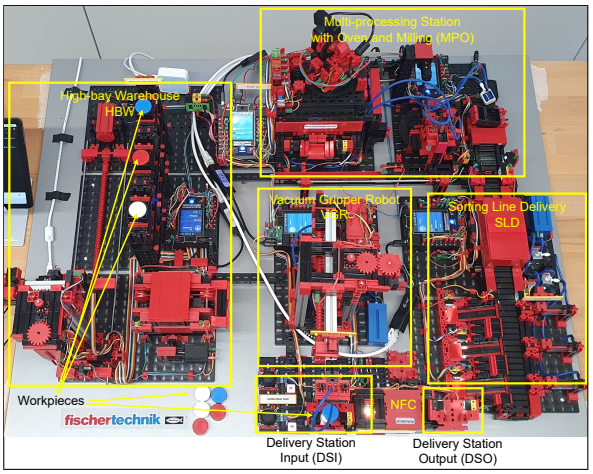

Figure 1: Smart factory setup used in this work.

Figure 2: Exemplary data streams from the MQTT interface.

and is typically applied in BPM research. We elaborate on the existing legacy IoT system and our experience with it in Section 2.1 to identify new requirements as objectives of a solution. In Section 3 we present the process of augmenting legacy IoT systems as main artifact derived from the smart factory use case. We discuss this artifact and its applicability in Section 4. Section 5 presents related work. Section 6 concludes the paper and shows future work.

2 BACKGROUND AND REQUIREMENTS

2.1 The "Smart" factory and its interfaces

The smart factory used as a legacy IoT system is a learning factory with components provided by Fischertechnik [21]. As depicted in Figure 1, it simulates a production line consisting of different production stations including a warehouse (HBW), vacuum gripper robot (VGR), sorting line (SLD), oven and milling station (MPO), each managed by an embedded controller with connected sensors and actuators. Small cylindrical, NFC-enabled workpieces of different colors are moved inside the production line to simulate processes.

In its default configuration, the smart factory offers an MQTT-based interface [35] to access the status of its operation in a publish-subscribe manner[1]. The messages contain IoT data at different abstraction levels including the status of all production stations, the warehouse stock, the status of an order, and the readings from an NFC sensor and environment station on different topics (cf. Figure 2). The only way to actively trigger an action in the factory is

[1]Technical description at https://github.com/fischertechnik/txt_training_factory

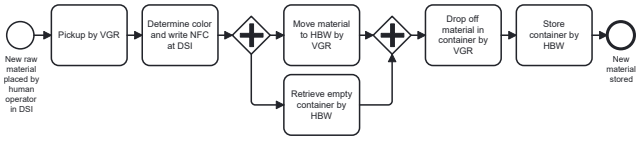

Figure 3: Storage process in the smart factory in BPMN 2.0.

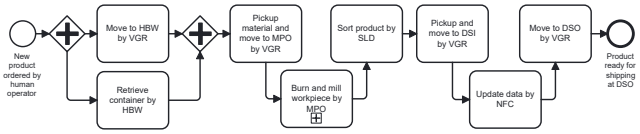

Figure 4: Production process in the factory in BPMN 2.0.

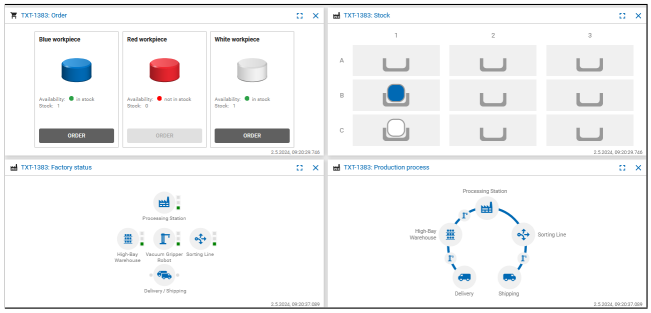

Figure 5: Dashboard for monitoring the factory operations.

to send an *order* message containing the color of the product to a specific MQTT topic, which starts the production process.

2.2 Processes in the smart factory

The factory features two basic processes that are hardwired in its programming. The first process supports the *storage* of raw material (cf. Figure 1): a new workpiece is put into the Delivery Station Input (DSI) by a human operator, automatically picked by the VGR, its color determined and data written on its NFC tag at DSI. Then, it is moved by the VGR to the HBW and an empty container is retrieved from the HBW in parallel, and finally it is handed over and stored in the HBW. Figure 3 shows a graphical representation of this process as a process model in BPMN 2.0 [27]. The second process implements the *production* of workpieces: an operator triggers a new product order, the corresponding raw material is unloaded from the HBW and the VGR moved in position. It is picked up by the VGR and moved to the MPO station to be burnt and milled. Then, it is sorted in the SLD, picked up by the VGR and moved to the NFC writer to update production data on its NFC tag, and finally moved in the delivery station output (DPO) by the VGR, where a human can pick up the product. Figure 4 shows a model of this process. Note that the two presented process models are just abstract representations of the processes for the purpose of communication. They cannot be instantiated to control the factory.

The smart factory provides a web-based dashboard (cf. Figure 5) that is subscribed to the MQTT interface described in Section 2.1 to monitor: (1) the execution of these two processes at very abstract

level, (2) the status of production stations, (3) the factory's additional sensors (e.g., NFC, environment, camera); and to initiate new orders.

2.3 Assumptions and requirements

In this work, we assume that the smart factory acts as legacy system. Its hardware-software setup and the MQTT interface are fixed and cannot be modified easily. We can only rely on the data provided via the interfaces, which only allow limited means of active control.

We have been working intensely with the default configuration of the smart factory for several years–primarily using it for research on BPM, IoT, and software engineering [21], and to educate our Bachelor and Master students in these topics. Based on our observations and experience with this smart factory, we derive the following new general requirements for this legacy IoT system as we identified important features that are currently missing in its configuration. During normal, error-free operations, the smart factory provides a good level of user experience via the dashboard (cf. Figure 5). The status of the individual stations (Fig. 5, middle left) and progress of the process execution (Fig. 5, middle right) can be monitored in this live view at an abstract level based on the symbolic representation of the production station that is highlighted as currently being active. However, to conduct more sophisticated BPM-related research and teaching focused on analyzing process executions and checking their conformance (i.e., process mining [38]), the following functional requirements (FRs) have to be additionally fulfilled to increase the quality of process-related data [4]:

FR1: Monitoring and tracking of activities. The monitoring and tracking of processes, especially *activities* in the processes, at runtime needs to be more fine-grained and detailed as there are often more relevant activities included in a process (cf. Section 2.2) compared to what legacy systems usually show (e.g., via a dashboard) [42]. Among others, this should include data about the start and end of an activity execution. The low-level data provided by the smart factory is not at the correct level of abstraction to monitor these executions, which is a typical problem of IoT systems providing too fine-grained or incomplete data for process monitoring and mining [3, 42]. To this end, domain experts should be enabled to define the activities to be monitored at the desired granularity level. The monitoring and tracking of multiple instances–not just one–of the same or a different process should also be possible.

FR2: Logging of process histories. Process analytics (mining) relies on *event logs* containing historical data about process executions [38]. In our experience, the smart factory and many other IoT settings produce incomplete and inappropriate process data, as these systems are typically not developed with a process-oriented view [4, 15]. The system should be able to create these logs with events containing complete data (e.g., according to XES [14]) and at the right level of abstraction for further process analysis [3, 42].

The user experience of the factory decreases when unexpected errors and exceptions occur in its operation [20]. These issues mostly relate to the factory's hardware, e.g., workpieces are not properly picked up or placed by the VGR, a machine gets stuck, workpieces have an incorrect orientation or fell down, or a wrong workpiece was loaded/unloaded from the HBW. These issues happen more frequently with the use and wear of the factory over time as it is only a low-cost simulation model with rather unreliable and imprecise hardware [21]. Moreover, software-related issues might occur, e.g., when ordering a product that is not in stock, the system simply does nothing. In all of these cases, the errors are not detected, the factory's operations just stop, and there is no user feedback. Similar issues in various error scenarios are likely to occur also in other IoT systems [11, 20], which adds new FRs that have to be fulfilled.

FR3: Error detection and handling. Error scenarios and exceptions in IoT systems are manifold and it is difficult to anticipate every error that may occur during process execution already in the original programming of the IoT system [22]. Thus, domain experts and process engineers should be able to specify potential error scenarios and corresponding remedies in the IoT systems themselves. These errors and exceptional behavior then need to be also detected in and handled by the system [22].

FR4: Human notifications and interactions. Operators should be notified about important events, especially about the occurrence of errors and exceptions in the IoT system's processes (e.g., a machine being unresponsive) as specified and detected based on FR3. Human intervention is a common pattern, which should be supported in the process execution when active control to resolve errors automatically is limited and problems have to be resolved manually [42]. Furthermore, human operators should have more means for interacting with the IoT systems, in normal operation mode and in error cases (e.g., to provide additional data or confirm the execution of manual steps), subsumed as *Human-in-the-Loop* [11].

The **goal** of our work is to address these new FRs for legacy IoT systems in a generic and lightweight manner through non-invasive software-based extensions–*augmentations*–to the existing IoT software systems without modifying their original implementations.

3 AUGMENTING THE LEGACY IOT SYSTEM

BPM provides methods and techniques for the modeling, automation, and analysis of processes in business contexts [7]. Among others, BPM has shown its strengths in enterprise application integration and service orchestration. Process modeling notations such as BPMN 2.0 [27] provide means for process engineers to model abstracted processes including activities, events, decision points, human interactions, exceptional situations, and timeouts–with execution semantics and in a domain-agnostic way [7]. The execution of these processes is supported by BPM systems (BPMS) featuring instantiation, monitoring, logging, state management, and versioning of business processes [7]. The application of BPM technologies in IoT has received increasing attention in recent years [15]. Processes play a central role in IoT systems such as smart factories where interactions among devices are complex, repetitive, and automated. Thus, we deem applying BPM viewing *Processes* as first class citizens in our work a promising direction to address the new FRs by adding business processes on top of existing IoT systems.

Microservices are a well established means in software engineering practice to define and implement small modular functionality as web services with high functional cohesion [30]. Apart from a flexible composition, integration, reuse, and orchestration (e.g., via BPMS), microservices allow for implementing business logic focusing on a well-defined and self-contained domain model within a *bounded context* (e.g., based on domain-driven design, DDD) [8]. Hence, microservices appear to be a suitable approach to extend

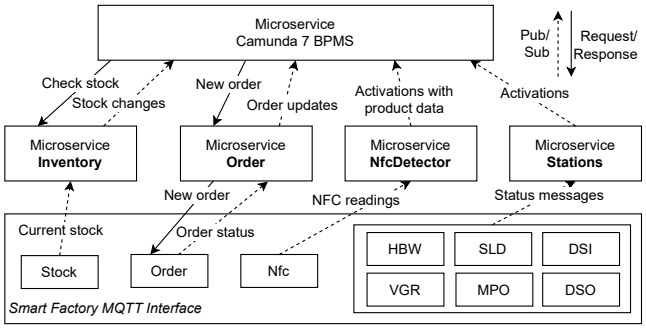

Figure 6: Software architecture and communication flow.

legacy IoT systems with additional functionality and more complex business logic, where the features of the business process modeling language and BPMS may not be sufficient [18].

3.1 Augmented business processes

We start our explanations with an overview of the augmented business processes. Both, the augmented storage process (cf. Figure 7) and the augmented production process (cf. Figure 8) are based on the simplified versions shown in Section 2.2. In contrast, these new augmented models can be instantiated and executed by a BPMS.

The *augmented storage process* starts with a human operator specifying the color of the new raw material and then putting the workpiece into the DSI station. A series of checks is performed to track the correct activation sequence of the individual activities from the original storage process (cf. Section 2.2). Timeout mechanisms are used together with several of these checks to determine if the VGR potentially failed to pickup the raw material or if the HBW got stuck, and then to inform the operator. Finally, the actual color of the new raw material is checked against the color specified in the beginning, notifying the operator in case of a mismatch.

The *augmented production process* starts with an operator specifying the color of the new product, followed by an inventory check to see if the corresponding workpiece is in stock once up-to-date inventory data is available. If not, the operator will be notified, otherwise a new production order is started in the factory. Several checks are then performed in the process to monitor and verify the correct sequence of production activities and changes in the factory. The operator is informed when production was successful, or when there was an issue with one of the activities based on timeouts.

3.2 Software architecture

To support the augmentations of the original processes in the factory and fulfill the new functional requirements FR1–FR4, we introduced new microservices 1) acting as intermediates (technical adapters) between the original MQTT-based interface and the BPMS, and 2) providing more complex business logic that was not practical to be realized via the business processes themselves. Figure 6 provides an overview of the software architecture including the smart factory's MQTT interface (bottom), the **BPMS** as new microservice (top), and four new **microservices** (middle) explained next. Note that due to the message-based interface of the smart factory and only limited means for active control (cf. Section 2.1), most of the communication is using the publish-subscribe (pub/sub) model to inform the microservices and BPMS [37].

- **Inventory** is subscribed to the topic with data about the current stock. It keeps a record of the latest stock and can be queried for the availability of workpieces of a specific type (i.e., color). It notifies the BPMS about changes in the stock, including the color of workpieces added or removed.
- **Order** is subscribed to the topic with messages about the status of order processing. The order service can be requested to send a command for initiating a new production, which will create a correlation between the production process instance and order processing in the factory. This correlation is used to notify the BPMS about progress updates regarding the specific production process instance.
- **NfcDetector** is subscribed to the MQTT topic with messages from the factory's NFC reader. It detects the activation of the NFC reader and notifies the BPMS about products on the NFC reader including product data (e.g., color).
- **Stations** is subscribed to the MQTT topics receiving messages about the status of the production stations. It detects when a station is active from the low-level IoT data and informs the BPMS about these activations.

3.3 FR1: Monitoring and tracking of activities

To achieve a more fine-grained monitoring of the process execution, we first modeled the activities of the storage and production processes in the augmented process models based on our observations of the original processes, similar to the models presented in Section 2.2. Additionally, we considered the data from the MQTT interface (cf. Section 2.1) in a *bottom up analysis* to decide about which activities can be detected and which additional, useful information can be derived based on the processing of provided data.

The primary source for data about the progress of processes is the *state* topic (cf. Figure 2), which receives the status of all stations. Based on this data we use *Service Tasks* [27] implemented by *External Task Workers* in the business processes to model the sequence of process activities to be activated as *checkpoints* in the two processes (e.g., *Detect new workpiece in input*, *Detect HBW activation* in Figure 7 or *Detect Oven Activation*, *Detect Sorting Activation* in Figure 8). Once the process execution reaches these service tasks, they wait to be completed by the external task worker, which is realized by the microservice *Stations* (cf. Figure 6). This service transforms the low-level IoT data into process-related events. Once this service detects the activation of one station, it will notify the BPMS to complete all instances of service tasks currently waiting for activation notifications for that particular station. These workers rely on a publish-subscribe model, which is reasonable here as the microservice will inform the waiting instances in the process execution once it detects the activation of the respective station.

By using BPMN 2.0 as the language to specify the augmented processes, the process engineer is able to define the level of detail for monitoring the process activities. Here, the data from the legacy system informing the activity executions needs to be analyzed to be potentially processed and enriched by microservices to reach the correct level of abstraction for monitoring activities at the specified level of detail [42]. By having this fine-grained representation

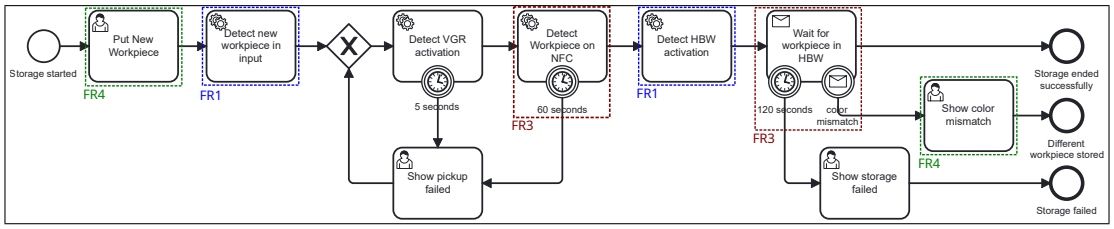

Figure 7: Storage process (in BPMN 2.0) with exemplary augmentations related to FR1–FR4.

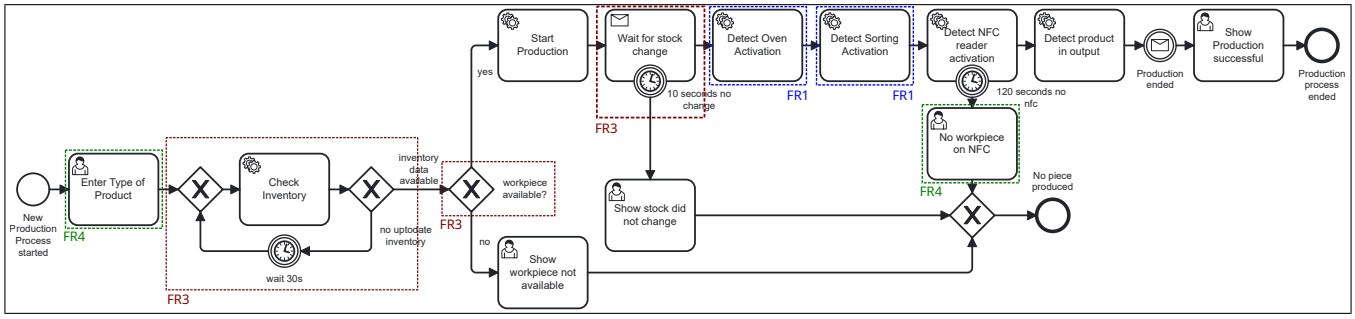

Figure 8: Production process (in BPMN 2.0) with exemplary augmentations related to FR1–FR4.

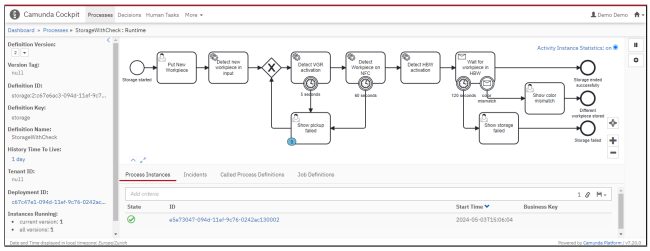

Figure 9: Cockpit to monitor process activities and instances.

of executable processes as modeled by the process engineer, and deployed in the BPMS, we can leverage the BPMS's built-in functionality to monitor the current execution of all business processes in a dedicated cockpit (cf. Figure 9), showing details about starting time of the instances and the currently active process activities indicated via tokens, at the specified granularity level.

3.4 FR2: Logging of process histories

Creating an event log of executed processes (*cases*) which contains events related to the execution of process activities (e.g., *start* event, *complete* event) is among the basic functionalities of a BPMS [7]. By using a BPMS to execute the modeled processes in the IoT system, we are automatically provided with an event log from the BPMS that can be used for process mining. The abstraction level of events corresponds to the granularity of activities as specified in the process models (cf. Section 3.3). In our setup, we use Camunda Platform 7[2] as BPMS embedded in a microservice (cf. Figure 6). This service offers a REST API to deploy process models, execute process instances, and inspect the histories (event logs).

[2]Camunda 7 Platform (Community Edition): https://camunda.com/platform-7/

3.5 FR3: Error detection and handling

Several BPM-related concepts are used in our augmented processes to detect and handle error situations that can be modeled by the domain expert. BPMN 2.0 features *exclusive gateways* with conditions to split the flow in a process based on process variables. Exclusive gateways make decisions explicitly visible in the models, which facilitates their understanding and tracking during process executions. However, if these decisions become too complex, the readability of the process models is negatively affected [32]. Approaches to model more complex decisions in business processes exist (e.g., DMN [12]), but they also increase the complexity of the models with additional artifacts [13]. Alternatively, these decisions can be encoded in the business logic of the microservices, but then they are hidden from the process models. In a trade-off discussion, we suggest to identify the decisions that are relevant to understand the processes, include them in the models, and let the microservices implement the logic to abstract these decisions at the same level as the rest of the process models [37]. We use exclusive gateways, e.g., in the production process to decide whether to start the production or inform the user regarding the unavailability of workpieces (cf. Figure 8) based on data abstracted by the *Inventory* service.

The service task retrieving the inventory is followed an exclusive gateway in a loop together with a *timer intermediate catch event* [27] configured to repeat every 30 seconds to check that inventory data is available. This is necessary as the *inventory* service is subscribed to the status of the stock, which does not update very frequently. Timer boundary events are also used with activities to monitor the progress of the processes. In both processes, activities with waiting behavior–*Service Tasks* with external task workers and *Receive Tasks* [27]–are extended with timeouts to activate a different branch in the processes in case the execution takes too long, which is an indication of unexpected behavior in the factory. We add timer

boundary events to activities where we often experience hardware-related issues (e.g., machines being stuck or unresponsive) or issues with handling the workpiece (e.g., the VGR not picking up the workpiece and the NFC reader not being activated later). As IoT systems manipulate the physical world, time-related patterns [19] and violations are suitable indicators for issues in the physical world processes [34], whose detection can be lifted to the BPM-level. While for the factory processes we defined rather simple error detection and handling, the augmentation of processes based on BPMN 2.0 also allows modeling and executing more sophisticated behavior. This includes workflow exception patterns [31], e.g., at the levels of work item and case, or deadline-based escalation measures.

Service tasks are completed based on activations from the *Stations* microservice (cf. Section 3.3). Similarly, we use the microservice *NfcDetector* to activate the waiting service tasks related to the NFC reader. The external task workers complete *all* waiting instances of a specific type of service task, which in our case is sufficient to track the activations of individual stations. In contrast, we use *receive tasks* to inform *one* specific activity instance waiting in a specific process instance to complete the task. The microservice *Inventory* detects changes in the factory's stock and informs the currently waiting receive task in a storage process instance about these changes, including emitting a *Message Boundary Event* when a mismatch between the color of the newly stored material and the originally intended color occurred (cf. Figure 7). To realize this more complex behavior, we decided to use a message-based receive task instead of a service task. The process engineer modeling the processes has to take this different behavior into consideration when deciding about the appropriate BPMN elements to use.

3.6 FR4: Human notifications and interactions

User Tasks are standard elements in BPMN 2.0 [27] to specify points of human interactions in business processes. In the augmented processes, user tasks are used to request input from the user (cf. Figure 10) and to inform about relevant events (e.g., an error). We apply these user tasks at the beginning of both processes to request input data from the user. In the storage process, the color of new material to be stored is requested and cross-checked with the stock update provided by the *inventory* service at the end. A user task is used to notify the user about a mismatch. In the production process we ask the operator to specify the color of the new product in a user task, which is used as input to the subsequent service task calling the *Inventory* service to check the availability of raw material. Once the order service detects the end of the production, it notifies the specific production process instance by activating the *message intermediate catch event* [27] "Production ended", which in turn notifies the operator via a user task (cf. Figure 8).

Moreover, we apply user tasks in the augmented processes at all points where the timer boundary events might interrupt an activity waiting too long for its completion. In these cases, human operators are notified via user tasks to check the status of the specific station for potential errors. As we do not have any additional means of actively controlling the factory, we have to rely on the human intervention pattern here, to manually check and fix the issue, cancel the process instance, and start a new process instance. Still, the processes of the legacy IoT system can be augmented this way

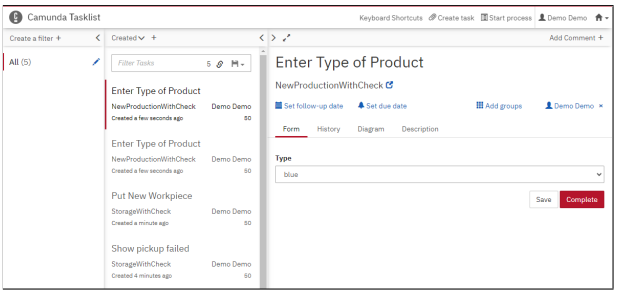

Figure 10: Task list with open, process-related tasks.

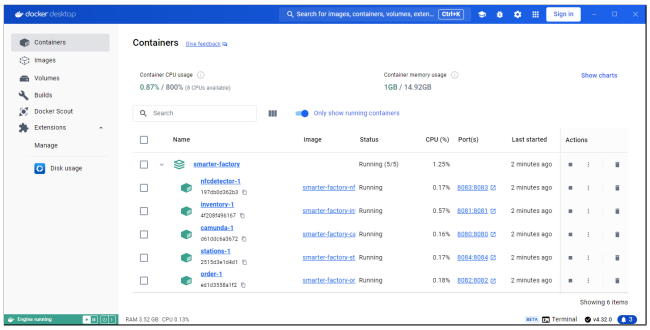

Figure 11: Docker configuration of the software system.

with more means of direct user interactions and feedback (e.g., to react to errors) in the processes as modeled by the process engineer.

3.7 Microservices Implementation

The microservices described in Section 3.2 were developed using Spring Boot following DDD [8]. They are layered, featuring: 1) a lightweight domain layer with business logic according to the services' responsibilities [18], 2) an application layer with application services, 3) a communication layer for the MQTT-based communication with the factory, and 4) a web service layer for communication among the services and the BPMS via REST. The entities in the domain model were generated from the documentation of the factory's JSON-based MQTT interface. The repository[3] contains the source code of all microservices and a configuration to start them as Docker containers (cf. Figure 11). This way, the standard configuration of the Fischertechnik factory model can be directly used with our augmented processes. The repository also contains recorded datastreams for exemplary executions of the original storage and production processes that can be used for simulation via MQTT.

3.8 Process to augment legacy IoT systems

From the elaborations in Sections 3.1–3.7 we propose the following generalized development process to augment legacy IoT systems using BPM and microservices (cf. Figure 12). Given a legacy IoT system with non-changeable interfaces, the process starts by defining new specific functionalities *top down* based on the general functional requirements FR1–FR4 that the IoT system's processes should fulfill [25]. This requirements elicitation can be based on, e.g., the

[3] https://github.com/ics-unisg/smarter-factory

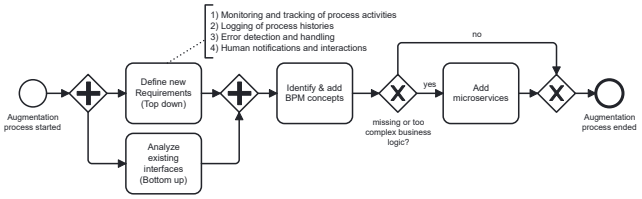

Figure 12: Proposed augmentation process in BPMN 2.0.

experience from working with the legacy system, integration scenarios, or new customer requests [5, 10]. In parallel, the interfaces and data provided by the legacy IoT system need to be analyzed *bottom up* to identify functionality and data that can be used and added to the processes to fulfill FR1–FR4, and to estimate the feasibility of addressing the new FRs derived in the top down-approach. Here, the developer also decides about the suitable granularity of process activity monitoring and the need for additional microservices that perform the data processing to reach this level of detail.

In a next step, suitable BPM concepts need to be identified and added to the models describing the business processes of the legacy IoT system to realize the new functionality. Here we refer to the discussions in Sections 3.2–3.6 to decide about using specific BPMN 2.0 elements and communication styles in the software architecture [37]. If the new requirements cannot be fulfilled by BPM concepts, additional microservices should be considered to realize required business logic and to act as technical adapters between the BPMS and legacy IoT system. Moreover, if the business logic in the process models becomes too complex and fine-grained, leading to complex models, it should be considered as part of a microservice. By using a process modeling language with execution semantics (e.g., BPMN 2.0), arbitrary business processes can be described using BPM concepts, and microservices can be arbitrarily invoked (orchestrated) to realize new functional requirements for the legacy IoT system without modifying its original programming. The augmentation process ends when all new FRs have been realized by a combination of BPM concepts and microservices.

4 DISCUSSION

In Section 3 we discussed factors and options that need to be considered when augmenting legacy IoT systems to fulfill new functional requirements, resulting in a proposal of a general augmentation process for legacy IoT systems. In this section we discuss additional aspects that are relevant to evaluate our proposal, namely the fulfillment of non-functional characteristics, its generalizability and limitations, and potential end-user evaluations.

Non-functional Characteristics. The main goal specified the approach to be *non-invasive* and *lightweight* as non-functional properties (cf. Section 2.3). Our proposed software architecture suggests to add a BPM-oriented layer on top of legacy IoT systems with microservices acting as mediators and business processes orchestrating the services. This way, we are able to flexibly and non-invasively extend the functionality of the legacy IoT system without the need of modifying the existing hardware or software. The implementation of the software containing the four microservices and embedded BPMS (cf. Section 3.2) comprises less than 1000 lines of Java code. Figure 11 shows the computing and memory consumption of running the software in Docker on a standard laptop (11th Gen Intel Core i7, 2.8 GHz, 4 Cores, 32 GB RAM, Win10). Based on these numbers, we attribute a relatively small footprint to the software system due to the lightweight nature of the containerized microservices. As among the relevant non-functional properties of a microservices-based architecture are *scalability*, *extensibility*, and *performance* [30], we expect the system to handle future growth in demand and functionality very well. However, we also acknowledge an increased need for communication among the microservices, the BPMS, and the legacy IoT system as indicated by the arrows in Figure 6, which represent points of communication. This might introduce coupling and impact the performance of the entire system negatively when the components are running in a distributed setup. Here the software architect needs to carefully decide about the deployment and specific points and styles of communication among the services and systems. Despite the possible communication overhead, we did not experience an increase in the execution times of the production processes in the smart factory as we do not introduce any delays and we only monitor the progress of the executions via the BPMS and microservices. These system only actively intervenes when the physical production processes appear to be already interrupted. The process-oriented approach to orchestrating microservices enhances the *flexibility* and *extensibility* of legacy IoT systems. Augmented processes (e.g., shown in Figures 7 and 8), can be integrated as sub-processes in larger business processes of a company. Additionally, process models can be easily modified, e.g., by adding new tasks or modifying the sequence flow, and the microservices can be reused in other processes modeling different orchestrations. To guarantee that flexibility and extensibility are maintained over time with BPM support, process model quality must be ensured. To this end, process modeling guidelines [1], and patterns modeling the normal operations [40] and the handling of exceptions [31, 39] should be adopted in the augmentation process.

Generalizability. Due to the general applicability of BPM and microservices to a plethora of domains, as demonstrated by multiple research works and industrial applications, we consider the proposed augmentation process to be highly generalizable and applicable to various IoT settings. In our implementation of the microservices, we already relied on tooling for generating the code of the microservices' domain models based on the provided interface documentation. We envision that the process of implementing the necessary microservices for bridging the systems and adding the required domain-dependent business logic can be automated even further by leveraging large-language models for partial code generation. Currently, we are applying the process to augment an IoT-based legacy system for monitoring manual treatment processes in a healthcare setting, with promising initial results. As a limitation we acknowledge that the augmentation process depends on the availability of interfaces and data of the legacy IoT system. The BPM-based retrofitting approach is not limited to specific protocols or data formats used by the legacy IoT systems. It assumes the existence of documented, standard-based or proprietary interfaces that can be accessed by external software components. The microservices are responsible for implementing the specific adapters in the communication layer to talk to the legacy systems and offer

service-based interfaces to the BPMS in the web service layer. IoT systems providing *no* usable data and interfaces preclude the augmentation process. In such cases, modifying the legacy system's software and *retrofitting* [16] by introducing additional hardware are potential interventions to enable the augmentation.

Evaluations with End-users. The functional requirements FR1–FR4 and extensions are mostly targeted at improving the interactions and experience of end-users with the legacy IoT system. This involves the monitoring and the analysis of processes in the legacy IoT system, which mostly adresses domain experts and process engineers. FR4 focuses on providing means for additional interactions to human operators working with or as part of the IoT system. The application of the augmentation process involves domain experts and process engineers that are familiar with BPM concepts, the legacy IoT system and its processes; and software engineers to develop the new microservices. While evaluations with all these user groups are reasonable, we propose to first evaluate the extent to which the approach supports the modeling of new processes or the enhancement of existing ones. To achieve this, a user study investigating the usability of our approach should be performed. A set of key metrics can be collected throughout this study to gain insights into the usability of BPMN-based modeling of augmentations and its effectiveness compared to the legacy IoT system. Following a between-subject design, participants can be divided into two groups, one with the augmented factory and one with the legacy system. Then, given a task where they are asked to apply extensions to a given process, we can collect data about the perceived usability/usefulness, performance, behavior, and mental effort (e.g., using eye-tracking) when performing the augmentation tasks.

5 RELATED WORK

Apart from classical retrofitting approaches adding new sensors to existing IoT systems [17], *smart* retrofitting has been proposed as a relatively inexpensive approach to integrate new technologies into legacy systems (cf. [16]). However, it results in invasive operations at both the hardware and software level [36], which for some legacy systems might not be possible. For our smart factory, for instance, we assume that modifying its existing hardware and software is not feasible, but still the goal is to augment and integrate the system with other components to extend its functionality. In contrast to approaches that aim to bridge IoT silos, and foster integration and loose interactions in different ways [23, 24], our work is concerned with IoT settings that have a strong focus on processes that predefine sequences of repetitive operational steps and that involve work to be done primarily by hardware and software components. Here we apply BPM technologies to model, monitor, execute, and extend the original processes from a business process perspective, which allows us to flexibly integrate new software services.

The adoption of microservices for the automation of distributed business processes is advocated in [18], where research challenges and a research agenda are discussed. Central to this agenda is the identification of significant use cases for microservice-based process execution. We argue that the smart factory presented in Sect. 2.1 constitutes a significant use case that highlights the necessity to augment legacy IoT systems. Given the increasing adoption of the factory for research and education purposes, we see its potential to become a cornerstone fostering collaboration to pursue the research agenda outlined in [18]. Approaches to IoT-enhanced business process modeling and executions are found in prior work such as [9, 28, 37]. Similar to our proposed approach, these works strive for process models of low complexity. However, they significantly differ from our approach in terms of the target system. While these works assume the possibility of modifying existing systems or designing new processes with fresh IoT integration, we consider legacy systems where no changes are possible in the existing implementations as a novel constraint, which has not been considered by related research. The work in [6] proposes to use business processes to orchestrate the execution of containerized microservices controlling IoT devices. The use case is different from ours as the IoT devices are only used to collect data and the microservices are used to perform distributed analytics on the data. In contrast, in our approach microservices and IoT play an active role in the process execution. The authors in [33] discuss a BPM-based approach to retrofit BPM systems themselves with self-x capabilities based on microservices. Their use case is also set in an IoT context, where the authors demonstrate the feasibility of using BPM and microservices for augmentations of existing systems, confirming our proposal.

6 CONCLUSION AND FUTURE WORK

In this work we presented a generic development process to augment legacy IoT systems with new functionality related to the modeling, monitoring and analysis of its processes, and to enable an improved interaction of human operators with the IoT system. BPM technologies prove to be a natural fit to realize this functionality, which we demonstrated by augmenting the original processes of a smart factory acting as legacy system. These augmented processes were complemented by microservices to 1) add missing business logic, 2) connect to the IoT systems, and 3) perform data processing to reach suitable levels of data abstractions. The introduction of a dedicated BPM layer on top of the legacy IoT system with microservices acting as intermediaries being orchestrated by a BPM system enabled the full range of BPM technologies to be applied for augmenting the IoT system. We are now able to flexibly model, extend, monitor and analyze processes of the legacy IoT system in a non-invasive way to improve the system's functionality and user experience in normal operation mode and in error cases.

As future work, we are planning to evaluate the functionalities and usability of the augmented smart factory including the application of the proposed process in a user study following the approach discussed in Section 4. Such an evaluation will involve students enrolled in our Masters course on software architecture as well as researchers and practitioners in software and process engineering. Furthermore, we will apply the proposed retrofitting process to legacy IoT systems of different domains (e.g., smart homes and smart healthcare) to evaluate it, both qualitatively with end-users and quantitatively (e.g., measuring the impact of the augmentations on software size, performance and resource consumption).

ACKNOWLEDGMENTS

This work has received funding from the Swiss National Science Foundation under Grant No. IZSTZ0_208497 (*ProAmbitIon* project).

REFERENCES

[1] Diego Toralles Avila, Rubens Ideron dos Santos, Jan Mendling, and Lucineia Heloisa Thom. 2020. A systematic literature review of process modeling guidelines and their empirical support. *Business Process Management Journal* 27, 1 (2020), 1–23.

[2] Martin Bauer, Nicola Bui, Jourik De Loof, Carsten Magerkurth, Andreas Nettsträter, Julinda Stefa, and Joachim W Walewski. 2013. IoT reference model. *Enabling things to talk: designing IoT solutions with the IoT architectural reference model* (2013), 113–162.

[3] Iris Beerepoot, Claudio Di Ciccio, Hajo A Reijers, Stefanie Rinderle-Ma, Wasana Bandara, Andrea Burattin, Diego Calvanese, Tianwa Chen, Izack Cohen, Benoît Depaire, et al. 2023. The biggest business process management problems to solve before we die. *Computers in Industry* 146 (2023), 103837.

[4] Yannis Bertrand, Jochen De Weerdt, and Estefanía Serral. 2021. A bridging model for process mining and IoT. In *International conference on process mining*. Springer, 98–110.

[5] Edwin Cabrera, Paola Cárdenas, Priscila Cedillo, and Paola Pesántez-Cabrera. 2020. Towards a Methodology for creating Internet of Things (IoT) Applications based on Microservices. In *2020 IEEE International Conference on Services Computing (SCC)*. 472–474.

[6] Tim d'Hondt, Anna Wilbik, Paul Grefen, Heiko Ludwig, Natalie Baracaldo, and Ali Anwar. 2019. Using bpm technology to deploy and manage distributed analytics in collaborative iot-driven business scenarios. In *Proceedings of the 9th International Conference on the Internet of Things*. 1–8.

[7] Marlon Dumas, L Marcello Rosa, Jan Mendling, and A Hajo Reijers. 2018. *Fundamentals of business process management*. Springer.

[8] Eric Evans. 2004. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional.

[9] Arianna Fedeli, Fabrizio Fornari, Andrea Polini, Barbara Re, Victoria Torres, and Pedro Valderas. 2024. FloBP: a model-driven approach for developing and executing IoT-enhanced business processes. *Software and Systems Modeling* (2024), 1–30.

[10] Omid Givehchi, Klaus Landsdorf, Pieter Simoens, and Armando Walter Colombo. 2017. Interoperability for Industrial Cyber-Physical Systems: An Approach for Legacy Systems. *IEEE Transactions on Industrial Informatics* 13, 6 (2017), 3370–3378.

[11] Dominic Gorecky, Mathias Schmitt, Matthias Loskyll, and Detlef Zühlke. 2014. Human-machine-interaction in the industry 4.0 era. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. 289–294.

[12] Object Management Group. 2019. *Decision Model and Notation (DMN) Version 1.3*. Object Management Group. https://www.omg.org/spec/DMN/1.3/

[13] Faruk Hasić, Estefanía Serral, and Monique Snoeck. 2020. Comparing BPMN to BPMN + DMN for IoT process modelling: a case-based inquiry. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing* (Brno, Czech Republic) *(SAC '20)*. Association for Computing Machinery, New York, NY, USA, 53–60.

[14] IEEE. 2023. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. *IEEE Std 1849-2023 (Revision of IEEE Std 1849-2016)* (2023), 1–55.

[15] Christian Janiesch, Agnes Koschmider, Massimo Mecella, Barbara Weber, Andrea Burattin, Claudio Di Ciccio, et al. 2020. The Internet of Things Meets Business Process Management: A Manifesto. *IEEE Systems, Man, and Cybernetics Magazine* 6, 4 (2020), 34–44.

[16] David Jaspert, Martin Ebel, Alexej Eckhardt, and Jens Poeppelbuss. 2021. Smart retrofitting in manufacturing: A systematic review. *Journal of Cleaner Production* 312 (2021), 127555.

[17] Sri Sudha Vijay Keshav Kolla, Diogo Machado Lourenço, Atal Anil Kumar, and Peter Plapper. 2022. Retrofitting of legacy machines in the context of Industrial Internet of Things (IIoT). *Procedia Computer Science* 200 (2022), 62–70.

[18] Agnes Koschmider. 2017. Microservices-based Business Process Model Execution.. In *RADAR+ EMISA@ CAiSE*. 158–161.

[19] Andreas Lanz, Barbara Weber, and Manfred Reichert. 2014. Time patterns for process-aware information systems. *Requirements engineering* 19 (2014), 113–141.

[20] Lukas Malburg, Florian Brand, and Ralph Bergmann. 2022. Adaptive management of cyber-physical workflows by means of case-based reasoning and automated planning. In *International Conference on Enterprise Design, Operations, and Computing*. Springer, 79–95.

[21] Lukas Malburg, Ronny Seiger, Ralph Bergmann, and Barbara Weber. 2020. Using physical factory simulation models for business process management research. In *International Conference on Business Process Management*. Springer, 95–107.

[22] Andrea Marrella, Massimo Mecella, and Sebastian Sardina. 2016. Intelligent process adaptation in the SmartPM system. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 2 (2016), 1–43.

[23] Simon Mayer, Andrei Ciortea, Alessandro Ricci, Maria Ines Robles, Matthias Kovatsch, and Angelo Croatti. 2018. Hypermedia to connect them all: Autonomous hypermedia agents and socio-technical interactions. *Internet Technology Letters* 1, 4 (2018), e50.

[24] Simon Mayer, Erik Wilde, and Florian Michahelles. 2015. A connective fabric for bridging internet of things silos. In *2015 5th International Conference on the Internet of Things (IOT)*. IEEE, 148–154.

[25] Rafael S Mendonca, Marenice Melo Carvalho, Guido Soprano Machado, Claudia Sabrina Monteiro Da Silva, Renan Landau Paiva De Medeiros, and Vicente Ferreira De Lucena. 2023. Development of the novel methodology to retrofit of the legacy system in context of Industry 4.0. *IEEE Access* (2023).

[26] Rebeca C. Motta, Káthia M. de Oliveira, and Guilherme H. Travassos. 2018. On challenges in engineering IoT software systems. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering* (Sao Carlos, Brazil) *(SBES '18)*. Association for Computing Machinery, New York, NY, USA, 42–51.

[27] OMG. 2011. Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/BPMN/2.0

[28] Jesús Ortiz, Victoria Torres, and Pedro Valderas. 2024. Combining Goal-Oriented and BPMN Modelling to Support Distributed Microservice Compositions.. In *ENASE*. 75–86.

[29] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A design science research methodology for information systems research. *Journal of management information systems* 24, 3 (2007), 45–77.

[30] Mark Richards and Neal Ford. 2020. *Fundamentals of software architecture: an engineering approach*. O'Reilly Media.

[31] Nick Russell, Wil Van Der Aalst, and Arthur Ter Hofstede. 2006. Workflow exception patterns. In *Advanced Information Systems Engineering: 18th International Conference, CAiSE 2006, Luxembourg, June 5-9, 2006*. Springer, 288–302.

[32] Clemens Schreiber, Amine Abbad-Andaloussi, and Barbara Weber. 2024. On the cognitive and behavioral effects of abstraction and fragmentation in modularized process models. *Information Systems* (2024), 102424.

[33] Ronny Seiger, Peter Heisig, and Uwe Aßmann. 2019. Retrofitting of workflow management systems with self-x capabilities for internet of things. In *Business Process Management Workshops: BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers 16*. Springer, 433–444.

[34] Aviral Shrivastava, Patricia Derler, Ya-Shian Li Baboud, Kevin Stanton, Mohammad Khayatian, Hugo A. Andrade, Marc Weiss, John Eidson, and Sundeep Chandhoke. 2016. Time in cyber-physical systems. In *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis* (Pittsburgh, Pennsylvania) *(CODES '16)*. Association for Computing Machinery, New York, NY, USA, Article 4, 10 pages.

[35] OASIS Standard. 2019. MQTT Version 5.0. *Retrieved June* 22 (2019), 2020.

[36] Dominik Tantscher and Barbara Mayer. 2022. Digital Retrofitting of legacy machines: A holistic procedure model for industrial companies. *CIRP Journal of Manufacturing Science and Technology* 36 (2022), 35–44.

[37] Pedro Valderas, Victoria Torres, and Estefanía Serral. 2022. Modelling and executing IoT-enhanced business processes through BPMN and microservices. *Journal of Systems and Software* 184 (2022), 111139.

[38] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter Van Den Brand, Ronald Brandtjen, Joos Buijs, et al. 2012. Process mining manifesto. In *Business Process Management Workshops: International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*. Springer, 169–194.

[39] Wil MP van der Aalst, Michael Rosemann, and Marlon Dumas. 2007. Deadline-based escalation in process-aware information systems. *Decision Support Systems* 43, 2 (2007), 492–511.

[40] Wil MP van Der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski, and Alistair P Barros. 2003. Workflow patterns. *Distributed and parallel databases* 14 (2003), 5–51.

[41] Bahtijar Vogel, Yuji Dong, Blerim Emruli, Paul Davidsson, and Romina Spalazzese. 2020. What Is an Open IoT Platform? Insights from a Systematic Mapping Study. *Future Internet* 12, 4 (2020).

[42] Barbara Weber, Amine Abbad-Andaloussi, Marco Franceschetti, Ronny Seiger, Hagen Völzer, and Francesca Zerbato. 2024. Leveraging Digital Trace Data to Investigate and Support Human-Centered Work Processes. In *Evaluation of Novel Approaches to Software Engineering*. Springer Nature Switzerland, Cham, 1–23.