Chapter 6 Developing Process Execution Support for High-Tech Manufacturing Processes

Irene Vanderfeesten, Jonnro Erasmus, Konstantinos Traganos, Panagiotis Bouklis, Anastasia Garbi, George Boultadakis, Remco Dijkman, and Paul Grefen

Abstract This chapter describes the development of an information system to control the execution of high-tech manufacturing processes from the business process level, based on executable process models. The development is described from process analysis to requirements elicitation to the definition of executable business process, for three pilot cases in our recent HORSE project. The HORSE project aims to develop technologies for smart factories, making end-to-end high-tech manufacturing processes, in which robots and humans collaborate, more flexible, more efficient, and more effective to produce small batches of customized products. This is done through the use of Internet of Things (IoT), Industry 4.0, collaborative robot technology, dynamic manufacturing process management, and flexible task allocation between robots and humans. The result is a manufacturing process management system (MPMS) that orchestrates the manufacturing process across work cells and production lines and operates based on executable business process models defined in BPMN.

6.1 Introduction

The manufacturing domain is moving toward more dynamic situations [11, 20, 39]. High-tech manufacturing companies currently face compelling challenges to increase efficiency and productivity while the products they produce become more complex and more customized, order batches become smaller, and delivery times need to be shortened to stay competitive [9, 26]. This requires new ways of

I. Vanderfeesten (✉) · J. Erasmus · K. Traganos · R. Dijkman · P. Grefen
School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, Netherlands
e-mail: i.t.p.vanderfeesten@tue.nl; j.erasmus@tue.nl; k.traganos@tue.nl; r.m.dijkman@tue.nl; p.w.p.j.grefen@tue.nl

P. Bouklis · A. Garbi · G. Boultadakis
European Dynamics, Athens, Greece
e-mail: panagiotis.bouklis@eurodyn.com; anastasia.garbi@eurodyn.com; george.boultadakis@eurodyn.com

D. Lübke, C. Pautasso (eds.), *Empirical Studies on the Development of Executable Business Processes*, https://doi.org/10.1007/978-3-030-17666-2_6

organizing their factories and explicit process management to deal with this required flexibility and dynamism [8].

This chapter describes the development of an information system to control the execution of high-tech manufacturing processes, based on executable process models. The system is part of the development of the EU H2020 research project HORSE [36]. The HORSE project aims to develop innovative technologies for smart factories, making end-to-end high-tech manufacturing processes, in which robots and humans collaborate, more flexible, more efficient, and more effective to produce small batches of customized products. This is done through the use of Internet of Things [25], Industry 4.0 [22], collaborative robot technology, dynamic manufacturing process management, and flexible task allocation between robots and humans.

The HORSE solution aims at bridging the IoT-oriented level of collaborative robotics and the BPM-oriented level of manufacturing processes, i.e., coupling the physical and event layers of business control with the process layer [18]. With this, it tries to answer the question whether real-time manufacturing processes can be supported and coordinated with business process management (BPM) technology. Current research in this area has focused on modeling [40] and simulation [30] of manufacturing processes and analyzing their performance without providing execution support. At best, previous efforts have demonstrated importing business process models into manufacturing execution systems to support process execution [10, 29].

As part of the overall HORSE solution, the project aims to deliver an exaptation [19] of contemporary BPM technology to the manufacturing domain and to integrate this with new technological developments on the factory floor such as collaborative robots, automated guided vehicles (AGVs), and augmented reality to support human operators. The backbone of the HORSE system is a process management system that orchestrates the manufacturing process across work cells and production lines and that operates based on executable business process models. It is termed manufacturing process management system (MPMS).

The development of the MPMS is described in this chapter, from physical structure and manufacturing process analysis to requirements elicitation to the definition and enactment of executable business processes and to a first evaluation of the system. The basis for this are the three pilot cases in the HORSE project which we treated as explorative case studies [7, 33] here. In the following sections, first, the case study approach is explained, followed by a process and requirements analysis of the case studies that leads to a general HORSE requirements framework. Next, the architecture of the system is outlined, and the executable process models are presented and reflected upon, giving detailed insights on the particular challenges we met to transform the process models developed for business analysis into executable process models. The chapter is concluded with some takeaways for researchers and practitioners and an outlook on future developments.

6.2 Approach

In order to develop the HORSE system, we used a systematic approach to analyze the current manufacturing processes, bottlenecks, potential redesigns, and requirements to the HORSE system and to develop a generic solution (see Fig. 6.1). A thorough analysis of the three pilot cases is at the core of this approach. Through interviews and observations, we first modeled their manufacturing structure and processes and then performed a systematic analysis to identify problems, bottlenecks, and challenges. The three pilot cases can therefore be seen as exploratory case studies [7, 33] that in the end led to a general requirements framework for the system to be developed. According to [7], a case study is *a technique for detailed exploratory investigations, both prospectively and retrospectively, that attempts to understand and explain phenomenon or test theories, using primarily qualitative analysis*. With these pilot cases, we intended to prospectively test whether exapted BPM theory and technology is suitable for the manufacturing domain.

Each case study is analyzed in the same systematic way:

1. **Physical hierarchy analysis**. Like an organizational chart, considering a manufacturing company from a physical hierarchy perspective is a useful starting point to identify and scope a problem. As a basis, we use the reference physical hierarchy of the widely adopted international standard IEC 62264:2013 [3]. The reference hierarchy provides a clear distinction between levels of decomposition that can be expected in a factory. A site is usually comprised of multiple production areas, which can in turn be comprised of multiple production lines. Figure 6.2 shows the reference physical hierarchy. Participants in the case

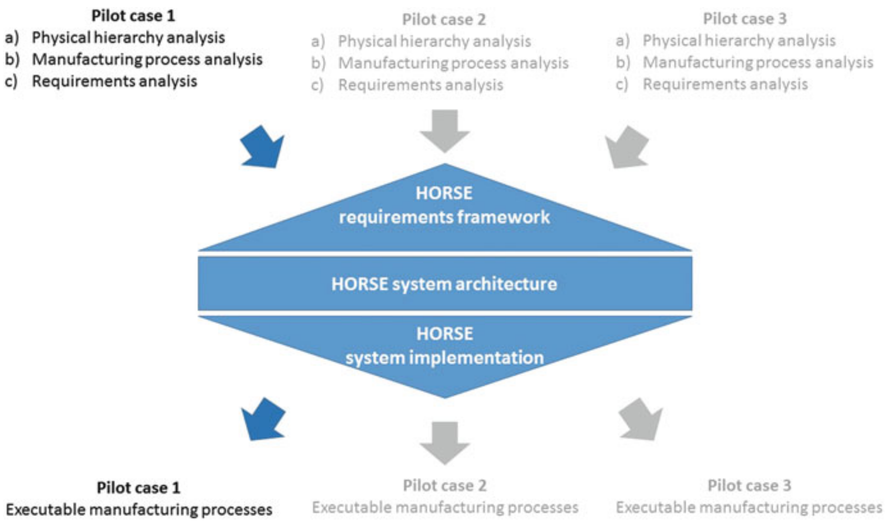
Pilot case 1
a) Physical hierarchy analysis
b) Manufacturing process analysis
c) Requirements analysis
Pilot case 2
a) Physical hierarchy analysis
b) Manufacturing process analysis
c) Requirements analysis
Pilot case 3
a) Physical hierarchy analysis
b) Manufacturing process analysis
c) Requirements analysis
HORSE requirements framework
HORSE system architecture
HORSE system implementation
Pilot case 1
Executable manufacturing processes
Pilot case 2
Executable manufacturing processes
Pilot case 3
Executable manufacturing processes

Fig. 6.1 Illustration of the case study approach

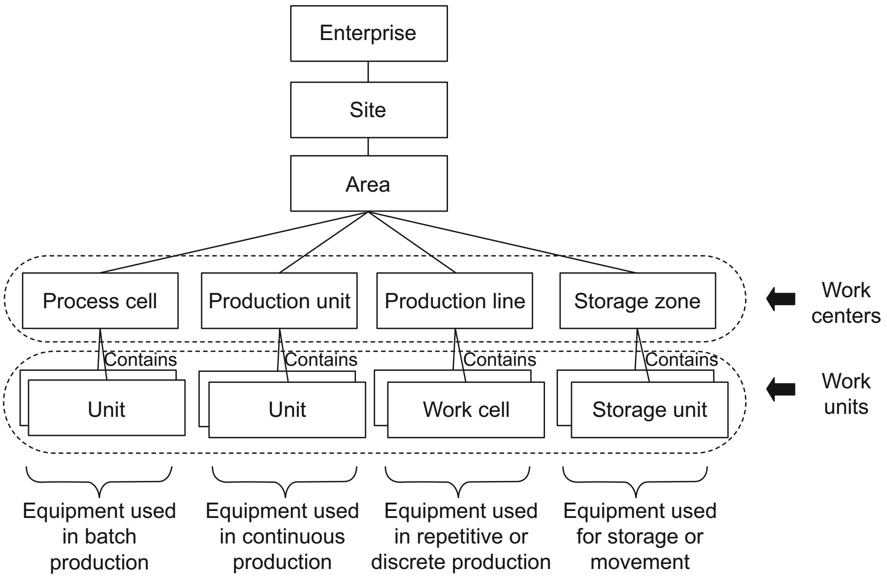
Enterprise
Site
Area
Process cell
Production unit
Production line
Storage zone
Work centers
Contains
Contains
Contains
Contains
Unit
Unit
Work cell
Storage unit
Work units
Equipment used in batch production
Equipment used in continuous production
Equipment used in repetitive or discrete production
Equipment used for storage or movement

Fig. 6.2 Reference physical hierarchy for a manufacturing enterprise [21]

study analysis workshops found the top-down, hierarchical perspective easy to understand and relate to. It also benefits from an almost complete absence of notation that may be open to interpretation. Finally, the physical structure of the enterprise helps to understand the scope of the processes in the enterprise and plan the layout of process models.

2. **Manufacturing process analysis**. Manufacturing is fundamentally a process that transforms material and energy into a product. A process-oriented view therefore provides substantial information about the manufacturing system. This is especially true for a manufacturing system with a significant degree of dynamism in its processes. The dynamism is a response to the market: demand fluctuation and mass customization implore the factory to adjust its volume and frequently change its operations to produce small batches of varied products. A process model of such a manufacturing system is able to express significant information without overextending the capabilities of a model. It also helps to pinpoint the problematic point in a sequence of activities and to understand the impact of that problem on upstream or downstream activities. Process models have been shown to be useful tools to foster understanding and agreement between people with diverse backgrounds [6]. In this analysis, we adopt the BPMN 2.0 standard for modeling manufacturing processes [28, 35].

3. **System requirements analysis**. Process models can be used to effectively compare the current situation with proposed future situations. Such side-by-side comparison of the current and future scenarios is an excellent facilitator of requirements elicitation. Requirements should be stated from the perspective of the system itself, of which the process is a core aspect. Furthermore, we recommend standardized requirements specification, such as the syntax proposed by Mavin et al. [27]: «optional preconditions»«optional trigger» the «name» shall «system response». For example, the introduction of a robotic arm to an assembly activity may lead to the following requirement statement: the system shall assemble 20 products per minute. Such a requirement statement refers to the system as a whole, i.e., the future scenario where the robotic arm is an actor in the process.

Next, the elicited requirements coming from the three case studies were compared, combined, and structured in a HORSE requirements framework. The architecture of the HORSE system was designed (first from a functional perspective [14] followed by a more technical software perspective), and a prototype implementation (realization) of the HORSE system satisfying the elicited requirements was developed. Then, the pilot cases were elaborated into executable processes, and the system was deployed and evaluated on the factory floor.

The developed system builds on BPM technology (as is further explained in Sect. 6.4) and is driven by an executable business process that is a refinement of the process models resulting from the business analysis and elicitation phase. In the next sections, this approach is illustrated by a detailed elaboration of one of the pilot cases. Due to space limitations, the detailed analysis of the other two pilot cases is not included here, but an illustration can be found in the online appendix of this chapter.[1] The lessons learned on how to develop executable processes from business-level process models based on our experience with the three cases are discussed afterward.

6.3 Case Study Analysis and Requirements Elicitation

For each of the three case studies, we followed the three-step analysis approach as described in the previous section. In the next sections, these steps are detailed for the first case study, with a main focus on the process models used in this phase. Due to space limitations, the other two pilot cases are not discussed here, but additional information can be found in the online appendix (see footnote 1). The process models for all three pilot cases can be described along the lines of Lübke et al.'s [4] process classification (see Table 6.1).

[1] http://is.ieis.tue.nl/staff/ivanderfeesten/Papers/ESDEBP2018/.

Table 6.1 Process classification of the three case studies according to the template in Chap. 2

Process name	Case study 1	Case study 2	Case study 3
Version			
Domain	High-tech manufacturing		
Geography	Netherlands	Poland	Spain
Time	2015–2017		
Boundaries	Intraorganizational/Within department		
Relationship	Event triggered		
Scope	Business scope: core		
Process model purpose	Descriptive		
People involvement	Partly		
Process language	BPMN 2.0		
Execution engine	NA		
Model maturity	Reviewed		

6.3.1 Case Study 1

The first case study is a small high-tech factory in the south of the Netherlands. This factory produces make-to-order [12, 31, 32] sliders that allow cupboard drawers to extrude and retract. These highly customizable sliders comprise of several metal profiles and ball bearings. The metal profiles undergo various cutting, bending, and surface treatment operations, depending on the specific requirements of the customer. The final step in production is assembly of the treated profiles and ball bearings, before the sliders are placed in packaging for delivery. The company faces a huge challenge for the future: to stay competitive, they have to make production more agile and flexible to deal with an increasingly high level of customization of their products, smaller batches, and a decrease in accepted delivery time.

Physical Hierarchy As the first step in the case study analysis, the physical decomposition of the factory is described according to the physical hierarchy of IEC 62264:2013 [21]: the factory has three production areas (PAs) and a single storage zone that acts as a buffer between the production areas. Figure 6.3 shows a depiction of this decomposition. The case study covers all three production areas and the storage zone. However, in the interest of brevity, only two processes will be discussed in detail here: *tool assembly* in production area 1 (PA1) and *loading* in production area 2 (PA2).

Process Analysis As the second step in the case study analysis, the end-to-end manufacturing process is modeled in BPMN 2.0. Such an end-to-end process model is particularly useful to understand the upstream or downstream impact of a problem. The three production areas are modeled as separate pools in Fig. 6.4. This is due to a significant difference of throughput. PA1 and PA3 have much higher capacity than PA2. Therefore, the output parts from PA1 are stored in a temporary buffer before they are processed at PA2. The storage zone is not explicitly included in the process model, but rather implied as external preservation service. The operator places items in storage or retrieves them from storage.

The two processes discussed in this chapter are highlighted with bold borders in Fig. 6.4. The first process occurs in PA1 and is concerned with the preparation of tools for cold forming. These tool assemblies consist of a base plate and at least ten tooling parts. Hundreds of tooling parts are available, resulting in thousands of potential tool configurations. Figure 6.5 shows a model of the current process to assemble tooling blocks. It should be noted that this process corresponds to the PL1-1 tool assembly production line and its two work units in Fig. 6.3: WU1.1.1 single tool assembly and WU1.1.2 tool set assembly. Therefore, the single process model includes work done in two distinct work stations. This process is currently entirely manual and depends largely on the experience and skills of operators. In the future situation, the assembly activity is supported with an augmented reality system and a robot fetching the tooling parts that need to be assembled from the warehouse (Fig. 6.6).

The second process occurs in PA2—the surface treatment of the parts. Profiles coming from PA1 are loaded on (and after treatment unloaded from) racks that go into a chemical bath. This is an entirely manual activity done by human operators. It demands concentration and places physical strain on the operators. This process will undergo the most significant changes as part of the HORSE project solution and is discussed here in detail. More specifically, the *loading* process will be discussed and is highlighted with a bold outline in Fig. 6.4. Figure 6.7 shows the next level of detail of the *loading* process in PA2.

Figure 6.7 depicts a largely manual process (as indicated by many manual and user tasks), at odds with the general trajectory of automation seen in the manufacturing industry. The lack of automation is due to the product customization. The product dimensions may be different for each production order, which also affects the way profiles are loaded for surface treatment. Robotic solutions have difficulty with such highly customizable situations.

Instead of attempting to replace the human with a robot, the process as a whole should be considered. The operator has two main tasks in this process: picking a handful of profiles and then hanging each profile individually. Splitting the responsibility of those two tasks makes the potential (robotic) solution simpler and more manageable. One robot can pick the profiles (*Place parts on conveyor belt*), while the other selects the right gripper for the profile (*Change gripper*) and hangs each profile on the rack (*Grab and hang single profile*). Figure 6.8 shows how the proposed solution will change the process.

Requirements Analysis The end-to-end manufacturing process model (see Fig. 6.4) is a valuable tool to identify broader requirements. In this first case study, it can be seen that the end-to-end process is divided into three individual pools (as also indicated by the physical hierarchy in Fig. 6.3). This is to allow asynchronism between parts of the process, realized by buffered storage between the separate production areas. However, it is still necessary for a production order to make its way through all three production orders. Thus, a production order should

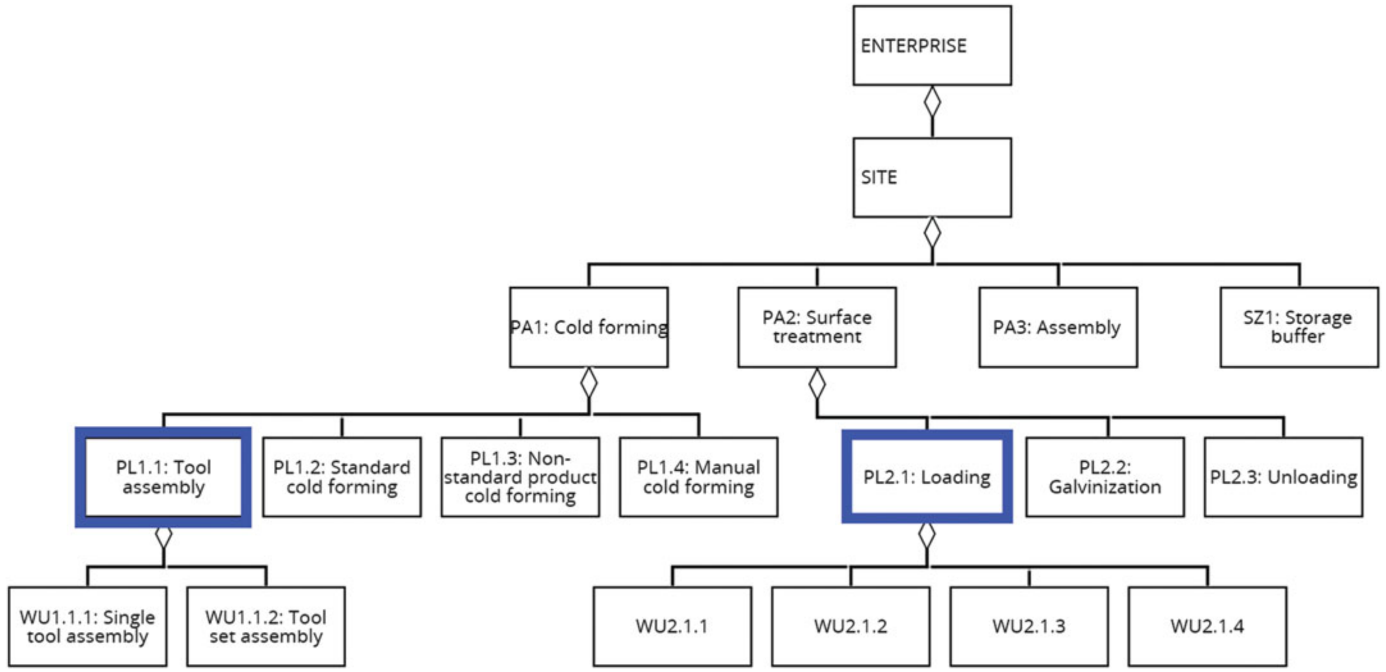

Fig. 6.3 Physical hierarchy of the manufacturing company of case study 1. The highlighted processes will be discussed in detail in the remainder of this chapter

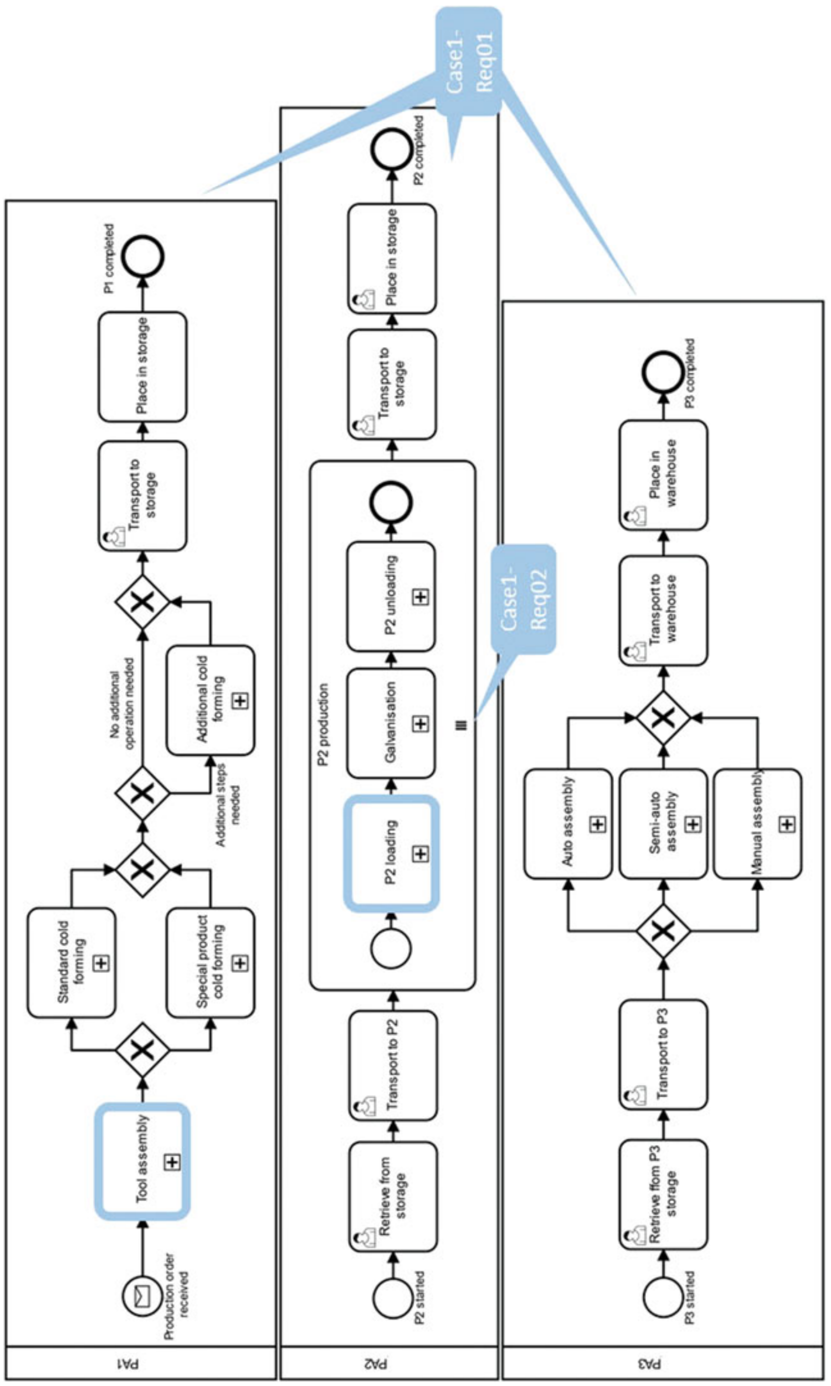

Fig. 6.4 Process model of the end-to-end production process for case study 1

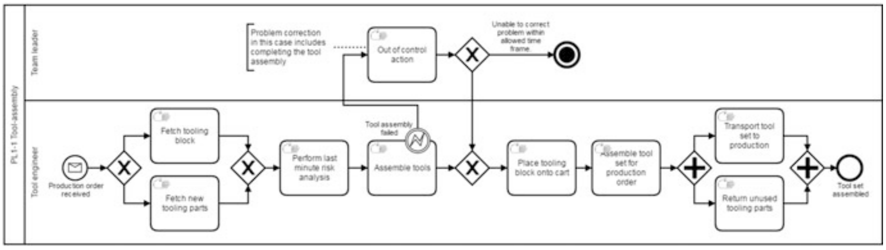
PL1-1 Tool-assembly
Team leader
Tool engineer
Problem correction in this case includes completing the tool assembly
Out of control action
Unable to correct problem within allowed time frame
Production order received
Fetch tooling block
Fetch new tooling parts
Perform last minute risk analysis
Tool assembly failed
Assemble tools
Place tooling block onto cart
Assemble tool set for production order
Transport tool set to production
Return unused tooling parts
Tool set assembled

Fig. 6.5 Process model of the current PA1 tool assembly process (PL1-1)

be managed as a case, with its own life cycle and requirements. This requirement can be expressed as the following: *Case1-Req01: The HORSE system shall manage and coordinate all activities for an individual production order, based on predefined requirements of that order.*

Additionally, the physical nature of manufacturing entails that some activities must be repeated to achieve the desired results. For example, the surface of a product may require several treatments to achieve expected quality. Batch processing also entails some multiplicity, where the constituents of a batch undergo individual processing. This can be seen in PA2 of case study 1, where a batch of profiles needs to be loaded on the rack one by one. A batch is divided into groups, limited by the size of the work units. Thus, a single case may spawn multiple instances of a product, depending on the requirements of the order. This requirement can be expressed as the following: *Case1-Req02: The process management system shall manage activity multiplicity based on predefined production order requirements.*

Requirements can also be elicited by comparing the current and future process models. Requirements can be identified from three aspects of the process model: (1) changes to the process as a whole, (2) changes to the roles of resources involved in the process, and (3) changes to individual tasks in the process.

For individual tasks or events, it is necessary to consider how they will be performed or triggered. In our case study, the tasks involved in the tool assembly may fail for a number of reasons, such as unavailability of necessary parts or unacceptable quality. If this happens, the operator must be able to trigger the failure exception shown in Fig. 6.6 to initiate the out-of-normal action task. This can be specified as the following requirement: *Case1-Req03: The process management system shall manage process exceptions by initiating a predefined response.* Additionally, task repetition is an important concept in manufacturing. The first task of the Placing Robot resource, as shown in Fig. 6.8, is denoted as a multi-instance activity. The robot must repeat the task for each profile in the batch. The following requirement can be inferred from this phenomenon: *Case1-Req05: The HORSE system shall queue tasks to be performed by the same resource.*

Second, we can elicit requirements related to specific resources involved in the process. As can be seen in Fig. 6.6, three resources are involved in the future process, as opposed to only two resources in the current process shown in Fig. 6.5. The

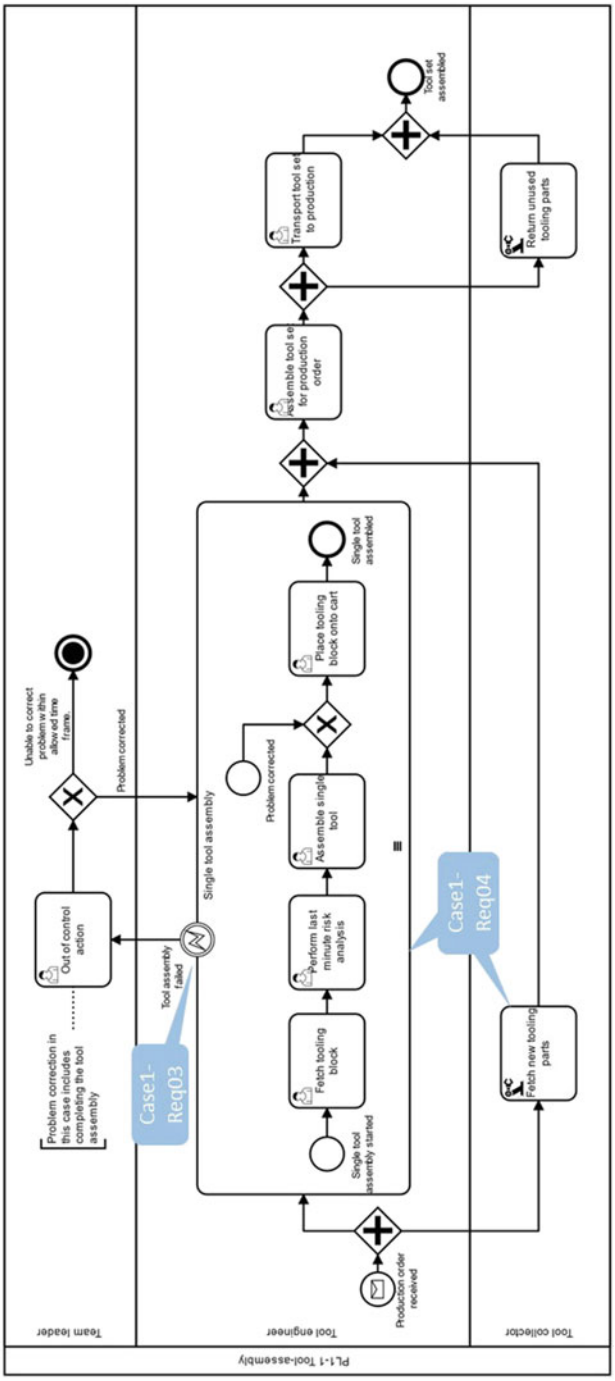
PL1-1 Tool-assembly
Team leader
Tool engineer
Tool collector
Production order received
Single tool assembly
Single tool assembly started
Fetch tooling block
Perform last minute risk analysis
Assemble single tool
Problem corrected
Place tooling block onto cart
Single tool assembled
Tool assembly failed
Out of control action
Problem correction in this case includes completing the tool assembly
Problem corrected
Unable to correct problem within allowed time frame.
Case1-Req03
Case1-Req04
Fetch new tooling parts
Assemble tool set for production order
Transport tool set to production
Return unused tooling parts
Tool set assembled

Fig. 6.6 Process model of the future PA1 tool assembly process (PL1-1). Please note that we used a BPMN extension to model robotic tasks from a business-level perspective, as defined by Aspridou [1]

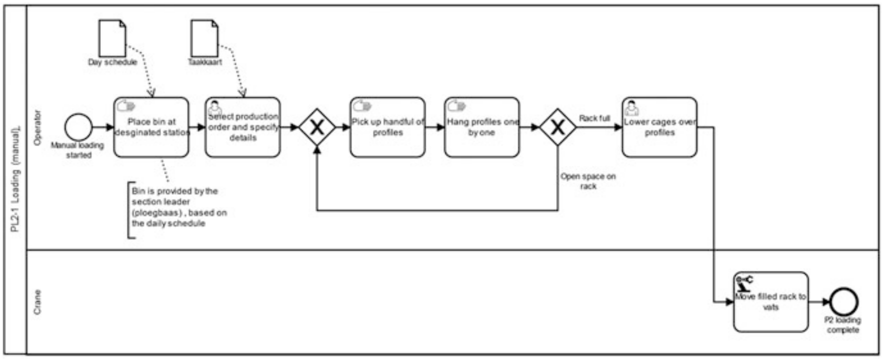
PL2-1 Loading (manual)
Operator
Crane
Day schedule
Taakkaart
Manual loading started
Place bin at desginated station
Select production order and specify details
Pick up handful of profiles
Hang profiles one by one
Rack full
Lower cages over profiles
Open space on rack
Bin is provided by the section leader (ploegbaas), based on the daily schedule
Move filled rack to vats
P2 loading complete

Fig. 6.7 Process model of the current PA2 (manual) loading process (PL2-1)

activities of these resources must be coordinated to ensure proper process execution. For example, the robot should not start removing tooling parts that the engineer still needs. This requirement can be specified as such: *Case1-Req04: The process management system shall coordinate the activities of multiple actors involved in the same manufacturing process.* As for individual resources, the third resource in Fig. 6.8 is named Loading Robot and is responsible for hanging profiles on the rack. Thus, we can infer the following requirement for this resource: *Case1-Req06: The Loading Robot shall be able to lift parts with a maximum mass of 5 kg.*

For the process as a whole, it may be necessary to specify requirements related to the nature or scope of the process. In our case study, multiple production orders may be active at the same time, or a single production order may require multiple executions of the same (sub)process. This is handled by creating multiple instances of a process definition, as indicated for PA2 in Fig. 6.4. Thus, multiple instances of the entire process shown in Fig. 6.8 can be active at the same time. Finally, activity automation often delivers improved performance, at the expense of safety to humans. In this case study, the human operator must still perform tasks near the two robots. Thus, the robots must include safety features to protect the human. We therefore define the following requirement: *Case1-Req07: The system shall actively monitor its proximity to ensure no harm is done to the human.*

6.3.2 General Requirements Framework

As described in the previous sections, requirements elicitation at three unrelated factories naturally yielded a diverse set of requirements, connected through the overall wish to control the execution of the process on a higher abstraction level, overseeing the individual machines, operators, and robots. The project endeavors to develop a single package of technologies that makes advanced manufacturing technology more accessible to small and medium factories. Thus, it is necessary

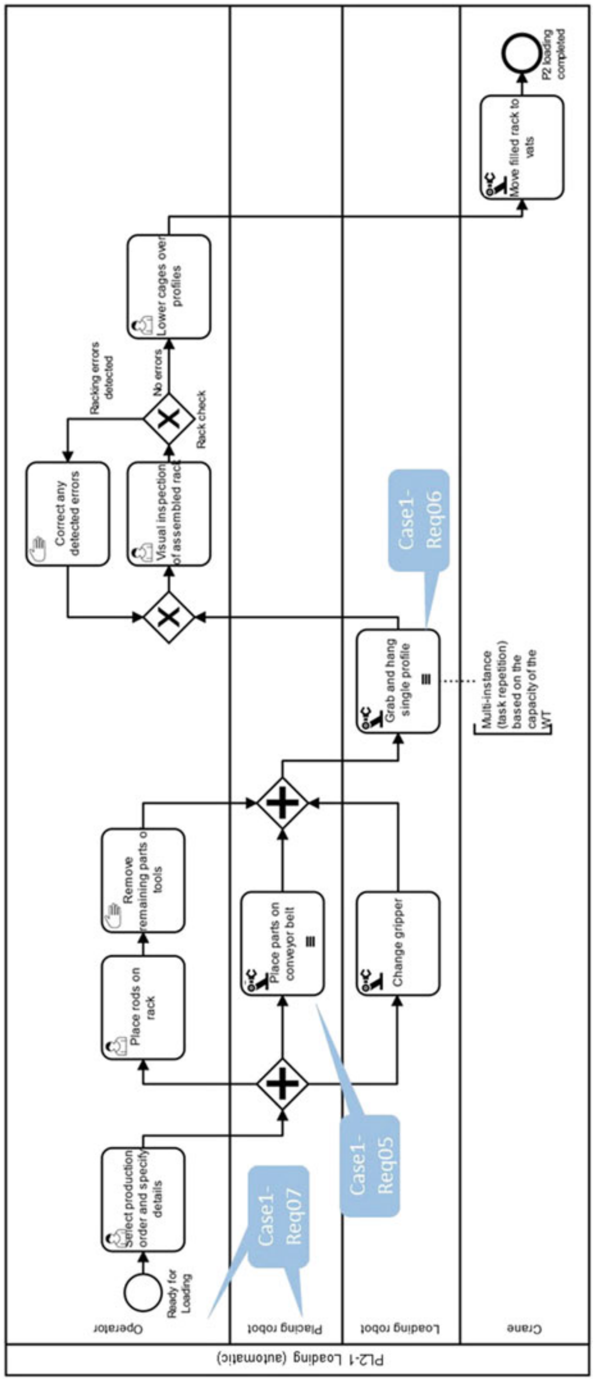
PL2-1 Loading (automatic)
Operator
Placing robot
Loading robot
Crane
Ready for Loading
Select production order and specify details
Place rods on rack
Remove remaining parts of tools
Place parts on conveyor belt
Change gripper
Grab and hang single profile
Multi-instance (task repetition) based on the capacity of the WT
Correct any detected errors
Visual inspection of assembled rack
Rack check
Racking errors detected
No errors
Lower cages over profiles
Move filled rack to vats
P2 loading completed
Case1-Req07
Case1-Req05
Case1-Req06

Fig. 6.8 Process model of the future PA2 (automatic) loading process (PL2-1). Please note that we used a BPMN extension to model robotic tasks from a business-level perspective, as defined by Aspridou [1]

to amalgamate the requirements into a more homogeneous and consistent set. Without homogenization, the project will risk developing disparate and case-specific subsystems that do not support integration or generalizability to other situations.

The requirements were analyzed and categorized to find common concepts. The requirements framework shown in Fig. 6.9 is the result of this analysis. The left part of this framework deals with developments in the robotics domain to better integrate human and robotic activities. For the purpose of this chapter, we do not need to go into the details of these developments; but for the interested reader, we refer to [36] for more detailed information on these developments.

The left part of the taxonomy in the figure (i.e., AF-01 to AF-04) shows the required functions for automated support of integrated activities in individual manufacturing work cells. The right part of the taxonomy (i.e., PF-01 to PF-03) shows the main required functions for horizontal and vertical process integration, meaning that the end-to-end manufacturing process shall be monitored and managed and that work cells shall be controlled and coordinated via the horizontal process management. We can relate these two sets of requirements to the AFIS four-layer reference framework for flexible information systems [18]. In this reference framework, the AF-labeled functions correspond to the management of physical entities (things in the sense of the Internet of Things) and the management of digital events (related to activities of these physical entities), so to the physical layer and event layer of the framework, respectively. The PF-labeled functions correspond to the process layer of the framework. The business layer of the framework is not covered by the requirements.

The main AF and PF functions are again decomposed into more detailed system functions (SFs) and into a large set of concrete requirements. For instance, Case1-Req01 falls under PF-01/SF-11, and Case1-Req05 belongs to AF-02/SF-07. Table 6.2 provides a mapping of the requirements included in this chapter to the system functions and main functions of Fig. 6.9. Many more requirements are specified for the HORSE system but are omitted due to confidentiality restrictions.

The HORSE requirements framework then was used to determine which extensions to contemporary BPM technology are needed to cover horizontal and vertical process integration functionalities. These are among others a direct connection with IoT devices (machines, robots, AGVs, etc.), a mechanism to deal with batching and unbatching of parts into bigger or smaller units, resource models, and role resolution mechanisms that include robot characteristics, exception handling for safety reasons and technical equipment failures, etc. [38].

6.4 Architecture of the HORSE System

In order to realize the HORSE system, a structured and systematic design approach was followed using two frameworks: (1) the well-known software engineering 4+1 framework of [24] to deal with the various views (logical, development, process, physical, scenario) of stakeholders and (2) a five-aspect framework for the

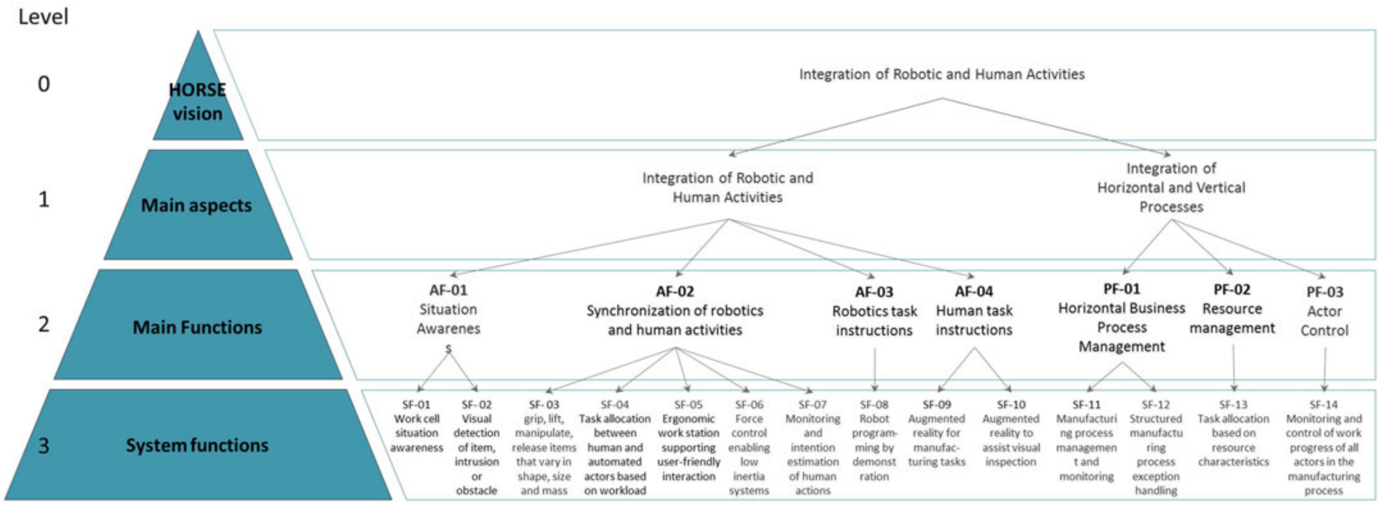
Level
0
HORSE vision
1
Main aspects
2
Main Functions
3
System functions
Integration of Robotic and Human Activities
Integration of Robotic and Human Activities
Integration of Horizontal and Vertical Processes
AF-01 Situation Awarenes s
AF-02 Synchronization of robotics and human activities
AF-03 Robotics task instructions
AF-04 Human task instructions
PF-01 Horizontal Business Process Management
PF-02 Resource management
PF-03 Actor Control
SF-01 Work cell situation awareness
SF-02 Visual detection of item, intrusion or obstacle
SF-03 grip, lift, manipulate, release items that vary in shape, size and mass
SF-04 Task allocation between human and automated actors based on workload
SF-05 Ergonomic work station supporting user-friendly interaction
SF-06 Force control enabling low inertia systems
SF-07 Monitoring and intention estimation of human actions
SF-08 Robot program-ming by demonst ration
SF-09 Augmented reality for manufac-turing tasks
SF-10 Augmented reality to assist visual inspection
SF-11 Manufacturi ng process managemen t and monitoring
SF-12 Structured manufactu ring process exception handling
SF-13 Task allocation based on resource characteristics
SF-14 Monitoring and control of work progress of all actors in the manufacturing process

Fig. 6.9 HORSE requirements framework. Please see online appendix for a larger version

Table 6.2 Mapping of requirements to system functions

Main functions	System functions	System requirements
AF-02 Synchronization of robotic and human activities	SF-07 Monitoring and intention estimation of human actions	Case1-Req07: The system shall actively monitor its proximity to ensure no harm is done to the human
PF-01 Horizontal business process management	SF-11 Manufacturing process management and monitoring	Case1-Req01: The HORSE system shall manage and coordinate all activities for an individual production order, based on predefined requirements of that order
		Case1-Req02: The process management system shall manage activity multiplicity based on predefined production order requirements
	SF-12 Structured manufacturing process exception handling	Case1-Req03: The process management system shall manage process exceptions by initiating a predefined response
PF-02 Resource management	SF-13 Task allocation based on resource characteristics	Case1-Req06: The Loading Robot shall be able to lift parts with a maximum mass of 5 kg
PF-03 Actor control	SF-14 Monitoring and control of progress of all actors in the manufacturing process	Case1-Req04: The process management system shall coordinate the activities of multiple actors involved in the same manufacturing process
		Case1-Req05: The HORSE system shall queue tasks to be performed by the same resource

design of business information systems [15, 37] to deal with the set of enterprise information aspects of a description of a complex information system: process, data, organization, software, and platform. The full design process is documented in [14, 17]. Here, we will only summarize the logical software architecture and the realization of that architecture in a concrete system.

Based on the system requirements elicited, first, a *logical architecture* of the HORSE system was developed [14, 24], before elaborating on specific technologies in the software and platform aspects of the physical views of the architecture. The system architecture has a layered style, with a division between global and local functions. The global layer includes the functions for process management that are

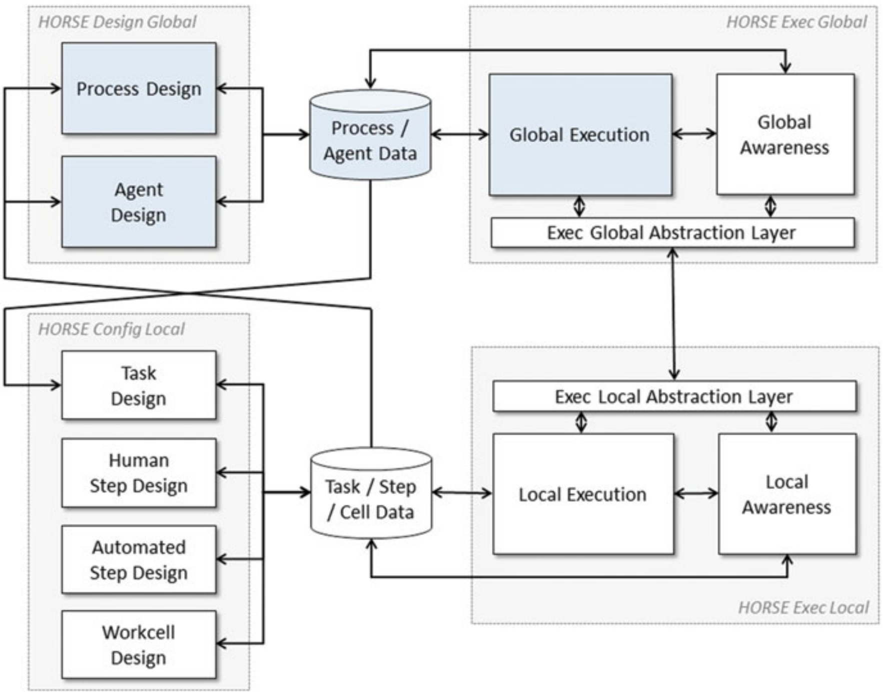

Fig. 6.10 The logical view of the HORSE system architecture; highlighted in light blue on the top left of this picture are the MPMS modules

applicable to multiple work units of the factory (work unit refers to the lowest level of the physical hierarchy shown in Fig. 6.3). The local layer provides functionality used within a single work unit. Figure 6.10 shows the *software aspect* [14, 37] of the logical architecture, at aggregation level 2. The vertical integration between global process management and local work cell execution is handled via the abstraction layers.

The architecture distinguishes between design-time and run-time functionality. Design-time functionality is used to define activities, agents, and physical constraints of the manufacturing system through (executable) process models, extended with manufacturing specific elements [1, 23]. Run-time functionality makes use of the design-time definitions to enact the processes and assign tasks to agents, within the bounds of the physical constraints. Thus, design-time functionality is used to develop the process models, and run-time functionality then enacts those process models and invokes the local layer of the HORSE system to execute activities on the factory floor.

The realization of this logical architecture was done using different technologies. The process management modules of this system (indicated in light blue in Fig. 6.10) are realized in the Camunda technology, an open-source platform for workflow and business process automation [2] using the BPMN 2.0 process modeling language [28]. Advanced functionalities (e.g., for advanced resource allocation) are implemented through extensions to the basic Camunda code. The abstraction layers are realized through OSGi middleware and local execution systems built on, e.g., ROS FlexBE, KUKA Sunrise robot platform, and OPC UA [17]. For the global part, one embodiment of the architecture is chosen; while for the local part, some flexibility in the embodiment is required to suit local infrastructures at factories. The HORSE system therefore is a lightweight, modular information system that enacts manufacturing processes and invokes the functions of a variety of technological developments, including situationally aware robotics, automated guided vehicles, and augmented reality.

6.5 Executable Process Models

The processes for all three case studies were implemented as executable models through the MPMS modules of the HORSE system. This section discusses the (development of the) executable models for the first case study and reflects upon the lessons learned and what was needed to develop these executable models based on the models used for business analysis and requirements elicitation. Due to space constraints, we only show resulting executable models for the first case study. The executable models for the other two case studies can be found in the online appendix (see footnote 1). The lessons learned, presented in Sect. 6.5.2 in the form of an extended method to develop executable processes, however, are based on our experiences with all three case studies.

6.5.1 Executable Processes for Case Study 1

As described in Sect. 6.3.1, the scope for the first case study was narrowed down to two subprocesses: the future PA1 tool assembly process and the future PA2 loading process. With the right additions and changes, which are discussed in more detail in the next section, the business process models of Figs. 6.6 and 6.8 were transformed into executable process models (see Figs. 6.11 and 6.12, respectively). An explanation of the main transformations is given in the discussion below. The executable processes can again be classified according to the process classification of Chap. 2 as indicated in Table 6.3.

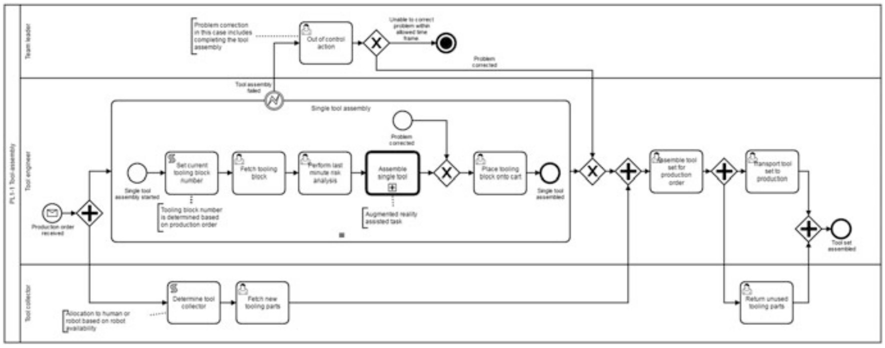
PL1-1 Tool-assembly
Team leader
Tool engineer
Tool collector
Problem correction in this case includes completing the tool assembly
Out of control action
Unable to correct problem within allowed time frame
Problem corrected
Tool assembly failed
Single tool assembly
Problem corrected
Production order received
Single tool assembly started
Set current tooling block number
Fetch tooling block
Perform last minute risk analysis
Assemble single tool
Place tooling block onto cart
Single tool assembled
Tooling block number is determined based on production order
Augmented reality assisted task
Assemble tool set for production order
Transport tool set to production
Tool set assembled
Determine tool collector
Fetch new tooling parts
Return unused tooling parts
Allocation to human or robot based on robot availability

Fig. 6.11 Executable process model of the PL1-1 tool assembly process in PA1 (cf. Fig. 6.6)

The Tool Assembly Process In the future tool assembly process (cf. Fig. 6.6 vs. Fig. 6.11), several *script tasks* were added to automatically select the right agent—human or robot—for the execution of the task *Fetch new tooling parts* and to automatically retrieve additional information on the exact tooling block to be assembled for this production order.

Furthermore, the *Assemble single tool* task which was modeled as a user task in the business model was implemented in the executable model with a reusable *call activity* that represents a standardized way of communication to a device on the local level of the HORSE architecture, such as the augmented reality system that guides the human operator through the tool assembly steps. The internals of this call activity are specified by Fig. 6.13. The messages with task instructions or task events are standardized and sent to/from the local-level device through the abstraction layers.

The Loading Process In the future loading process (cf. Fig. 6.8 vs. Fig. 6.12), an exception was defined handling a failure of the robotic task *Place parts on conveyor belt*. In case something goes wrong in this task and the Placing Robot cannot properly finish it, an *out-of-control action* should happen. This was not specified at the business-level process model as it was considered a technical issue.

Furthermore, internal variables were specified for the correct handling of the multi-instance tasks *Place parts on conveyor belt* and *Grab and hang single profile*. These variables are filled in a form that is presented to the user when initiating the process instance. Ideally, this information can be automatically derived from the production order information.

Moreover, as in the tool assembly process, tasks that involve direct communication with local-level devices such as the Loading Robot, Placing Robot, and Crane were again replaced by the standardized call activity (Fig. 6.13).

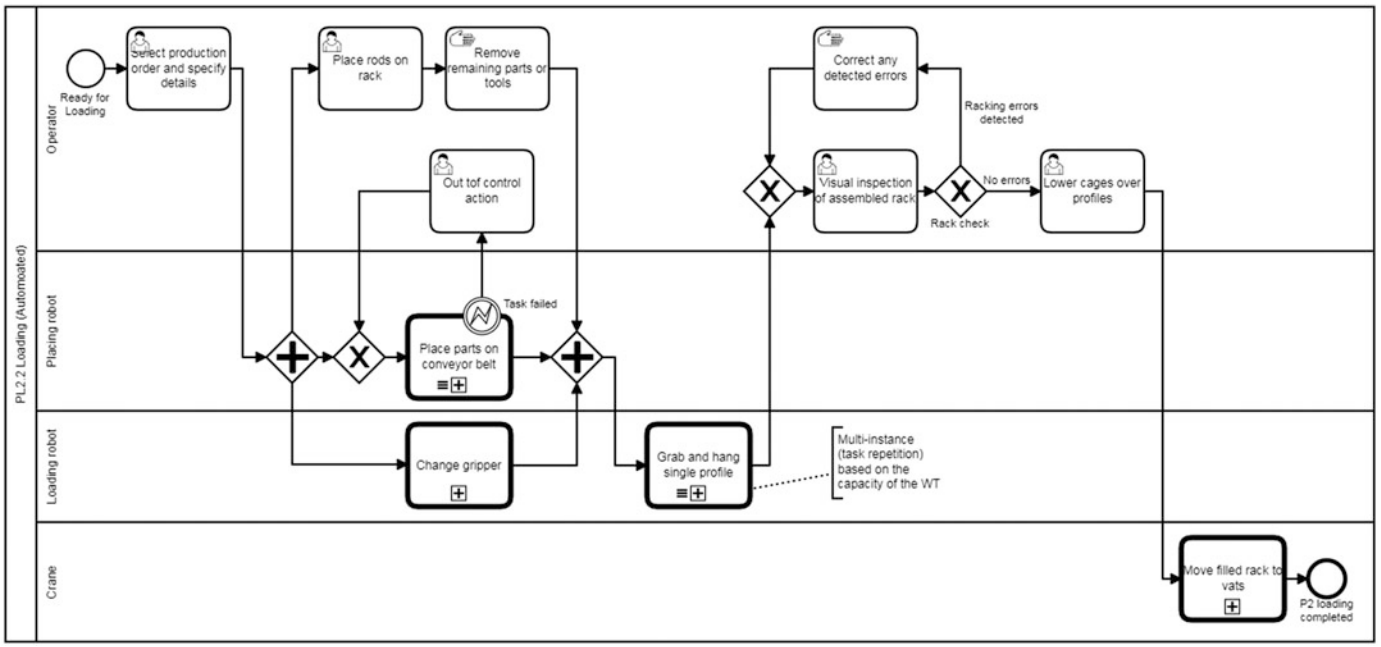
PL2.2 Loading (Automated)
Operator
Placing robot
Loading robot
Crane
Ready for Loading
Select production order and specify details
Place rods on rack
Remove remaining parts or tools
Out tof control action
Task failed
Place parts on conveyor belt
Change gripper
Grab and hang single profile
Multi-instance (task repetition) based on the capacity of the WT
Correct any detected errors
Visual inspection of assembled rack
Rack check
Racking errors detected
No errors
Lower cages over profiles
Move filled rack to vats
P2 loading completed

Fig. 6.12 Executable process model of the PL2-1 loading process in PA2 (cf. Fig. 6.8)

Table 6.3 Process classification of case study 1 executable process models according to Lübke et al.'s [4]

Process name	Case study 1 PL 1-1	Case study 1 PL 2-1
Version	1.0	1.0
Domain	High-tech manufacturing	High-tech manufacturing
Geography	Netherlands	Netherlands
Time	2018	2018
Boundaries	Intraorganizational	Intraorganizational
Relationship	Is being called	Is being called
Scope	Business scope, core; technical scope	Business scope, core; technical scope
Process model purpose	Execution	Execution
People involvement	Partly	Partly
Process language	BPMN 2.0	BPMN 2.0
Execution engine	Camunda	Camunda
Model maturity	Prototypical	Prototypical

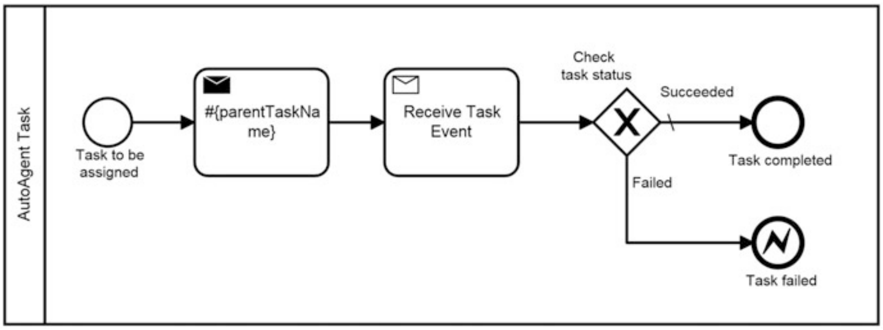

Fig. 6.13 The reusable subprocess to assign tasks and instructions to automated agents, e.g., the Crane, Loading Robot, and Placing Robot in the executable process model of the loading process in PA2 (cf. Fig. 6.12)

6.5.2 Method to Develop Executable Process Models

In order to develop the executable models for the pilot cases, again a systematic approach was followed. Dumas et al. [6] describe a five-step method to convert the business analysis process model—fit for communication and analysis purposes—into an executable model, suitable to feed into a business process execution engine: (1) Identify the automation boundaries. (2) Review manual tasks. (3) Complete the process model. (4) Bring the process model to an adequate granularity level. (5) Specify execution properties. We adopted the general guidelines of the above method and adapted and extended them where necessary, as we explain later in this section. In order to arrive at the detailed executable processes, we followed the following steps:

Step 1: Identify the Type of Tasks (Automated, User, Manual Tasks) This step is a combination of the first two steps of the method described in [6]. The aim is to assess which parts of the process can be coordinated by the MPMS and which cannot. The characterization of the type of each task (manual, user, automated, robotic) is an initial and important step that has to be done for all tasks.

In manufacturing processes, there are a lot of manual tasks, for instance, *Hang profiles* (in PA2 of the first case study), that do not add any value to a process management system. These are not executed by a process engine but can still be present in a process model. The manufacturing tasks that are interesting from an MPMS point of view are the ones performed by humans with the aid of a software application, robots, and/or automated vehicles. These are implemented as BPMN user tasks (for humans) or with the "send"-"receive" pattern (for robots) and the ones that should be configured in order to be ready for real execution. For example, in the loading process of Fig. 6.12, the task *Place rods on rack* is modeled and executed as a user task since the agent to perform it is a human operator with the assistance of a tasklist handler. On the other hand, the manual task *Remove remaining parts or tools* is deemed to be not relevant for execution and is not assisted by MPMS. In the tool assembly process of Fig. 6.11, we also see the task *Fetch new tooling parts* which is performed by a either a robot or a human operator.

Step 2: Bring the Process to Adequate Task Granularity Level The goal of this step is to make sure that tasks on the lowest subprocess level actually are logical units of work. They should consist of a coherent set of steps and have a clear goal that can be reached by one agent (or team of agents).

In the HORSE architecture [14], there is a distinction between tasks and steps. A task is defined as a *set of steps under responsibility of a single team of agents*. A task may not contain subtasks. Steps are more detailed actions within a task and are defined as *units of work performed by a single agent*. A step may contain substeps. A task then consists of a number of steps. Tasks, as a more high-level aspect, are handled by MPMS, while steps are handled by the local level of the HORSE architecture. An illustrative example is the robot task *Grab, lift, and hang single profile* of Fig. 6.12. This is a high-level action for which an MPMS cares for the assignment of the task to the robot and the confirmation of task completion (or failure). The internal robotic steps needed to perform the task (like "Close gripper," "Move the robot X cm," etc.) are details handled by modules of the local level, often specified in a robot execution script or taught by demonstration to the robot. Tasks should be demarcated correctly. If, on the one hand, they are too fine grained, the MPMS will continuously interrupt the agent performing the task to tell what the next task is that should be performed. For example, if "Grab profile," "Lift profile," and "Hang profile" are separate tasks, the agent must confirm each of the tasks separately and will receive separate work orders from the MPMS for each of them. If, on the other hand, tasks are too coarse grained, the agent may need to involve other agents, while it should be the MPMS that tells each agent what to do. For example, if "Hang profile" and "Paint profile" should be done by different (teams of) agents, they should be separate tasks. Otherwise, the agent who hung the profile should inform the agent who has to paint the profile. This guideline applies first of all to the business-level model, but needs to be reviewed when the executable model is created such that in case of misalignment with designed robotic tasks (in the task design module on the local configuration level), a task needs to be split or aggregated.

In our approach, we see this step as a logical sequence of the first step, in which the type of tasks (and the tasks themselves) has been identified. For this reason, we adapted the sequence of steps of the method described in [6].

Step 3: Redesign Pools and Swim Lanes for Interacting Processes In this step, the goal is to decide whether (and how) message flows between different pools in the model should be automated and to stratify communication between a main process and the (possibly multiple) related instances of its subprocesses.

Pools are used here to represent resource classes or whole processes, while swim lanes are used to partition a pool into subclasses or single resources. They are useful for a business-oriented model but have no execution semantics. When the pools are used to represent different processes, most likely they are connected to each other at some point. However, when executing such processes, it may be difficult to synchronize their instances. For example, a main production process may run in a sequential multi-instance scenario, and at some point, it calls another supplementary process that in turn has to call back the main process. The callback from the supplementary process may happen at a time in which the calling instance of the main process has been terminated (and the next instance of the multi-instance pattern has initiated). This may lead to wrong information updates or no synchronization at all. Therefore, it is preferred to model two (or more) processes in the same pool/lane in order to be executed properly and as intended to be. This was the case in case study 3, in which an alert is sent to the supervisor in case of a defected product—see the online appendix (see footnote 1). In the business process model, the supervision role is modeled as a separate pool; but in the executable process model, we had to incorporate it in the main pool, which was implemented as a multi-instance case.

This step is an addition to the approach presented in [6], since we consider it important.

Step 4: Complete the Process Model (e.g., with Exceptions) The models resulting from the elicitation phase often lack certain detailed information. The goal of this step is to enhance and complete the model with the necessary information. This step is the same as the third step of the method described in [6]. Based on our experience with the case studies, we complete Dumas et al.'s list to the following:

- Data objects. Input and output information for all tasks and decisions should be specified through (global) variables, attributes, and forms (on start events or user tasks).
- Exceptions. Alternative paths and (technical) exceptions should be specified such that the process can handle all possible situations. This requires specifying all (mutually exclusive) options after a decision gateway, alternative routes,

exceptions, timeouts, failures, corrective actions, etc. For instance, in Fig. 6.12 to make the PA2 loading process executable, all possible robot task failures must be covered. Otherwise, the process engine may get stuck when one of these failures occurs. These failures can be specified as BPMN exceptions. For example, in task *Place parts on conveyor belt*, a task failure can raise an exception leading to a standardized *out-of-control action* task.

- Resource assignment. The assignment of resources to tasks (via, e.g., direct assignees, groups (candidate users), or more advanced selections through expressions).
- Multi-instance activities. Make sure that multi-instance activities/subprocesses are repeated for the correct number of times and that the right type of multi-instance is specified (parallel or sequential). Multi-instance activities are used to deal with different abstraction levels of the case in the process. For instance, one order can lead to the production of 50 sliders that each need 3 profiles. Then on the main process level, the instance is the order, which invokes 50 instances of the slider production process and others.
- Reusable subprocesses. If similar functionality is needed, try to, if possible, define parameterizable reusable subprocesses that may be invoked from different places in the main process, e.g., the call activity of Fig. 6.13 realizing the technical communication to the robotic actors in the process.

Step 5: Integrate with Other Systems (Messaging Middleware, DB Server) An executable process rarely relies on its own execution engine system alone to run. It normally interacts with other information systems or database servers. In the HORSE system, the MPMS module needs to communicate with other modules and databases as specified in the HORSE architecture in Fig. 6.10. The goal of this step is to realize the technical integration with these systems.

Therefore, in an executable process model, there should be extra service tasks to invoke other systems (or any delegate code on user tasks or DB connectors, as we will see later on the "BPMS-specific properties" step). Such integration services could be, for instance, any web service or REST calls. In all three case studies, we implemented connections to a database, when information was needed to be retrieved or stored, and also connection to the local level of the HORSE architecture through the abstraction layers, when tasks had to be assigned to robots.

This step is also an addition to the approach described in [6], since integration of MPMS to the other modules of the HORSE architecture is vital for the whole system.

Step 6: Specify Execution Properties (i.e., Implementation Details That Are Not Depicted in the BPMN Model but Needed by the Execution Engine) In the business-oriented process model, many details are omitted for the sake of simplicity. Such details are as follows:

- Process variables, messages, signals, and errors. Process variables are used to store data information used throughout a process instance. It can be information on, e.g., a production order number, a part ID, etc. Messages, signals, and errors also have to be specified on a process model so that the engine can execute them.

- Task and event variables and their mappings to process variables. Tasks and events may have their internal variables that carry information. These have to be specified and mapped accordingly.
- Service details for service, send, and receive tasks and for message and signal events. Tasks, either service, send/receive, or user tasks, implement a business logic through a technical specification, such as a delegate code, a web service call, an execution listener, etc. In many cases, this requires the most effort from a developer to make a business-oriented model executable.
- Code snippets for script tasks. Similar to the service details of a task, some scripting may be required in some points of a workflow, for example, to update a process variable after a decision point.
- Participant assignment rules and user interface structure for user tasks. User tasks are the ones performed by a participant with the help of an application's user interface (or in the case of robot, tasks with the help of a script, implemented also as user tasks as we said previously). That means that the participant has first to be determined (either during design time or dynamically during runtime). Such a participant assignment rule is used on *Fetch new tooling parts* of Fig. 6.11, where the *script task* before will first invoke a dynamic agent allocation algorithm which will determine whether a robot or a human operator can perform the task. Then, the mechanism to notify the participant for the task assignment has to be implemented (BPMS normally has their own tasklist application; but extra notifications like emails, SMS, and push messages may be required). Finally, the user interface to present the right input information and capture the output result should be designed and implemented.
- Task, event, and sequence flow expressions. Various expressions may be needed on tasks (e.g., loop conditions), events, and sequence flows (any conditions, for instance).
- BPMS-specific properties. In addition to all of the above, extra settings and configurations may be needed on some BPMN elements to make them executable. Normally, a BPMS provides patterns for such configurations, for example, connectors.

6.6 Evaluation

At the time of writing this book chapter, the first version of the HORSE system is under installation and deployment at the pilot cases. Since these deployment experiments are not finished yet, only a first and preliminary evaluation of the PA1 (tool assembly process) solution from case study 1 can be presented here. The solution is evaluated in two ways:

1. As a proof-of-concept or feasibility test. The executable process models are defined in the MPMS modules, deployed on the factory floor, and enacted through the HORSE system enabling interaction with the augmented reality system and real process participants.
2. As an acceptance test. The process participants are interrogated regarding their perceived usefulness and ease of use.

Nineteen operators were asked to work with the system and complete the tool assembly process for one case (i.e., one tool block was assembled). Afterward, they were surveyed and interviewed to gauge their experience with the new process. The technology acceptance model (TAM) [5] is used as both survey and outline for the semi-structured interviews. The model includes twelve questions divided into two sections for perceived usefulness (PU) and perceived ease of use (PEOU). Importantly, the questions aim to determine whether the user prefers to use the new technology, compared to the previous way of working (which for the tool assembly process was completely manual as specified in Fig. 6.5). All twelve questions are measured on a Likert scale ranging from 1 (extremely likely) to 7 (extremely unlikely).

The 19 surveys and interviews generated significant data to be used for evaluation. Table 6.4 shows the 12 TAM statements posed to the interviewees and the average ratings as reported by the interviewees. As an overview, the system was rated favorably, with only two statements garnering a rating slightly unfavorably (i.e., numbers 8 and 10).

The most common complaints by participants were that the system forced them to work a certain way. Manufacturing processes that involve human participants tend to offer some flexibility to the participants on the precise order of tasks, or the resolution of mistakes and errors, especially if they are fully manual. With

Table 6.4 Average score per TAM statement

nr	Statement	Average rating (1–7)
1	Using the system in my job would enable me to accomplish tasks more quickly	2.32
2	Using the system would improve my job performance	2.26
3	Using the system in my job would increase my productivity	2.37
4	Using the system would enhance my effectiveness on the job	2.21
5	Using the system would make it easier to do my job	1.79
6	I would find the system useful in my job	2.16
7	Learning to operate the system would be easy for me	1.42
8	I would find it easy to get the system to do what I want it to do	3.53
9	My interaction with the system would be clear and understandable	1.84
10	I would find the system to be flexible to interact with	3.58
11	It would be easy for me to become skilful at using the system	1.68
12	I would find the system easy to use	1.79

an executable process enacted by an MPMS, this is no longer possible. This complaint is reflected in the average scores of the flexibility statements 8 and 10 in Table 6.1. The operators felt restricted and constrained by the system, minimizing their opportunity to pursue process improvement. However, a strict process enforced by the system would prevent many of the mistakes currently made by the human operators.

On the positive side, the participants were highly enthusiastic of the usefulness of the system. This optimism isn't necessarily related to the increased automation in the processes, leading to less burden on the operator, but rather related to the procedural nature of an MPMS-coordinated process. They acknowledged the value of having a system that encourages disciplined process execution. This is even more important for inexperienced operators, who can be trained faster to participate in complex processes. Apart from increased discipline, some participants also appreciated the lessened mental burden. The HORSE system presents the relevant information to perform a task, thus making it easier for an operator to follow instructions and perform the work.

6.7 Conclusions and Outlook

In this chapter, our experiences with developing executable processes for the high-tech manufacturing domain are discussed. We systematically derived requirements for a manufacturing process management system (MPMS) through a thorough analysis of three case studies. After the design and realization of the MPMS, we illustrated the development of the executable processes through the first case study, followed by a discussion of our lessons learned in the form of an extended stepwise method to transform a business process model into an executable model. We have already conducted some trials for the first pilot case—showing positive results. All in all, we conclude that our proof of concept shows that it is possible to support and coordinate high-tech manufacturing processes at real time with business process management technology, but that there are quite some technical and conceptual challenges to tackle and that standardization would be important for an industry strength solution.

6.7.1 Conclusions for Researchers

The research aspect behind this case study experience report is the exaptation of contemporary BPM technology (originating from the service industry) to the high-tech manufacturing domain. With these case studies and the general technology developments following that, we showed that manufacturing processes can be supported by BPM technology through, e.g., the HORSE MPMS system. This adoption of technology in a completely new and challenging domain, however, does not go without extensions. The main challenges we met here are the physical constraints in a manufacturing process (e.g., the intermediate storage and the multi-instance solutions for the different levels of granularity in the process) and the interfacing with robotic technology (for which no standard functionality or communication protocols were available yet). All in all, we conclude that the MPMS can bring many advantages to a manufacturing company but that there are still many research developments and innovations needed to make the MPMS an industry strength solution.

6.7.2 Takeaways for Practitioners

On the practical side, the three case studies have shown the possible advantages that explicit process management can bring to the manufacturing domain in order to better deal with the increasing dynamism [39]. Advantages are mainly found in a better overview of the production status and the flexible and automatic control of the process execution. The business process models turned out to be very useful for company stakeholders to get an integrated end-to-end view on their production processes, which they were often lacking. From a more technical perspective, we learned that transforming business-oriented process models into executable processes is not easy and time consuming. It requires additional specifications on various levels. We adopted a method available, but found it was not complete and, based on our experience with the three case studies, added and extended it. Our adjusted method is the most concrete contribution for practitioners as they can take it along to give structure to their efforts when creating executable manufacturing processes. Obviously, the current version of the method may be further evaluated and detailed to be of even more practical value.

6.7.3 Outlook

In this chapter, we presented a first version of the HORSE framework, mainly focused on the MPMS functionality. In the near future, this system will be further refined (e.g., including more specific manufacturing characteristics, more advanced exception handling, more advanced dynamic role resolution). Furthermore, in the next year, the HORSE framework will be extensively tested in real situations in the three pilot cases that already served as case studies here and in another seven new case studies. This will allow us to extensively evaluate the usefulness and value of the system and of our method and to further refine these. On the longer term, we envision the MPMS system could also be the basis for realizing a strong coupling of the management of manufacturing processes to the end-to-end corporate processes (i.e., from sales process to after-sales service) as discussed in [13] and even for flexible manufacturing network processes, such as described in [16, 34].

Acknowledgements The HORSE project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 680734.

References

1. M. Aspridou, Extending BPMN for modeling manufacturing processes, Master thesis, Business Information Systems, TU/e, 2017. https://tinyurl.com/ya6c8czh
2. Camunda: http://www.camunda.org
3. D. Chen, Enterprise-control system integration - an international standard. Int. J. Prod. Res. **43**(20), 4335–4357 (2005)
4. L. Daniel, I. Ana, P. Cesare, A template for categorizing empirical business process metrics. BPM Forum, Barcelona (Springer, 2017), pp. 36–52
5. F.D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. **13**(3), 319–340 (1989)
6. M. Dumas, M. La Rosa, J. Mendling, H. Reijers, *Fundamentals of Business Process Management* (Springer, Berlin, 2013)
7. S. Easterbrook et al., Selecting empirical methods for software engineering research, in *Guide to Advanced Empirical Software Engineering* (Springer, London, 2008), pp. 285–311
8. A. Estruch, J.A. Heredia Álvaro, Event-driven manufacturing process management approach, in *Business Process Management: 10th International Conference Proceedings*, ed. by A. Barros, A. Gal, E. Kindler, vol. 7481 (Springer, Berlin/Heidelberg, 2012), pp. 120–133
9. E. Filos, Four years of factories of the future in Europe: achievements and outlook. Int. J. Comput. Integr. Manuf. **30**, 18 (2015). https://www.tandfonline.com/doi/abs/10.1080/0951192X.2015.1044759
10. T. Gerber, A. Theorin, C. Johnsson, Towards a seamless integration between process modeling descriptions at business and production levels: work in progress. J. Intell. Manuf. **25**(5), 1089–1099 (2014)
11. D. Gerwin, An agenda for research on the flexibility of manufacturing processes. Int. J. Oper. Prod. Manag. **7**(1), 38–49 (1987)
12. P. Giesberts, L. Van den Tang, Dynamics of the customer order decoupling point: impact on information systems for production control. Prod. Plan. Control **3**(3), 300–313 (1992)
13. P. Grefen et al., Dynamic business network process management in instant virtual enterprises. Comput. Ind. **60**(2), 86–103 (Elsevier, 2009)
14. P. Grefen, I. Vanderfeesten, G. Boultadakis, Architecture design of the HORSE hybrid manufacturing process control system. Beta WP Series 518, TU/e (2016)
15. P. Grefen, R. Eshuis, N. Mehandjiev, G. Kouvas, G. Weichhart, *Business Information System Architecture*, Fall 2016 Edition. (Eindhoven University of Technology, Eindhoven, 2016)
16. P. Grefen, S. Rinderle-Ma, S. Dustdar, W. Fdhila, J. Mendling, S. Schulte, Charting process-based collaboration support in agile business networks. IEEE Internet Comput. **22**, 48–57 (IEEE Computer Society, 2018)
17. P. Grefen, I. Vanderfeesten, G. Boultadakis, Developing a cyber-physical system for hybrid manufacturing in an Internet-of-things context, in *Protocols and Applications for the Industrial Internet of Things* (IGI Global, 2018)
18. P. Grefen, R. Eshuis, O. Turetken, I. Vanderfeesten, A Reference Framework for Advanced Flexible Information Systems, in *Advanced Information Systems Engineering Workshops Proceedings*. LNBIP, vol. 316 (Springer, Cham, 2018), pp. 253–264
19. S. Gregor, A.R. Hevner, Positioning and presenting design science research for maximum impact. MIS Q. **37**(2), 337–356 (2013)
20. E. Hofmann, M. Rüsch, Industry 4.0 and the current status as well as future prospects on logistics. Comput. Ind. **89**, 23–34 (2017)

21. IEC, *Enterprise-Control System Integration - Part 1: Models and Terminology*, 2nd edn., vol. 1, 5 vols. (Int. Electrotechnical Commission, Geneva, 2013)
22. Industrie 4.0: Smart Manufacturing for the Future. Germany Trade and Invest (2014)
23. X. Jie-A-Looi, A method to enable ability-based resource allocation for runtime process management in manufacturing, Master thesis, TU/e, 2017. https://tinyurl.com/y75e7q8g
24. P. Kruchten, Architectural blueprints - the "4+1" view model of software architecture. IEEE Softw. **12**(6), 42–50 (1995)
25. S. Li, L. Da Xu, S. Zhao, The Internet of Things: a survey. Inf. Syst. Front. **17**(2), 243–259 (Springer, 2015)
26. D. Lucke, C. Constantinescu, E. Westkämper, Smart Factory - A Step towards the Next Generation of Manufacturing, in *Manufacturing Systems and Technologies for the New Frontier: The 41st CIRP Conference on Manufacturing Systems* May 26–28, 2008, Tokyo, ed. by M. Mitsuishi, K. Ueda, F. Kimura (Springer, London, 2008), pp. 115–118
27. A. Mavin et al., Easy approach to requirements syntax (EARS), in *2009 17th IEEE International Requirements Engineering Conference*, pp. 317–322 (2009)
28. Object Management Group. Business Process Model and Notation 2.0 specification. http://www.bpmn.org/
29. L. Prades, F. Romero, A. Estruch, A. García-Dominguez, J. Serrano, Defining a methodology to design and implement business process models in BPMN according to the standard ANSI/ISA-95 in a manufacturing enterprise. Proc. Eng. Suppl. C **63**, 115–122 (2013)
30. J.B.B. Rogers, F.W. Dewhurst, K.D. Barber, R.L.D.H. Burns, Business process modelling and simulation for manufacturing management: a practical way forward. Bus. Process. Manag. J. **9**(4), 527–542 (2003)
31. J. Romme, S. Hoekstra (eds.), *Integral Logistic Structures. Developing Customer-Oriented Goods Flow* (Industrial Press, New York, 1992)
32. M. Rudberg, W. Joakim, Mass customization in terms of the customer order decoupling point. Prod. Plan. Control **15**(4), 445–458 (2004)
33. P. Runeson, M. Host, A. Rainer, B. Regnell, *Case Study Research in Software Engineering* (Wiley, Hoboken, 2012)
34. S. Schulte, D. Schuller, R. Steinmetz, S. Abels, Plug-and-play virtual factories. Internet Comput. **16**(5), 78–82 (IEEE, 2012)
35. B. Silver, *BPMN Method and Style, with BPMN Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0* (Cody-Cassidy Press, Aptos, 2011)
36. The HORSE project: http://www.horse-project.eu
37. J. Truijens, A. Oosterhaven et al., *Informatie-Infrastructuur: een Instrument voor het Management* (Kluwer Bedrijfs-wetenschappen, 1990) (in Dutch)
38. I. Vanderfeesten, P. Grefen, Business process management technology for discrete manufacturing. Beta WP Series 486, TU/e (2015). http://onderzoeksschool-beta.nl/wp-content/uploads/wp_486-.pdf
39. Worldwide Manufacturing Predictions for 2015 (IDC Manuf. Press Rel., 2014)
40. S. Zor, F. Leymann, D. Schumm, A proposal of BPMN extensions for the manufacturing domain, in *Proceedings of 44th CIRP International Conference on Manufacturing Systems* (2011)