The HORSE framework: A reference architecture for cyber-physical systems in hybrid smart manufacturing

Citation for published version (APA):

Document license:
CC BY

DOI:
10.1016/j.jmsy.2021.09.003

Document status and date:
Published: 13/10/2021

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 27. May, 2022
The HORSE framework: A reference architecture for cyber-physical systems in hybrid smart manufacturing

Konstantinos Traganos\textsuperscript{a,}\textsuperscript{*}, Paul Grefen\textsuperscript{a,}\textsuperscript{b}, Irene Vanderfeesten\textsuperscript{a,}\textsuperscript{c}, Jonnro Erasmus\textsuperscript{a}, Georgios Boultadakis\textsuperscript{d}, Panagiotis Bouklis\textsuperscript{d}

\textsuperscript{a} School of Industrial Engineering, Eindhoven University of Technology, De Zaalle, 5600MB, Eindhoven, The Netherlands
\textsuperscript{b} Atos Digital Transformation Consulting, Flight Forum 3000, 5657 BW, Eindhoven, The Netherlands
\textsuperscript{c} Department of Information Science, Open University of the Netherlands, Valkenburgerweg 177, 6419AT, Heerlen, The Netherlands
\textsuperscript{d} European Dynamics S.A., Kifissias Av. & Arkadiou Str. 209, 15124, Maroussi, Athens, Greece

ARTICLE INFO

Keywords:
Smart manufacturing
Industry 4.0
Cyber-physical systems
Reference architecture
HORSE EU project

ABSTRACT

In the Industry 4.0 era, manufacturers strive to remain competitive by using advanced technologies such as collaborative robots, automated guided vehicles, augmented reality support and smart devices. However, only if these technological advancements are integrated into their system context in a seamless way, they can deliver their full potential to a manufacturing organization. This integration requires a system architecture as a blueprint for positioning and interconnection of the technologies. For this purpose, the HORSE framework, resulting from the HORSE EU H2020 project, has been developed to act as a reference architecture of a cyber-physical system to integrate various Industry 4.0 technologies and support hybrid manufacturing processes, i.e., processes in which human and robotic workers collaborate. The architecture has been created using design science research, based on well-known software engineering frameworks, established manufacturing domain standards and practical industry requirements. The value of a reference architecture is mainly established by application in practice. For this purpose, this paper presents the application and evaluation of the HORSE framework in 10 manufacturing plants across Europe, each with its own characteristics. Through the physical deployment and demonstration, the framework proved its goal to be basis for the well-structured design of an operational smart manufacturing cyber-physical system that provides horizontal, cross-functional management of manufacturing processes and vertical control of heterogeneous technologies in work cells. We report on valuable insights on the difficulties to realize such systems in specific situations. The experiences form the basis for improved adoption, further improvement and extension of the framework. In sum, this paper shows how a reference architecture framework supports the structured application of Industry 4.0 technologies in manufacturing environments that so far have relied on more traditional digital technology.

1. Introduction

Currently, markets for many product categories are becoming increasingly dynamic. The electronics and automotive markets are typical examples. This development implies that manufacturers have to become increasingly flexible in their operations. Customers demand more tailor-made products, with shorter delivery times. Manufacturing processes have become more complex to satisfy this demand and enterprises have to be reactive to stay competitive. The Industry 4.0 developments with advanced robotics, Internet-of-Things (IoT), Cyber-Physical Systems (CPS) (the reader can refer to Appendix A for a list of abbreviations), and Cloud Computing promise significant gains in production efficiency, manufacturing flexibility and product customization [1]. The realization in industrial practice, though, of these developments is not an easy task, as it faces many challenges [2–4].

1.1. Research context and problem identification

The evolution of the manufacturing paradigm has always been characterized by both market trends and technology advancements. In
the ongoing 4th industrial revolution, customers demand more individual and personal products, with high quality and fast delivery. The versatility of products and the configuration options upon ordering give rise to mass customization and personalization [5,6]. This shift, though, puts pressure on manufacturers, which must adapt and reconfigure their systems in order to offer the product variety within short lead times. Production batch sizes get smaller with frequent tool change-over, production variety introduces complexity, employees should be skilled and flexible to perform new operations [7]. Such effects have been mitigated by flexible or reconfigurable manufacturing systems [8], [9]. However, typically these systems are not well-suited to support concepts that Industry 4.0 brings, such as interoperability and consciousness [10].

The recent technology developments offer more flexibility in production operations. Versatile robots, with the appropriate end effectors attached, can switch modes and perform various operations, and even address ad hoc, unexpected changes [11]. By programming by demonstration [12], manipulators can more easily add new functionalities to robots, making their utilization more efficient. Collaborative robots (cobots) increase efficiency by allowing robots and human operators to work together [13,14]. Augmented Reality (AR) systems support operators in their daily tasks, which are getting more complex [15,16]. Automated guided vehicles (AGV) transport material and products around a factory, without human intervention [17]. Smart sensors gather any kind of values from devices that help in predictive maintenance or decision making. And all these developments are leveraged by the connectivity that the IoT [18] and cloud computing [19,20] provide.

Even small and medium enterprises (SMEs) in the manufacturing industry can now afford buying any of these smart equipment and technologies, but their utilization is not optimal yet. Primarily, this is because robotic solutions are often employed in disparate work cells, following a vertical orientation in their robot control processes. This normally leads to isolated, fragmented developments that do not solve the need for production adaptability and flexibility at the level of the overall, end-to-end manufacturing process. As Kagermann et al. [21] recommend, apart from vertical integration of manufacturing systems within the factory, horizontal integration of value networks and end-to-end digital integration of engineering across the entire value chain are key concepts of implementing the Industry 4.0 initiative. But as the number of systems and technologies increase, typically based on different control regimes and offered by multiple vendors [22], their integration is a challenge.

The introduction of robotics may also be suboptimal in complex production scenarios in which dynamic actor allocation is needed for extra flexibility. Think, for example, of a scenario in which an AGV raises an alert of low battery capacity and a human operator has to take over a materials transportation task. It is typically hard to transfer tasks from robotics to humans and vice versa, as each actor class is controlled differently and independently [23]. Robots and machines are forced to action through their control systems, while humans receive instructions orally, written, or visually through screens. Moreover, the co-presence and collaboration of humans and robots requires adequate attention with respect to safety requirements to prevent hazards for humans.

It is obvious thus, that while smart technologies in manufacturing are present and promising, their potentials are not fully exploited as their cross-functional and vertical integrations in hybrid-actor settings remain challenging.

1.2. Goal of the paper

The production complexity that the current market trends introduce in discrete manufacturing requires a system to couple the digital/cyber aspect of information systems for flexible manufacturing process management with the physical aspect of robotics and smart devices. The system should enable a seamless integration of vertical, within work cells developments, with a horizontal, across work cells, process orientation, supporting the co-presence and safe collaboration of human workers and robotics. The horizontal process integration is addressed by orchestration of activities performed by the right actors at the right time with the right information. The vertical integration is tackled with interoperable, modular solutions that control the heterogeneous smart technologies.

Such a cyber-physical system has been designed, developed and demonstrated in the HORSE project1, which is a Research and Innovation Project in the European Union (EU) Horizon 2020 program. HORSE ran from 2015 to 2020, as a collaboration of 15 organizations across Europe, among which academic and applied research institutions,

1 http://www.horse-project.eu/.
technology providers and manufacturing enterprises. The goal of the project is to make a significant advancement in the industrial use of smart manufacturing by developing an integrated framework that extends and unifies several state-of-the-art technologies, including smart robotics, business process management technology, as well as the development of software technology components to implement this framework in practice. The project focuses on SMEs, as these enterprises face the greatest challenges in the adoption of smart manufacturing technology. Also, it focuses on the discrete manufacturing domain, i.e., on companies that manufacture individual or batches of individual products, as this domain faces the most pressure from the mass customization and personalization trends.

The HORSE project follows the design science research (DSR) paradigm, with the purpose to generate prescriptive knowledge that can be applied in practice [24]. That means that rigorous scientific knowledge, taking into consideration relevant industry needs, is applied to develop an artefact [25] to address practical problems. The artefact created within the HORSE project is a reference architecture for designing and building a CPS for smart manufacturing control. The project is structured with the design science research framework of Hevner et al. [26], shown in Fig. 1. By analyzing challenges and needs encountered in real environments in manufacturing industry, together with the push by Industry 4.0 technology drivers, we ensure the practical applicability of the research. By following design principles, derived from both domain independent scientific knowledge and domain dependent industry standards, we ensure rigor in the development of the reference architecture.

Thus, the conceptual design is that of a modular reference architecture [27] of a CPS that serves as a blueprint for building solutions for enterprises in the discrete manufacturing domain, towards their goal to safely integrate robotics in their end-to-end operations to address their challenges. The blueprint, by design, enables interoperability of heterogeneous systems and technologies, and provides flexibility, through its modularity, to be adapted to specific needs. The implementation and the physical deployment of the architecture with concrete systems proves feasibility in a broad range of manufacturing scenarios.

In this paper, we present how the design of a CPS following the reference architecture and its technical realization were applied in real-world industry scenarios to solve practical challenges in production. These challenges are results of the increasing demand of mass customization and personalization and the need for automated and flexible manufacturing operations – as introduced before. As these challenges demand for the application of new, emerging technologies, this paper shows how the reference architecture supports integration of these technologies in a structured way that can support various manufacturing settings.

The evaluation of the implemented systems indicates the value of the reference framework and provide valuable insights of how manufacturing enterprises can embrace smart manufacturing.

The structure of this paper is as follows. First, we start in Section 2 with the practical challenges of manufacturing scenarios, as analyzed in the pilots cases of the HORSE project. The analysis leads to a list of requirements that the CPS to be developed should meet. Section 3 presents the high-level design of the HORSE logical system architecture for the CPS and its technology embodiment. Section 4 demonstrates the application of the HORSE framework with respect to the challenges analyzed in Section 2. We reflect on the value of the framework in Section 5, outlining also important extensions for its better exploitation. In Section 6 we discuss related work on initiatives in smart manufacturing. Finally, we conclude the paper in Section 7.

2. Requirements analysis

The industrial application organizations of the HORSE consortium served, initially, as the source for inspiration for the framework’s requirements and, then, as a test-bed to ensure the practical relevance of the developed solution. In total, ten factories, initial three pilots and seven open call cases, across all Europe, manufacturing products in various domains, consist a representative set of industry needs (see Appendix B for a brief overview of the pilots and the open call experiments). In this section, we briefly present the practical scenarios of three factories (as interesting and diverse cases) and the elicited system requirements of the HORSE framework.

2.1. Pilots description

Through on-site visits, observations and interviews, a systematic approach was followed to analyze the manufacturing structure and processes of the involved case studies, in order to identify problems, bottlenecks and challenges. The approach is discussed in [28] and the complete pilots analysis is presented in [29]. Here, in the interest of brevity, we briefly present only three case studies.

2.1.1. Bosch (BOS)

Robert Bosch España, Fábrica de Castellet produces wiper system assemblies (WSAs) (like the ones in Fig. 2(a)) for the automotive industry. The variation on car design and windshield sizes require appropriately sized wiper systems to ensure efficient and reliable windshield cleaning. A single production line for front WSAs manufacturing consists of several working stations (about seven of which are fully automatic, one semi-automatic and finally five manual working stations), operating in sequence for assembling the different
components of the WSAs. Different WSA part numbers can be produced at the same production line. But as the final products vary in shape, mass, size and parts, production lines are frequently stopped to adjust tools and equipment to be prepared for a different, small, batch of products.

The final workstations deal with the final product verification and is in the scope of the HORSE project. A worker picks up each WSA from a conveyor belt with both hands. Depending on the part number, specific points have to be inspected (for missing labels, damaged ball sockets, clamped dust cups, etc.). That requires rotating the part a few times. In case the WSA does not require any small fix or is not for scrap, the operator places it in a customer’s packaging box (in adverse back position) according to packaging instructions and with the most optimal way to save space (ready to be shipped). Once the box floor is full, a new carton layer is placed on top to get more WSAs. This process, shown in Fig. 2(b), is manual because the workers (especially the experienced ones) can quickly check the specified points and position the parts into the packaging boxes.

However, this type of work that the workers perform in this process puts them in occupational hazards and health issues. As an average WSA weighs around 3 kg, there is a lot of strain on a worker’s back, wrist and there is risk of hitting their fingers. Also, the product variation and the various configurations in packaging require a lot of flexibility which is now covered by only experienced operators that are occupied in each working station.

BOS aims to automate this process and alleviate workers’ work by:

- Automating the quality inspection of WSAs
- Eliminating non-ergonomically movements by operators
- Automating and making more flexible the packaging into customer’s boxes

The automated process should also respect the current cycle times.

---

Fig. 3. TRI high end ball bearing slides & vertical balance systems.

Fig. 4. TRI entire current manufacturing process.

Fig. 5. TRI pilot – (a) Unorganized bin stacking with profiles (b) Manual profiles hanging.
2.1.2. Thomas Regout International (TRI)

Thomas Regout International is a Dutch manufacturer specialized in the production and design of customized telescopic slides (Fig. 3) for several industrial equipment applications. The market in which TRI operates is characterized by high quality demands (zero defects), just-in-time delivery, and customized small series of products, with short time-to-market times.

The entire production process, illustrated in Fig. 4, consists of three production areas (PAs) — cold forming of steel, stamping and welding (P1), surface treatment of the steel profiles (P2), and final assembly (P3). As the stamping step at P1 requires specific tools, these are prepared before P1 starts. The highly product customization requires a wide range of tooling configuration, and currently this is heavily dependent on operators’ expertise.

Storage areas for intermediate products are needed before and after P2. The connection of P2 to P1 and P3 is not automated yet. When profiles are welded at P1, they are placed in bins in a rather unorganized way (Fig. 5(a)). Operators attach paper cards on the bins with product specifications, but due to high product variability, often these cards contain errors. There are also cases in which these cards are lost during transportation of the bins, causing a lot of delays in production. The disconnected PAs hinders the visibility and efficiency of operations on the shopfloor.

Another challenge that TRI faces is the heavy manual labor at P2. Workers lift the profiles from bins and hang them on racks to undergo a galvanization process (Fig. 5(b)). This repetitive process causes a lot of physical strains.

TRI aims to transform into a smart factory in the following areas (within the HORSE project):

- Overall process orchestration and monitoring across all PAs
- Automated profile hanging in P2
- Reducing the dependency on operators’ expertise

2.1.3. Guided Safety

The Guided Safety experiment handled the safe design, configuration and execution of the small parts assembly process of DENSO, a leading supplier of automotive electronics and mechatronics parts (Fig. 6(a)) for automakers. While many production stages are fully automated, assembly is still largely carried out manually. This is mostly because the small batch sizes and the diversity in product variants.

DENSO wants to introduce robots in the assembly process. A new assembly production line, within the HORSE project scope, consists of four stations, connected by a circular conveyor, where the return line is allocated vertically. Two elevators are used for moving pallets between the two lines. A pallet buffer is allocated before each station and elevator. In total, there can be up to 10 pallets circulating in the line. The first two stations are setup for manual assembly (Fig. 6(b)), while the last two for automated assembly.

However, simply putting two robots in assembly does not offer the desired efficiency. DENSO aims to increase the flexibility and re-
configurability of the assembly line by being able to quickly switch setups and configuration based on the assembly scenario. For instance, they want to switch from a scenario where one human station and two robot stations are occupied to a scenario where two human stations and one robot station are needed, within a strict cycle time.

2.2. System requirements

Case studies like the ones described above indicate real challenges in discrete manufacturing in the era of mass-customization and personalization. The analysis of all pilots in the HORSE project resulted in a set of requirements with a common goal to automate processes, increase flexibility and guard safe human-robot collaboration [30]. The requirements were categorized to ensure homogenization and avoid the risk of developing case-specific systems. Fig. 7 shows the conceptual requirements framework, divided into two main categories: functions for the integration of robotic and human activities in individual work cells (Activity Functions – AF, left part of taxonomy), and functions for horizontal and vertical process integration (Process Functions – PF, right part of taxonomy).

The main functions have been further decomposed into more detailed system functions which are then translated into concrete functional and non-functional system requirements [29]. These guided the design of the conceptual system presented in next section.

3. HORSE system

This section describes the HORSE system in its complete form; from a conceptual reference architecture to the technology realization. Before, though, we present the system, we discuss in Section 3.1 how its design is conceptually aligned with industry standards. This alignment guards the rigor of our DSR approach (see Fig. 1) by incorporating existing and applicable knowledge. Then, in Section 3.2 we present the high-level logical software architecture, concerned with the functionality of the system. Section 3.3 discusses how the architecture is embodied with concrete technologies.

3.1. Industrial standards

Case studies like the ones described above indicate real challenges in discrete manufacturing in the era of mass-customization and personalization. The analysis of all pilots in the HORSE project resulted in a set of requirements with a common goal to automate processes, increase flexibility and guard safe human-robot collaboration [30]. The requirements were categorized to ensure homogenization and avoid the risk of developing case-specific systems. Fig. 7 shows the conceptual requirements framework, divided into two main categories: functions for the integration of robotic and human activities in individual work cells (Activity Functions – AF, left part of taxonomy), and functions for horizontal and vertical process integration (Process Functions – PF, right part of taxonomy).

The main functions have been further decomposed into more detailed system functions which are then translated into concrete functional and non-functional system requirements [29]. These guided the design of the conceptual system presented in next section.

3. HORSE system

This section describes the HORSE system in its complete form; from a conceptual reference architecture to the technology realization. Before, though, we present the system, we discuss in Section 3.1 how its design is conceptually aligned with industry standards. This alignment guards the rigor of our DSR approach (see Fig. 1) by incorporating existing and applicable knowledge. Then, in Section 3.2 we present the high-level logical software architecture, concerned with the functionality of the system. Section 3.3 discusses how the architecture is embodied with concrete technologies.

3.1. Industrial standards

Case studies like the ones described above indicate real challenges in discrete manufacturing in the era of mass-customization and personalization. The analysis of all pilots in the HORSE project resulted in a set of requirements with a common goal to automate processes, increase flexibility and guard safe human-robot collaboration [30]. The requirements were categorized to ensure homogenization and avoid the risk of developing case-specific systems. Fig. 7 shows the conceptual requirements framework, divided into two main categories: functions for the integration of robotic and human activities in individual work cells (Activity Functions – AF, left part of taxonomy), and functions for horizontal and vertical process integration (Process Functions – PF, right part of taxonomy).

The main functions have been further decomposed into more detailed system functions which are then translated into concrete functional and non-functional system requirements [29]. These guided the design of the conceptual system presented in next section.
scope of the HORSE project is the ones of the discrete manufacturing domain, as we have already mentioned in Section 1.2 Finally, Level 0 represents the actual, physical production process, as executed by humans and industrial hardware (such as robots).

Various, existing and mature information systems types can be placed in the four levels of the hierarchy. A mapping can be found in [35]. In Section 3.3, we discuss how the developed HORSE system is mapped to these levels.

3.1.2. RAMI 4.0

RAMI 4.0 [33] was selected as a reference architecture that conceptualizes Industry 4.0 aspects in various dimensions. RAMI is generally accepted in the manufacturing domain as a contemporary framework to structure developments in smart industry. In Fig. 9, we show the intended coverage of the RAMI 4.0 framework by the HORSE architecture design (by the red added boxes in the figure). This can be summarized as follows:

- In the life cycle & value stream dimension, the HORSE architecture mainly aims at the actual production processes of manufactured goods, as well as on the design of these processes, leaving out of scope maintenance processes.
- In the hierarchy levels dimension, the HORSE architecture focuses on work centers (production lines or combinations of production lines), workstations (manufacturing cells) and control devices. The architecture provides embedding at the enterprise level (and possibly the connected world level), but does not provide detailed architecture support for this.
- In the layers dimension, the architecture focuses on the functional to integration spectrum. The HORSE project does pay attention to business aspects, but this is not part of the system architecture design (but of the exploitation model design). The asset level is left out of scope as the HORSE architecture does not consider internal specifications of the assets but cares about their functional usage.

By integrating the rigorous knowledge and the relevant input from the practical environment, we present in the next section the high-level logical architecture.

3.2. High-level logical architecture

The problem and requirements analysis had made apparent that the HORSE system would be complex. Trying to address all functionality concerns (from the end-user perspective) and all technical concerns (from the software developer and software integrator perspectives) in one go, would inevitably lead to a messy system specification and thus to a poor system design. Therefore, the system design process should be based on structured framework(s) that create proper separation of concerns throughout the various development stages.

The framework from Kruchten [36] provided the structure during the phases of the architecture design and the software realization, sequencing the activities during the project, and involving the right stakeholders in each step. A contemporary version [37] of the 5-aspect framework of Truyens [38] was used to consider the different aspects of the specification of the system. The application of these two frameworks and the entire architecting process are described in details in [39, 40]. Here, for the purposes of this paper, we focus on the logical software and data aspects to present the functionality of the system (current Section 3.2) and on the platform aspect to discuss its embodiment (Section 3.3).

3.2.1. Logical software architecture – level 3

From the requirements analysis and the analysis of RAMI 4.0, two observations had been identified with respect to supporting a smart manufacturing environment; First, there is a significant difference between the functionality of the design of manufacturing activities and the support of execution of the designed activities. This is in accordance to the drawn red boxes in the ‘Life Cycle and Value Stream’ dimension of the RAMI 4.0 framework shown in Fig. 9. Second, there is a significant difference between controlling activities in production line or area level on the one hand, and in work cell level on the other hand. In the latter case, individual characteristics, e.g., real-time constraints or tasks performed by a single (team of) actor(s) in a single location are taken into account. In the former case, all individual characteristics are considered.

This second difference coincides with the transition between the work centers and station values in the “Hierarchy Levels” dimension of the RAMI 4.0 framework in Fig. 9.

These two observations led to an initial separation of concerns in two dimensions, by distinguishing between “Design”/“Execution” phases, and “Global”/“Local” levels. Decomposing the functionality of the
HORSE system across these two dimensions and further detailing it into aggregation levels led to the high-level logical software architecture of Fig. 10 (which we label as Level 3 – Level 0 represents the HORSE system as a monolith, i.e., without internal structure, Level 1 the system after decomposition across Global/Local levels, and Level 2 after further decomposition across Design/Execution phases). The functional modules of the architecture at level 3 are briefly described below:

3.2.1.1. HORSE Design Global. The HORSE Design Global subsystem covers functionality to design smart manufacturing processes at the global level, i.e., at the site, area and production line levels. We distinguish three modules:

- **process design**, covering functionality to specify manufacturing processes in terms of process models, i.e., what is the sequence of activities and which agents are involved (specifications of roles);
- **agent design**, containing the functionality to specify the characteristics of agents. Agents can be humans, e.g. operators, or automated agents like robots;
- **shop floor design**, containing the functionality to specify the entire production area/site, both in terms of physical layout and safety aspects of all production lines/work cells and their inter-connections.

3.2.1.2. HORSE Exec Global. The HORSE Exec Global subsystem contains functionality to execute manufacturing activities across work cells, i.e. at the site, area and production line levels. This includes two main modules:

- **global execution**, responsible to execute manufacturing processes. It retrieves process definitions from the process/agent/shop floor database that have been created by the design modules and automatically executes them in runtime;
- **global awareness**, monitoring the global state of the environment to guarantee the overall safety. It observes what is happening during the execution of the processes and in case of safety hazards communicates with the global execution module to interrupt them.

To make the implementation decisions regarding the HORSE Exec Global subsystem independent to the HORSE Exec Local subsystem, we include an abstraction layer (Exec Global Abstraction layer) that eases the communication of these two subsystems.

3.2.1.3. HORSE Design Local. The HORSE Design Local subsystem covers functionality to design manufacturing tasks at the local level, i.e., at the work cell level. The subsystem involves four main modules:

- **task design**, containing the functionality to design a manufacturing activity spanning a work cell, which can consist of multiple steps and which require agent(s) (human, automated, or a hybrid team of them) to execute them;
- **human step design**, covering the functionality to design and specify manufacturing steps that are performed by a human agent;
- **automated step design**, covering functionality to design and specify manufacturing steps that are performed by an automated agent;
- **work cell design**, containing functionality to support the physical design of a work cell.

3.2.1.4. HORSE Exec Local. The HORSE Exec Local subsystem contains functionality to support the execution of manufacturing activities within individual work cells, i.e., at the work cell level. It includes three main modules:

- **local execution**, responsible to control the actual execution of manufacturing tasks and steps performed by (a team of) agents;

![Fig. 10. HORSE overall logical software architecture, aggregation level 3.](image-url)
• **local awareness**, covering functionality to observe the physical status of a work cell, by receiving signals from sensors, cameras and human interface devices, analyzing them and notifying either the local execution module or the global awareness module;

• **local configuration**, containing functionality to configure resources on the physical shop floor, i.e., in the execution environment (as opposed to the functionality in the HORSE Design Local which typically happens in a design office). The configuration typically involves setting parameters that are very closely linked to the physical execution environment (such as teaching a robot by demonstration).

Similarly to HORSE Exec Global, we include an abstraction layer in the interface to the HORSE Exec Global subsystem. An explicit Hardware Abstraction Layer is also included to shield design choices for the functionality in the Local Execution. Local Awareness and Local Configuration modules from technical specifications of connected devices, such as robots and AR devices.

We need to mention here that after the design of the logical software architecture in both levels 2 and 3, we confronted it with the set of top-down functional requirements (FRQ) (as specified in [29]). The purpose of such a confrontation is to check:

• whether each functional requirement is covered by at least one module in the architecture (completeness);

• whether each architectural module is required by at least one functional requirement (minimality).

More information about the complete confrontation can be found in [40], where the requirements check shows that there are no unsupported requirements and no superfluous modules.

### 3.2.2. Logical data architecture

At aggregation level 3, the HORSE logical software architecture is still in an abstract level in which any data structures do not determine the functionality of the software structures. To further decompose the software architecture, where things get more detailed, we should first look on the data architecture as it influences software design and implementation choices as well. We present here the data architecture from a logical point of view (which is the basis of the development of a concrete data model, e.g., in the form of a relational database scheme), consisting of the following concept models:

• the **agent** concept model, which specifies the concepts and relations between concepts that describe actors in a manufacturing context, i.e., entities that can perform manufacturing activities;

• the **activity** concept model, which specifies the concepts and relations between concepts that describe the activities to be performed in a manufacturing context by agents;

• a **shop floor** concept model, which specifies the physical environment in which a manufacturing process is executed;

• the **event** concept model, which specifies the concepts and relations between concepts that describe events that require reactions in a manufacturing context; this concept model is included because monitoring and safety are important aspects in HORSE.

The high-level logical data architecture integrating the main elements of the four concept models is shown in Fig. 11. The details of each of the four individual concept models can be found in [39], [40]. In Fig. 11, we see that manufacturing processes, executed on a shop floor, consist of a number of tasks (executed in work cells), which in turn consist of a number of steps. A task can involve multiple agents, which are formed into teams to perform it. Each step of the task is then mapped to specific members of the team. We also see how events can be either activity-related (e.g., a task exceeds a desired maximum execution time) or agent-related (e.g., a mobile robot signals its low battery capacity). An event is linked to its use, where it is processed, for example to make a decision or to store data in a log.

### 3.2.3. HORSE logical software architecture, aggregation level 4

The refinement of the logical software architecture into aggregation level 4 refers to all main modules of level 3 (see Fig. 10). For reasons of brevity, here we focus only on the execution modules, both global and local, omitting the design modules. Global modules on the execution side are important for the actual support of manufacturing activities at a global level, i.e., at the site, area and production line levels. Local modules on the execution side are interesting since they directly link to the physical environment that a CPS supports.

#### 3.2.3.1. Exec Global Modules – aggregation level 4

Fig. 12 shows the
overview of the Exec Global software architecture as a decomposition of Global Execution and Global Awareness modules of aggregation level 3 (Fig. 10). Production execution control is the core module for executing the designed manufacturing process models (designed in the Exec Global modules). A process engine takes care for the automation and orchestration of the activities by selecting the best (team of) agents to perform a task. The Worklist Delivery module supports task delivery to agents. Production Execution Monitoring module supports real-time monitoring of execution in terms of processes and orders statuses, and agents availability (human and automated). Event Processing module handles all events at the global level and those propagated from the local level.

3.2.3.2. Exec Local Modules – aggregation level 4. Fig. 13 shows the overview of the Exec Local software architecture as a decomposition of Local Execution, Local Awareness and Local Configuration modules of aggregation level 3 (Fig. 10). The Hybrid Task Supervisor (HTS) synchronizes the activities of a team of multiple agents performing a task within an individual manufacturing work cell. The agents are controlled through the two Step Execution modules. These modules have interfaces (abstraction layers) to the physical agents, which we place in the Hardware Abstraction Layer. The Local Awareness subsystem consists of five modules; the Sensing Supervisor and Local Perception modules are used to physically observe events or objects in a work cell. The Deviation Monitor is responsible to identify value anomalies in sensor or camera.
inputs. The Augmented Reality module provides AR functionality to humans. The Local Safety Guard is responsible for the safe execution of activities. The Local Configuration subsystem consists of a shop floor Mapping module, a Calibration module for auto agents, a Robot Teaching module and an AR Config module.

3.3. Technology embodiment

We present below the type of software that implements the functionality of the logical software modules of the system architecture. We start with abstract classes/types of software, mentioning specific technologies used in the HORSE project when necessary. We also discuss the main concepts of deployment when realizing the system.

3.3.1. Software platform

The design, enactment and monitoring of manufacturing processes is realized by a Manufacturing Process Management System (MPMS), based on typical Business Process Management Systems [41] (BPMS, mostly applied in the administrative domain). The MPMS covers global functionality of the HORSE system (Process Design and Agent Design modules from HORSE Design global and all modules of Global Execution). The abstraction layers are realized with a cyber-physical middleware, necessary to connect all different technologies. The Hybrid Task Supervisor (HTS) software is responsible for designing the detailed steps of a manufacturing task (covering the design phase) and synchronizing the activities of the agents of the team that perform the task (covering the execution phase).

These three software systems are the backbone of the HORSE system, able to support the manufacturing operations in an enterprise. This is in alignment with the reference functional hierarchy of manufacturing control of the IEC62264:2013 (as shown in Fig. 8). In Fig. 14, we map the core HORSE software components onto the hierarchy levels of the standard.

Note that MPMS spans over levels 3 and 4, as we believe that such an orchestration system, for global, end-to-end process management, can cover both business and manufacturing functions in a unified process management approach in manufacturing [35] (for example covering functionality that an Enterprise Resource Planning (ERP) system can offer and/or easing the integration of an ERP with a Manufacturing Execution System (MES)).

In Fig. 14, we include also technologies of the local level, which we briefly describe in the full software stack below.

- Manufacturing Process Management System (MPMS)

Such a system was built in the HORSE project on top of Camunda BPMN\footnote{https://camunda.com/}, an open-source workflow management platform. Camunda Modeler covers functionality of the Process Design and Agent design modules. Processes are modeled with the Business Process Model & Notation 2.0\footnote{https://www.omg.org/spec/BPMN/2.0/} (BPMN 2.0), which we believe is suitable to model process in both level 4 (i.e., business processes) and level 3 (i.e., manufacturing processes) of the IEC function hierarchy levels [42]. The Camunda process engine and decision engine, enhanced by us with sophisticated mechanisms for dynamic resource allocation [43] and exception handling, cover the production execution functionality. Camunda Cockpit web application covers the process monitoring functionality (we enriched the standard application with plugins that show extra information such as active task durations). Camunda Tasklist web application for human agents covers part of the Worklist Delivery functionality (for the cases task items are directly assigned to humans without intervention of the HTS).
- **Database Management software**
  
  Used for persistence of the data stores of the architecture. In HORSE, we used PostgreSQL Server, it can be any other database system.

- **Cyber-physical Middleware, to connect the various systems of HORSE**
  
  In HORSE, a Websocket-based Message Bus was used based on the Open Services Gateway initiative (OSGi) [44] specification. OSGi is a modular system architecture and a service platform that implements a complete and dynamic component model for general module interconnection. It also provides a universal publish-subscribe messaging bus for communication among system modules. The Message Bus is part of proprietary solutions of a technical project partner, adapted for the needs of the HORSE project. Furthermore, tailored-made OSGi Applications can be used, as software packages that offer powerful and sophisticated component management and interoperability, as well as context-aware assistance of agents (workers, robots) on the production floor in the execution of their tasks.

- **Hybrid Task Supervisor (HTS) software**
  
  The role of such software is to design and synchronize execution steps by agents. Typically, state machines can offer the detailed execution of robotic steps. FlexBe3F [45] is such a system and was used in HORSE project. It features OSGi plugins to existing OSGi nodes, Robot Operating System (ROS) [46] integration to robots and interfaces to industrial equipment. FlexBe was customized and extended by a project partner to be able to connect to other HORSE components.

- **Hardware interface software**
  
  - ROS is a commonly used, open-source, meta-operating system for robots and provides functionality such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management [47]. Within ROS environment, several applications can be used: ROS MoveIt! [48] for force feedback control to enable more precise handling (gripping, placing, etc.) of objects; ROS Kinetic for simple operating system functionality and the interfacing between the above-mentioned ROS applications and the robot. These were used in HORSE as well, with the needed adaptations and customizations.
  
  - Technology-specific robotics interfaces. Depending on the type and brand of robots that a manufacturing organization utilizes, various software platforms can be used for hardware interfacing.

  - OPC Unified Architecture (UA) [49] is used as the interface to advertise and invoke robotic services.

  - Other software packages that are normally used in manufacturing environments with advanced robots are: software for advanced robot motion planning, to cover functionality of the Local Safety Guard module. To provide the processing power required by such software, a parallel computing platform and application programming interface model is required. Such a high-performance processing platform enables real-time 3D projections.

  - Software to support augmented reality.

Focusing now on the local execution level, which mostly represents the cyber-physical aspect of the HORSE system, we summarize in Fig. 15 the technology stack.

The software components mentioned above were developed during the HORSE project, either by configuring proprietary software, extending existing open-source solutions or building custom applications from scratch (such as a tool for configuring the safety behavior of production resources, as discussed later in Section 4.3). A list with the available software packages and corresponding developer is available on project’s website.

---

4 KUKA Sunrise is an example of robotics interface, used in the HORSE project, providing all functions to operate lightweight KUKA robots.

5 GPU Voxels [50] has been used in the HORSE project to provide advanced robot motion planning.

6 nVidia CUDA [51] has been used in the HORSE project to provide computational power to GPU Voxels.

7 Light Guide Systems (LGS) was adopted and customized in the HORSE project by a project partner.

8 [http://horse-project.eu/How_to_get_access_to_HORSE_Prototype_System](http://horse-project.eu/How_to_get_access_to_HORSE_Prototype_System)
3.3.2. System deployment

The HORSE system has been realized for deployment in the pilots and the open call experiments. The backbone systems (i.e., MPMS, Message Bus and HTS) have been deployed in all use cases. Case-specific components have been deployed depending on the requirements of a specific application case (as illustrated in Section 4).

Each of the components have been physically deployed on pilots’ infrastructures. Their deployment, in order to make an integrated running system that executes a scenario, followed the hierarchy of the architecture, by keeping the same separation between global and local modules. This is because global modules can support operations at the site, area and production line levels, while local modules support operation at the work cell level. That also means that there can be one global deployment, integrated into many local ones.

The websocket Message Bus also follows the global-local separation. The server side consists of Broker modules (i.e., one global Broker and many local ones in case of multiple local deployments), responsible to forward the messages to the designate recipients, and the Dispatcher module, which mediates between the brokers, e.g. between the global broker and the local ones. The rest HORSE components, both global and local, are deployed as client nodes of the Message Bus (i.e., each of the components have implemented functionality to register themselves as clients in the corresponding broker). This approach is illustrated in Fig. 16.

Per decision, the Dispatcher is included within the global deployment since there is only one Dispatcher in the entire system.

4. Demonstration

In this section, we discuss how the HORSE system addressed the challenges of the use cases described in Section 2.1, by presenting the physical demonstrators.

4.1. BOSCH (BOS)

The automated visual inspection and packaging of WSA parts were realized with the orchestration of activities of a robotic arm and a vision system (camera) (design of the work cell visualized in Fig. 17). The robotic arm picks-up a WSA from the conveyor belt, moves it under the camera and rotates it accordingly (to specific angles and points) so as the camera performs the inspection. In case of no detected defects, the robotic arm places the part into the packaging box (taking stacking and orientation into account). In case one or more points are assessed as faulty, a worker is notified (with a beacon light (industrial light bulb) on top of the work cell and/or a message in a handheld smart device), and the robot presents the part to him. The AR system highlights the (possibly) defect points and the worker either fixes them (if possible) or indicates the points that cannot be fixed (for further analysis). In the latter case, the robotic arm places the part in a carrier from which the worker can discard it. Video footage of the executed process is available online.

The HORSE logical software architecture led the implementation of the integrated CPS by dictating the functional communication among the relevant modules needed to realize the system. Focusing on the execution phase (omitting the design phase), we illustrate in Fig. 18 the main communication flows of the scenario (by scenario we mean the sequence of activities that we described in the previous paragraph and has been modeled as a process model in BPMN 2.0 in the modeler component of MPMS – which realizes the Process Design module of HORSE Design Global of Fig. 10. This process model is then used as input by the process engine of MPMS – which realizes the Production Execution Control for the actual execution of the scenario. Note that Fig. 18, with the relevant for the pilot modules highlighted in light blue, is an

Fig. 16. HORSE components deployment approach.

Fig. 17. BOS demonstrator – design of work cell for automated visual inspection of WSA parts.

9 https://www.youtube.com/watch?v=P6XbSHQIj_s.
instantiation of the complete HORSE Exec logical software architecture (concatenation of Figs. 12 and 13). The normal flow of the scenario refers to the task assignment messages from the global level to the local level and subsequently to the agents (green path in Fig. 18), and the task completion messages from the agents back to the global level (blue path), for the regular cycle of non-defected WSA parts (i.e., visual inspection and placing in packaging). More specifically, the Production Execution Control selects the most suitable agent (with the dynamic agent allocation mechanism [43]) for the next task(s) of the process and sends task assignment messages (JavaScript Object Notation (JSON)-formatted payload messages according to the HORSE Message Bus specifications) to the HTS. HTS, then, initiates the action of the agents (e.g., asking the robot to pick-up the next WSA from the conveyor belt and prepare it for inspection). When the agent performs the actual task (e.g., WSA is picked-up and is positioned in front of the camera), HTS informs back the Production Execution Control that this task is completed and so, the workflow can advance to the next one (e.g., performing the visual inspection). In case the visual inspection indicates that the part might be defected, an operator is called and the robot presents the part to him. With the help of an AR system, the possibly defected points are highlighted to help the operator recognize them (brown path – the continuation of the communication is the task completion (i.e., the operator either fixes the part or decides on its quality) through the HTS back to Production Execution Control, which is part of the normal blue path). In case something goes wrong (e.g., robot stopped its motion), the Sensing Supervisor informs the HTS through the Local Safety Guard, which in turn informs the Structured Exception Handling module of the Global Execution in order the Production Execution Control takes action on a global level (e.g., call an operator to check or takeover the task)). This is indicated with the purple path. Note here that the architecture indicates that exceptions (and events in general) that should be propagated to the global level should first go through the Event Processing module of the Global Awareness and then to the Structured Exception Handling module. However, for the scope of the pilot it was decided that the Event Processing module was not required. In case an exception or a safety risk needs to (or can) be tackled locally, then there is no need at all to propagate it to the global level. Think, for example, of the case where the operator gets too close to the robot while this is still moving. A Human Tracking submodule (which is defined in the Software Aspect Level 5 of the architecture and can be seen in [40]) of the Local Perception module informs the Local Safety

Fig. 18. EMG data from five different testers carrying a 2kg weight.

Guard through the Sensing Supervisor. Then, the HTS is informed which
takes action (e.g., pauses the motion of the robot or slows its speed
down). This logical message flow is shown with the red path. Fig. 18
shows also the flow of messages from an external system, for instance the
signal from the conveyor belt that the next WSA is on place for being
picked-up, through the Device Abstraction and the Sensing Supervisor
modules towards the Message Bus (from where Production control can
catch as an event and trigger the next action of the scenario). This flow is
shown with the yellow path.

All the relevant modules of the HORSE Exec logical software archi-
tecture needed to implement the BOS pilot scenario were realized with
the developed components that were listed in Section 3.3.1. Projecting
those to the technology stack of Fig. 15, and adding the Global com-
ponents (as Fig. 15 refers only to the Local part) we get the concrete
technology task for the CPS for the BOS pilot, shown in Fig. 19. The
Extented ROS FlexBe realizes the HTS. Note that the external systems,
such as the vision system, the beacon light and the conveyor belt, were
integrated to the HORSE system through an OSGi adapter (specifically
implemented for the BOS pilot). For instance, a beacon light, connected
to the local programmable logic controller (PLC) system and managed
over OPC-UA (over PLC) protocol, can be turned on by the HORSE
components through an adapter that translates Message Bus messages
onto ones that the PLC can understand.

As we mentioned above, MPMS has been used to sequence the ac-

tivities of the workflow, leaving the detailed synchronization of actions to
FlexBE. Fig. 20 illustrates the distinction of a task with its corre-

designed software architecture [11].

Fig. 19. EMG data for various weights carried by the left and right arm.

case a worker crosses the safety zone).

MPMS is also responsible to allocate activities across work cells,

covered with a robotic arm mounted on it fetches necessary parts from the
storage area to the work station (Fig. 22). The realized system for this
scenario includes MPMS for the process orchestration, the AR system for
the augmented instructions, the mobile robot controller software and
the Message Bus for message communication. Depending on the tool
type, MPMS triggers the AR system directly (there was not need to

\[\text{https://www.youtube.com/watch?v=hD1vqzykLkU.}\]
include a HTS here), which retrieves the corresponding instructions and starts projecting them on the workstation. MPMS also dictates the actions of the mobile robot. In case it runs out of battery or raises an alert, MPMS dynamically allocates transportation tasks to operators (the events raised by the agents through the Message Bus are caught by MPMS which executes the allocation algorithm to match task requirements with agents’ skills, abilities and availability [42]). Video footage of the executed scenario is available online12.

Regarding the automated profile stacking (part of “Production area 1” process model of Fig. 21), a universal robot has been used to organize the stacking in bins after P1 (Fig. 23(a)). When such organized bins are transported to P2, another robot can more easily pick them up and hang them on the racks (for the galvanization process). The automated profiles hanging, shown in Fig. 23(b), is a promising, huge improvement compared to the manual hanging (see Fig. 5(b)). The main components used in this scenario are the MPMS for enacting the tasks, FlexBE for triggering the robots, and the Message Bus to enable the message communication.

4.3. Guided Safety

The new production line of DENSO was tested in a prototype environment consisting of one manual and two automated assembly working stations (Fig. 24).

The sequence of activities has been modeled and enacted by MPMS, taking into account exceptions and errors. MPMS Cockpit (shown on the screen on the left part of Fig. 24) offers real-time production monitoring. The HTS has been realized with both FlexBe and Drag&Bot13 (for each of the two automated stations). Message Bus offers the message communication. Robot controllers based on ROS interact with the HTS components. The safety of the workers is achieved through the usage of laser scanner covering various work areas. A tool for selecting the appropriate safety behavior of the production resources has been implemented and demonstrated14. An early prototype of a configuration tool for visualizing and editing safety related information using augmented reality has been demonstrated15 as well. The deployment architecture is shown in

12 https://www.youtube.com/watch?v=bqTDEZvOdVI.
13 https://www.dragandbot.com/.
14 https://owncloud.fraunhofer.de/index.php/s/7oT9CVlC3ym8yIQ.
15 https://owncloud.fraunhofer.de/index.php/s/wUeBNPkl2wNaylA.
For addressing the challenge of easy re-configuration of assembly scenarios, a plugin application on MPMS Cockpit (ShiftManager) has been implemented, shown in Fig. 26. With an intuitive way, the production line manager can configure which manual tasks shall be performed by workers and which specific robotic programs by the automated stations. Also, he can configure to bypass a station during the assembly process (e.g., when it is down for maintenance). MPMS process engine utilizes such configuration during runtime for the resource allocation.

Video footage of the realized system is available online. \(^\text{16}\)

In the next section, we discuss the validation of the HORSE system, after its realization and demonstration at real-world use cases.

5. Validation

As we have already mentioned, the design and implementation of the HORSE framework follows the DSR framework, and as such, it aims at solving practical problems. The initial user requirements (stemming from the practical problems and needs) had to be validated by end-users through the designed artefact, i.e., the CPS which is based on the reference architecture. In HORSE project, there was a continuous and iterative process of testing and validation of developments (i.e., from requirements into architecture design, into components design, into their implementation and integration, and the final application). In this section, we first introduce the integration and testing process that was followed in order to achieve working components and integrated systems. Then, we discuss the validation of the developed components and systems, as these were deployed in the initial three pilots [\text{52,53}] and the seven open call cases [\text{54}]. We also discuss important, in a complex manufacturing setting, non-functional requirements (NFRQ), the

---

fulfillment of which makes the use of the system valuable. More specifically, we discuss how heterogeneity, modularity, portability and response times, as part of efficiency and safety, were satisfied in the HORSE framework. Afterwards, we highlight a few important take-away messages for the application of the framework, as useful insights from our experiences in the course of the HORSE project. Finally, we discuss a couple of limitations of the framework that we see as interesting points for future research.

5.1. Integration and testing process

The ultimate platform validation should be done by the end-users at the designated pilot test sites. But since the allocation of industrial equipment and human resources at the manufacturing sites can be very difficult and expensive, for deploying especially a complex system as the HORSE system, well-designed, correctly and timely executed testing procedures are highly valuable for the early discovery of errors in the components’ design and implementation. The internal prototypes were developed, tested and validated at a test environment and with test equipment, through the process illustrated in Fig. 27 (the orange boxes denote interaction with the test integration infrastructure).

The process is iterative and incremental. Each iteration starts with analysis of the results of the previous iteration. For the very first iteration the analysis is done based on the output of the initial requirements [29]. Based on the analysis, the component developers design the features that their component should include to satisfy the requirements, setup the development environment, implement the features and perform internal validation (e.g., through unit tests, code review, etc.). This testing and internal validation can lead to extra iterations in the features development before submitting the code into the Version Control System (VCS) with a “QC1 OK” mark (Quality Check 1), ready for the integration testing. Independently to the components development, auxiliary applications and scripts, aimed to verify and validate specific capabilities of the HORSE system and its components, are developed as Test Cases (TCs). The design of TCs includes formal descriptions of the use cases (as identified in [29]), specification of the tested components, initial conditions and input data, interaction with the tested modules and expected output data. Examples of test cases are shown in Appendix C. Test cases are developed with the help of development tools, tested internally by the teams to ensure quality and made available for use by the testing environment. The steps of the components’ development and the test cases’ development are illustrated in Fig. 28.

The test configuration lists all components that are eligible for integration testing. At predefined periods (e.g., at night) or conditions (e.g., new QC1-tagged component update), a continuous integration (CI) server initiates the compilation and consistence check of the QC1 tagged component update. It executes the building tools and scripts to compile and build temporary instances of these components. In case of success, the component is tagged as “QC2 OK” (Quality Check 2). Otherwise the
update is tagged as “QC2 Fail” and a report is sent to the developer. The list of the test-ready components (the test configuration) is updated with the new version of the freshly certified component. A deployable build of the component is stored in an artefacts’ repository. The CI Server retrieves, then, those deployable builds and deploys/install them on the test platforms. Finally, the testing application attempts to run the available TCS on the temporary builds. Each TC feeds the testable components with the predefined input data, interacts with the system simulating the external systems, agents and devices, and keeps record of the processes and communication. The debug information is collected and a test protocol is created. Once the TC execution ends, a comparison of the observed end conditions and the expected ones is done. The test report is created and stored in the code repository. In case of problems or misbehavior, a ticket in the Issue Tracking System is created.

After the successful completion of all test cases, the test configuration is considered ready for trial at pilot sites (i.e., in industrial conditions.
with real equipment). There, the TCs are executed on the stable prototypes in order to be validated by the end-users.

The overall results of the implementation and test coverage are summarized in Table 1. We can conclude that all functional requirements have been addressed by the HORSE components. A great part (84%) have been completely implemented while the rest small part (16%) have been covered partially. The implementation coverage of the non-functional requirements is 70%, where just one requirement has not been implemented (more specifically – “NFRQ11: the robotic actor should be able to improve its task performance while it receives new task performance data”), which was considered as of low priority. Regarding testing coverage, almost all FRQs (34 out of 37) have been validated with test cases executed by the HORSE components or as features of the integrated products. It should be pointed out that the testing coverage for the high priority requirements is about 96% (22 out of 23). The validation of the rest FRs is related to more intensive efforts (dealing with more product parameters, calculation and implementation of complex rules, processing of more sensor data) and integration of additional sensors and agents. The achieved coverage of NFRQ is rather high (85%), considering that the HORSE platform is designed as a prototype for controlled use and demonstration.

The detailed mapping and coverage of the requirements (as listed in [29]) by the test cases are available at the project’s deliverable on test reporting [55].

5.2. Components and system deployment validation

We validated both the components individually (i.e., without any integration, but as standalone components with simulated inputs/outputs) and the complete, integrated system as deployed within the initial three pilots and the seven open call experiments. The complete system
validation took place in two rounds, i.e., at an intermediate stage (for the initial three pilots) and at the final system deployment (the feedback from the first round was used to refine the system at its final version). Details can be found in project deliverables [56–58]. Here, we briefly discuss the selected validation criteria and examples of the final end-users results.

We looked on well-known system and software quality models, such as the ISO/IEC 25010 [59] and ISO/IEC 9126-1 [60] standards. By following a scenario-based testing approach (ISO/IEC/IEEE 29119-4 [61]), we focused on Functional Suitability and Usability product quality aspects. We also opted for Effectiveness, Efficiency and Satisfaction quality-in-use aspects. For each of these aspects, we selected validation characteristics and their measurable attributes, i.e., metrics.

Table 2 shows the overview of the selected aspects, characteristics and attributes. The attributes were qualitatively assessed with a five-level scale (1-Completely disagree to 5-Completely agree) structured questionnaires and open-text questions (see Appendix D for the description of the attributes and the questions). The attributes were also prioritized by the end-users in order to further evaluate the scores of the components for each attribute. The weight of each attribute was estimated with pairwise comparisons [62] and the eigenvalue method [63] was used to calculate the average weights.

Fig. 29 shows the evaluation results of the MPMS component, as assessed within the initial three pilots (individual results are shown in Appendix E). In general, the component covered the expected functionality of the pilots. As in TRI pilot, MPMS was also used to model the complete overview of processes, in a rather high abstraction level, without detailed description of all tasks (e.g., “Production area 3” process model of Fig. 21 was not implemented as part of a CPS), the component scored quite low in some aspects. For instance, the completeness of description (i.e., degree to which the system functions are understood by the user) was evaluated as low, as it was hard for users to grasp new concepts such as BPM, BPMN process modeling, process engine, etc. Moreover, the prototypical character of the component, with a focus on the functionality and not on user interfaces, resulted in a neutral assessment in terms of attractiveness (in all three pilots). Regarding the operational consistency (i.e., degree to which the user finds the system functions consistent), the various subcomponents of MPMS (i.e., Modeler, Cockpit and Tasklist applications) made end-users get the impression that there is some inconsistency in functions (the component was presented and demonstrated as a whole in the end-users for an overall evaluation and not per subcomponent). This was more obvious in TRI and BOS where MPMS was demonstrated with its full-fledged functionality, while in OPSA users got familiar mostly with the Tasklist application (the scope of the scenario was rather small, by having operators confirm a few tasks on the Tasklist application).

However, we believe that the rather neutral reaction with respect to operational consistency should not be perceived as a negative factor for the adoption, usage and usefulness of MPMS as each subcomponent is meant for different type of users – Modeler is for process designers, Cockpit for factory/shopfloor supervisors, and Tasklist for shopfloor operators.

With respect to the low scores in Task performance and Task accuracy aspects, it was clear that the end-users were not familiar with the functionalities of the MPMS component. This was evident from the low scores in the “degree to which the user finds the system functions consistent” aspect, especially in the TRI and BOS pilots. However, the neutral reaction with respect to operational consistency should not be perceived as a negative factor for the adoption, usage and usefulness of MPMS as each subcomponent is meant for different type of users – Modeler is for process designers, Cockpit for factory/shopfloor supervisors, and Tasklist for shopfloor operators.

![Fig. 29. Evaluation scores of MPMS component from three pilots [56].](image-url)
accuracy in the OPSA pilot, we believe that these are mainly attributed
to the nature of the manufacturing process, the low-level of automation
and the characteristics of the physical environment in general. The pilot
refers to a sand-casting foundry which produces tons of iron castings in
different configurations and for various industries. The foundry includes
areas for sand, molds and cores preparation, areas for casting operations
and areas for finishing, fettling and surface treatment operations. These
operations are often performed in an unstructured way, in a dusty and
noisy environment (see Fig. 30(a)). While the cutting process, currently
done manually, can potentially be replaced by a robot (through teaching
by demonstration as demonstrated in the HORSE project, Fig. 30(b)), the
use of MPMS by the operators in their daily tasks is not considered
efficient. Having a screen or a tablet through which operators might
have to get tasks and indicate their completion (in the Tasklist appli-
cation) is rather a cumbersome scenario in such production
environments.

Similar diagrams to Fig. 29 were created for all validated compo-
nents. The performed factor analysis yielded a good overall assessment
for all components [56].

The quality aspects for the individual components validation (of
Table 2) were extended with further aspects, characteristics and attri-
butes for validating the complete system. We added aspects such as
reliability, recoverability, stability, safety, etc. (full list in [57,58]). For
each of the selected attributes/metrics, we applied well-known and
adopted methods to collect and analyze the feedback, such as the Soft-
ware Usability Measurement Inventory (SUMI) method [64], and the
National Aeronautics and Space Administration (NASA) Task Load Index
[65]. Pilots such as TRI (Fig. 31) evaluated the HORSE system as useful
and appropriate for their operations and needs. Detailed evaluation of
the solution implemented for the tool assembly process can be found in
[66]. Even the ones which assessed it as less positive provided useful
feedback for future improvements (e.g., provide more detailed messages
to operators when using the system, improve the user interface of some
components or minimize the latency of the exchanged messages, which
might have an impact on cycle times).

Fig. 30. OPSA foundry – (a) production environment with manual operations and (b) teaching by demonstration and automatic metal cutting demonstrator.

Fig. 31. Pilot case (TRI) evaluation scores of the implemented HORSE system [57].
5.3. Heterogeneity

Various kinds of software and hardware exist in a manufacturing setting. The same holds for the different agents (humans and robots). The HORSE system has been designed and built to be able to cope with this heterogeneity, by adopting a middleware approach as a technical solution. The use of a Message Bus, as a connecting way among heterogeneous components, can solve integration problems that may appear in an enterprise, which typically has both legacy and new acquired systems and technologies, normally from various vendors. The uniform way to exchange messages among HORSE components and the HORSE-OtherTech bridges applications enabled the integration of technologies towards CPS solutions.

All initial three pilot cases of the HORSE project, and the subsequent seven open call experiments managed to integrate and use the system in their industrial environments. That, of course, required a lot of guidance from the HORSE component developers and detailed technical documentations (which in general is an important aspect for adoption of the framework and there is always room for improvements, based on pilots’ feedback).

Fig. 32 shows how two of the total ten cases used the HORSE framework in their effort to overcome their challenges (for the TRI pilot, the case that is in the scope presented in Fig. 32 is the tool assembly process). We see there a list of various information systems, interfaces, hardware and physical entities.

All the ten cases, covering industries such as manufacturing of automotive parts, automated warehouses, advanced coating applications and iron casting foundries, include a variety of software and hardware, different per case. The reader can refer to Appendix F for the full list of all heterogeneous systems. The HORSE reference architecture framework covered all these cases, proving that heterogeneity can be tackled.

5.4. Modularity

The HORSE framework, as a reference architecture, covers as much functionality for hybrid smart manufacturing as possible. However, each manufacturing setting has its own needs and not all subsystem components are required. Of course, there are core components, as has already been mentioned, i.e., MPMS, Message Bus and HTS, but a manufacturing company can select and apply those extra components that fit best.

That is also shown in Fig. 32, where we see that two different pilots used a different set of HORSE components and agents.

5.5. Portability

The human agents in a factory shop floor receive task instructions by the HORSE system via various physical channels, such as computer monitors, fixed screens, portable devices (e.g. tablets), handheld devices (e.g. smartwatches) or smart glasses with augmentation capabilities. Therefore, different interfaces may be needed per manufacturing setting. Similarly, different kind of automated agents require different software control standards. The HORSE system logical architecture satisfies this portability requirement by providing modularity in the interfaces towards these agents [67]. Through the Hardware Abstraction Layer (of Fig. 13), different and alternative interfaces modules can be implemented, depending on the specific setup in an industrial scenario. In the BOS pilot case, for instance, the notification to a worker to perform a task was implemented by showing directly a new task on a touch-screen monitor. However, in case he is busy in another workstation and he does not see the monitor, a beacon on top of the work cell was used as well to notify him.

Another case of portability is the use of different software/technology for providing the right functionality of the logical software architecture modules. For example, in Section 3.3.1, we mentioned FlexBE as the software playing the role of the HTS, both for designing and executing manufacturing steps, as parts of tasks. An alternative can be Drag&Bot, as this was used by Guided Safety open call experiment (discussed in Section 4.3). Similarly, the abstraction layers which were realized with a web-socket Message Bus might be implemented with a different type of middleware that can be more easily or flexibly integrated to existing systems. This was also the case in one of the HORSE open call experiments (see Appendix F for all portability examples). Of course, this is a typical property of a reference architecture, which provides more abstract information for building a system than a
concrete architecture, which includes more context details of the case.

5.6. Response times

As already stated in the introduction, the goal of the HORSE system is to support mixed-agent manufacturing processes; humans can perform their work in the same manufacturing work cell with robots and autonomous devices, without fences, separators, or walls. While this offers a lot of flexibility, it entails safety risks as humans may be harmed by robots. To prevent any (fatal) accidents, the HORSE system includes global and local awareness modules, with response time being an important factor to distinguish their functionalities and responsibilities.

Critical events with sub-second response times that occur within a work cell need to be addressed by the local awareness module. The module should track and analyze, in real-time, physical movements within a work cell (e.g. with laser curtains, 3D-space or thermal cameras). When there is any imminent human-robot collision, it has the responsibility and the authority to provide the right instructions to the involved agents for immediate and effective action (for instance, stopping the robot or raising emergency alerts). In case safety breaches can impact a bigger area than a work cell, or the risks are not real-time critical, the captured events need to be propagated to the global situation awareness module.

This hierarchy is taken into account in the HORSE system design and the physical deployments of all pilot cases, to provide a safe working environment. For instance, in the BOS pilot case, the Sensing Supervisor (Human Tracking) module was responsible to identify whether the operator is close to the robot arm so it could slow down or halt its motion.

5.7. Lessons learned

We discussed above the validation of the final results of the application of the HORSE framework. Through the entire process, though, of designing, implementing, deploying and using the concrete cyber-physical systems in practice, we gained valuable experiences that might be useful for the adoption and application of the framework. We highlight the most important points below:

- The HORSE system comprises of a set of heterogeneous components, developed with different technologies. While the reference architecture shows a way, with clear interfaces, to put them all together in an integrated CPS, the actual integration and the physical deployment of the system requires quite substantial effort from various groups of people, each of them responsible for their part (e.g., developers of MPMS need to set-up and configure the process engine, while developers of an AR system need to set-up projectors, light system, etc.). The role of integration testing before the onsite deployment is crucial to avoid/minimize waste of time and resources. Once the communication flows among the involved components have been defined, as part of the target solution, (like the one presented for BOS pilot in Section 4.1), well-defined test cases, running on robust test infrastructure that mimics the target environment as much as possible, guarantee a first level of satisfactory system validation. Apart from peer-to-peer communication testing (i.e., components communicating bilaterally), communication involving the entire set of components of the targeted solution should be tested. Errors appearing in such entire test scenarios often require the whole team of developers to be involved, as the root cause might be outside the scope of the (set of) system(s) that raised them (for example, an error between HTS and MPMS was caused due to a wrong message from Local Safety Guard to HTS that HTS failed to recognize. Component developers of these three modules had to work together to solve the issue and refine the solution). Of course, while testing message communication with dummy agents or simulators might be successful, teams should not underestimate that the integration of real equipment can also bring up new unexpected errors. For the onsite integration, systems integrators are essential to glue all HORSE components and legacy infrastructure together (based on detailed and clear documentation and instructions created by the HORSE component developers).

- The distinction between global and local levels is important to separate concepts and keep structured hierarchy levels. Modeling a process from a global perspective (with MPMS) provides a good overview of activities to process owners and production managers. Modeling the physical activities with a more detailed view (e.g. with FlexBe), provides a clear view on how things happen, which is important for the process participants (i.e., human operators) as well. However, the line between these two levels is not always clear. That is mostly obvious on modeling tasks and steps. For instance, one could combine the two consecutive tasks, “Move WSA for inspection from conveyor belt” and “Check WSA visually”, of the Fig. 20, into one, e.g., “Pick-up and Inspect WSA”, depending on how much control is desired on the global level. Thus, the granularity of the process models and workflows might be a challenge. However, we believe that it is a matter of agreements between process modelers on the views and the control they want to provide in each level.

- The process-orientated control that a system like MPMS provides, seems to be a useful approach to fill the gap of missing or insufficient process overview, visibility, production status, resource allocation and orchestration, that many manufacturers face on their production sites. The techniques, though, might be hard for some group of people to grasp and adopt. The use of BPM as a language to model manufacturing processes, while is gaining a lot of interest in the manufacturing domain [68-71], requires experienced process modelers. Moreover, operators without knowledge of the notation might have difficulties to understand the modeled workflows. However, the expressiveness of the notation concerning integration and execution semantics [72], together with the implementation solutions we provided within MPMS, should not hinder manufacturers from considering our approach.

- The understandability of the functionality and usage of the components is important to ensure that the system will be valuable and useful. On the one hand, operators want to understand how to use a system, which in most cases will comprise of various components. Providing clear instructions and (error) messages to the end-users on what they have to do in production (as indicated in the feedback sessions), especially in time-pressure environments, is a great factor of successful adoption of the HORSE system. In the same direction, providing flexibility to production managers to easily adapt production scenarios during runtime, is a clear indication of the responsiveness and adaptivity of the system (see for example the ShiftManager solution described in Section 4.3 for easy and fast reconfiguration of production scenarios). On the other hand, when manufacturers desire to alter operations or introduce new ones on their shopfloor (e.g. introducing a new AGV in a warehouse), developers and integrators need to have good knowledge of the functionality of the offered components in order to make the right adaptations (mostly during design phase). The HORSE system, while still not in a mature level, provides good levels of understandability, with further improvements as an important extension point.

- We discussed previously the importance of the response times. The involvement of a level of components in a workcell control of physical actors might impact these. A task assignment message from MPMS, through the Message Bus, through HTS and finally to a robot can cause some latency, as this was practically experienced in some pilots. This has a negative effect not only on performance and efficiency (by increasing cycle times), but also on safety, as reaction to events might be crucial. Thus, the selection of the communication protocols and powerful and robust infrastructures to host the applications are important factors.
The introduction of robots and automated devices in production environments where humans used to work and now have to collaborate with them, poses a lot of challenges to meet the safety requirements. Experts had to take into account strict regulations and provide detailed specifications on the implemented physical solutions. So, apart from designing a concrete CPS based on how the solutions work from a functional perspective (e.g., “Local Safety Guard informs HTS when a human is approaching an auto agent, which in turn should slow down its speed”, as we discussed in Section 4.1 with the safety flow in Fig. 18), separate work is required to specify and select the right robotic solutions to realize those solutions on the physical level (e.g., where the sensors should be placed to easily and safely monitor a safety zone). The framework covers these (safety) specification aspects through the design modules (e.g. Workcell Design module in Fig. 10) but it is obvious that the way the to-be execution modules will function might affect the results/arts-facts that the design modules should give.

5.8. Limitations & future work

The HORSE project, as a research and innovation project, aimed at building prototypical solutions to be validated in real production environments. As such, we consider the developed CPS in technology readiness level 6 [73,74]. Since the emphasis was mainly on testing and proving intended functionality, other aspects such as such robustness, scalability, user-friendly interfaces, and security, were out of main focus. Any criticism from the pilots on these aspects, should therefore be treated with caution for assessing the adoption of the framework.

Apart from these aspects with had low priority and were not covered, the HORSE framework, at its current state, has inherently a few important limitations. Therefore, further developments are essential for its better exploitation, apart, of course, from the points that did not receive the most positive evaluation as we have discussed here. These developments are in the direction of extending the scope of the system for broader usage, and taking advantage of the cloud developments in the Industry 4.0 era for better efficiency. We outline such development points below.

5.8.1. Interoperability/extensibility

In the HORSE project, we mostly tested the system in isolation from other systems, as we started from a limited scope towards the complex task of integrating various technologies. However, to better serve the role of end-to-end, horizontal process management with vertical integration to physical systems, it has to be connected to systems that are already in place such as an ERP or an MES. For example, MPMS could retrieve business orders from an ERP system and then initiate the production (the detailed schedule of which could well be retrieved by an MES). This integration can well be bi-directional –for example, an ERP system can query MPMS to get the (production) status of an order.

Another aspect of interoperability that should be addressed in further development of the HORSE reference architecture, is the extension of the framework by including an analysis phase, apart from the design and execution phases. Running systems, robots, and devices produce a lot of data that can be useful both in short and long-term for optimization of processes. The HORSE framework should include/connect to data collection and analysis tools with the goal to improve the manufacturing operations. Such an approach is followed by a recently started (January 2020) EU Horizon 2020 program, called “Smart Human Oriented Platform for Connected Factories” (SHOP4CF17). In that project, the HORSE framework is a core part for developing an open architecture platform that can support humans in production activities.

5.8.2. Cross-organizational manufacturing

The applications of the HORSE framework mentioned in this paper typically cover orchestration and automation of manufacturing processes within the physical boundaries of a single organization or even a single site. However, nowadays, processes for manufacturing products have an inter-organizational scope, where they span across multiple locations or even different, collaborating enterprises, in a complete manufacturing chain or even manufacturing networks [75].

Cross-organizational manufacturing was the main concept of the CrossWork project [76,77], in which a centralized global business process controls the activities of local processes of multiple organizations. The same approach can be applied with the use of the HORSE framework, where a global process of a manufacturing organization, orchestrated by the MPMS, can synchronize local processes, both within the same organization (but e.g. in different locations) and across other organizations. Such an approach is illustrated with an example in Fig. 33.

And of course, the collaboration among enterprises does not stay only on the business level (level 4 of Fig. 8) but goes in operational and actual control levels through the vertical integration that the HORSE framework provides.

This networked process management [78] is a trend gaining more attention in Industry 4.0 as well [79,80], especially due to technologies such as cloud computing. An inter-organizational approach can lead to full automation in complex manufacturing network environments. Note that where necessary for security or privacy issues, the management of the local and cross-organizational processes can be handled by separate systems, as suggested by the CrossWork framework [76,77].

5.8.3. Cloud computing

The main goal of the HORSE project is to create a framework for smart manufacturing control that is accessible for SMEs, towards their effort to stay competitive in their markets. Often, an SME does not deploy robust information systems solutions (e.g. a commercial MES) or high-end control systems. Moreover, there is often limited usage of advanced robotics, due to lack of expertise and/or financial resources, and thus, full automation is not there yet. This limited availability of on-premise infrastructure and computing resources can be addressed with cloud computing technologies [81]. This means that the HORSE technologies as discussed in this paper should support the cloud services paradigm (as at the moment the solutions were deployed locally, on sites’ premises).

The application of cloud solutions may not influence the logical view of the HORSE reference architecture, in terms of how the designed modules function, but it will definitely influence the physical view (of the Kruchten framework), in terms of how the system will be deployed to exploit computing infrastructure investments. Performance and timing constraints are crucial factors in deciding which services can be brought to the cloud. Modules that require sub-second response times (especially local execution ones) should be better deployed on-premise (local) infrastructures, probably combined with fog computing [82]. On the contrary, modules used during the design phase of processes or modules that are not heavily impacted by internet traffic and communication delays can be hosted as cloud applications, either as Software-as-a-Service (SaaS) or Platform-as-a-Service (PaaS) solutions [83].

The use of cloud computing can also further enable cross-organizational manufacturing as discussed in the second extension point in Section 5.8.2 [84,85].

6. Related work

Many initiatives have emerged to design and implement CPS for smart manufacturing. In the context of our work, we discuss the most important of these below.

The Reference Architecture Model Industry 4.0 (RAMI 4.0) [33,86] was established to structure the evolving technologies and standards in

17 https://www.shop4cf.eu/.
smart manufacturing. It relates concepts of the life cycle, business and hierarchical views of an asset. The Industrial Internet Reference Architecture (IIRA), designed by the Industrial Internet Consortium, consists of four viewpoints to support the design and implementation of an Industrial IoT (IIoT) system [87]. The Smart Manufacturing Ecosystem (SME) [88], developed by the National Institute of Standards and Technology (NIST), provides a complete overview of various lifecycles of smart manufacturing. The framework’s standardization mainly focuses on ICT application systems and how information is exchanged among software applications, omitting infrastructures concepts like IoT and cloud computing [89]. Fraile et al. [90] present and realize the Industrial Internet Integrated Reference Model (I3RM), which integrates features of NIST smart manufacturing standards, IIRA, and RAMI 4.0 reference models and architectural patterns, to facilitate the definition of the system architecture of digital manufacturing platforms. The Internet of Things Architectural Reference Framework (IoT-ARF), designed within the Internet of Things – Architecture (IoT-A) project [91], provides a functional overview of the various IoT components, as well as an information, a communication, and a trust, security and privacy models. However, no clear interaction among the components is proposed, as it is dependent on design decisions. The Software-defined Industrial Internet of Things architecture (SD-IIoT) [92] mainly provides clear communication protocols for data transmission from the physical layer to cloud environments. The architecture, though, does not provide guidance to develop IoT systems. The 8C architecture [93] was proposed as an improved extension of the 5C architecture [94] for CPS for smart factories. The 3 added facets improved the emphasis on the horizontal integration, while 5C focuses mainly on the vertical integration. An example of a developed CPS system is presented but without structured instructions.

Li et al. [89] performed a comparative analysis of most of the reference architectures mentioned above, including also others such as the Intelligent Manufacturing System Architecture (IMSA) [95], developed by the Ministry of Industry and Information Technology of China (MIIT) and Standardization Administration of China (SAC), and the Industrial Value Chain Reference Architecture (IVRA) [96], developed by the Industrial Value Chain Initiative (IVI). They concluded that the construction of the reference architectures is based on three main principles; decomposition into multiple dimensions, focalization by focusing on specific smart manufacturing aspects and concepts and excluding others, and strategic consistency by embodying related national manufacturing strategies and initiatives, such as Industry 4.0. Bader et al. [97] provide a structured analysis of existing reference frameworks, their classifications and the concerns they target. The work of Brouns [98], not only positions numerous reference architectures in the fourth industrial revolution area but also proposes an integration framework for defining a roadmap towards smart manufacturing.

All these initiatives and reference architectures have their

---

**Fig. 33. Cross-organizational, networked manufacturing (inspired by [76]).**
viewpoints, methods and models to structure concepts around Industry 4.0. Most of them though, do not provide concrete solutions to realize operational systems. This lack of clear instructions results in implementation barriers of adopting these reference architectures in practice [99]. An approach to adopt a reference architecture for smart manufacturing is found in [100], in which RAMI 4.0 is used as a guide. The implemented layered architecture is a solution to discover equipment to process operations according to the product requirements, but there is no reference to integration of smart technologies like robotics or AR systems. Other approaches for developing a CPS for Industry 4.0, for instance in [101], which also follows the RAMI 4.0 and the IEC functional hierarchy standard, or in [102], which also supports OPC-UA like the HORSE architecture, put focus on the vertical integration, lacking support on the horizontal integration [103]. Finally, even though maturity models have been proposed for the implementation of Industry 4.0 technologies (e.g., [104,105]), empirical evidence on the actual adoption by manufacturers is rather poor [106].

The HORSE framework not only provides a clear blueprint for integrating heterogeneous smart technologies in hybrid manufacturing processes, but also provides the practical implementation details to realize concrete cyber-physical systems. These systems provide both horizontal and vertical control of processes and actors, taking safety aspects into account. The deployment in different manufacturing scenarios proves also practical application and promising adoption of the framework.

With respect to the process-oriented manufacturing operations control, as we apply it with the MPMS component, various works exist that apply BPM concepts in manufacturing. Approaches like in [107–109] focus only on modeling IoT-aware processes without execution support. Schönig et al. [110] provide an integrated approach that exploits IoT on top of BPM concepts, but the focus is on (mobile) task assistance and decision support systems. There is no discussion on resource allocation, exception handling or process monitoring, as we have applied in a rather full-fledged BPMS for manufacturing domain, i.e., the MPMS. Pauker et al. [111] also apply BPM in smart manufacturing. They do not discuss aspects such as safety or situational awareness that the HORSE system addresses. Subject-oriented business process management (S-BPM) [112] is an approach applied to smart manufacturing for process integration [113]. While practical demonstrations and evaluations have been performed [114], the uptake of the approach is rather limited despite the promises, possibly due to the lack of interest of the used modeling notation (compared to the well-defined BPMN language that we use).

7. Conclusion

In the Industry 4.0 era, many advanced technologies have emerged to further automate production processes and offer the required flexibility to address high demand fluctuations and market trends such as mass customization and personalization. Versatile robots, augmented reality systems, mobile robots, handheld devices, available in a factory shop floor, promise to make manufacturing smarter and more efficient. However, the integration of all these new developments poses many challenges. Systems are heterogeneous, local developments without a broader scope hinder transparency, a large part of information produced by the physical devices is not exploited by information systems, are a few among those. Moreover, the co-existence and collaboration of humans and robots in the same working environment requires adequate attention in terms of safety.

In this paper, we have presented the HORSE framework, as a reference architecture for cyber-physical systems that integrate smart technologies and provide manufacturing operations management in hybrid-actors settings. The framework is a modular architecture with clear subsystems and interfaces at several levels of aggregation, resulting from a structured, hierarchical system design, based on theoretical principles and guidelines. We have discussed the functionality of the developed architecture and presented the technology embodiment for realizing a CPS. Such operational systems have been developed, demonstrated and validated in 10 real-world manufacturing use cases of the HORSE project, providing useful insights on how the framework can be adopted by manufacturers.

The framework provides a backbone of main systems (MPMS, OSGi Message Bus, HTS) that offer horizontal (across work cells) process orchestration with vertical (within work cells) activities synchronization that cover various manufacturing scenarios with different production settings (demonstrated with 10 different concrete scenarios). Moreover, the technology-agnostic middleware approach allows for integration of various systems and technologies. More specifically, (legacy) systems and applications can be integrated either through subscription as clients to the message broker of the OSGi Message Bus or through customized bridges (e.g., the HORSE-Kafka bridge). Similarly, various industrial equipment types and devices can be integrated through agent interfaces. These integration options extend the application of the framework to cover a broad range of scenarios. Of course, there can be settings in which automation with new technologies is hard to achieve (as we found in the OPSA case), but this does not undermine the value of the framework. In these cases, the application of the framework shows that problems outside the scope of the framework have to be solved first, before advanced Industry 4.0 technologies can be applied.

The validation and evaluation of the HORSE framework is deemed as positive, despite the integration efforts to realize the final versions of real-world systems, the prototypical level of the demonstrators and the known limitations. Definitely, further developments, as discussed in this paper, are essential for improved exploitation of the framework. Nevertheless, the HORSE project has clearly demonstrated that the framework already serves as a reference architecture and as a solid basis for building cyber-physical systems that provide efficiency and safe use of robotics in the smart manufacturing era.

Author contributions


Funding

The work described in this paper was part of the HORSE project, and has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement no. 680734. It was also partially supported by the SHOP4CF Project funded by the European Union’s Horizon 2020 research and innovation program under grant agreement no. 873087.

Declaration of Competing Interest

The authors declare no conflict of interest.

Acknowledgment

All members of HORSE consortium and open call experiments are acknowledged for their contribution to the system design, implementation and evaluation processes.
Appendix A

Table A3 lists, in alphabetical order, the abbreviations appeared in the main text.

Table A3
Abbreviations.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGV</td>
<td>Automated Guided Vehicle</td>
</tr>
<tr>
<td>AR</td>
<td>Augmented Reality</td>
</tr>
<tr>
<td>BOS</td>
<td>Bosch</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model &amp; Notation</td>
</tr>
<tr>
<td>BPMMS</td>
<td>Business Process Management System</td>
</tr>
<tr>
<td>CI</td>
<td>Continuous Integration</td>
</tr>
<tr>
<td>CPS</td>
<td>Cyber-Physical System</td>
</tr>
<tr>
<td>DSR</td>
<td>Design and Science Research</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>FRQ</td>
<td>Functional Requirement</td>
</tr>
<tr>
<td>HTS</td>
<td>Hybrid Task Supervisor</td>
</tr>
<tr>
<td>I3RM</td>
<td>Internet Integrated Reference Model</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electrotechnical Commission</td>
</tr>
<tr>
<td>IRA</td>
<td>Industrial Internet Reference Architecture</td>
</tr>
<tr>
<td>IMSA</td>
<td>Intelligent Manufacturing System Architecture</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet-of-Things</td>
</tr>
<tr>
<td>IoT-A</td>
<td>Internet of Things – Architecture</td>
</tr>
<tr>
<td>IoT-ARF</td>
<td>Internet of Things Architectural Reference Framework</td>
</tr>
<tr>
<td>IIoT</td>
<td>Industrial IoT</td>
</tr>
<tr>
<td>IVRA</td>
<td>Industrial Value Chain Reference Architecture</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MES</td>
<td>Manufacturing Execution System</td>
</tr>
<tr>
<td>MII</td>
<td>Ministry of Industry and Information Technology of China</td>
</tr>
<tr>
<td>MPMS</td>
<td>Manufacturing Process Management System</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NFRQ</td>
<td>Non-functional requirement</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>OPC-UA</td>
<td>OPC Unified Architecture</td>
</tr>
<tr>
<td>OSGi</td>
<td>Open Services Gateway initiative</td>
</tr>
<tr>
<td>PA</td>
<td>Production Area</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>RAMI 4.0</td>
<td>Reference Architectural Model for Industry 4.0</td>
</tr>
<tr>
<td>ROS</td>
<td>Robot Operating System</td>
</tr>
<tr>
<td>S-BPM</td>
<td>Subject-oriented Business Process Management</td>
</tr>
<tr>
<td>SAC</td>
<td>Standardization Administration of China</td>
</tr>
<tr>
<td>SME</td>
<td>Smart Manufacturing Ecosystem</td>
</tr>
<tr>
<td>SMEs</td>
<td>Small and Medium Enterprises</td>
</tr>
<tr>
<td>SD-IIoT</td>
<td>Software-defined Industrial Internet of Things</td>
</tr>
<tr>
<td>SUMI</td>
<td>Software Usability Measurement Inventory</td>
</tr>
<tr>
<td>TC</td>
<td>Test Case</td>
</tr>
<tr>
<td>TRI</td>
<td>Thomas Regout International</td>
</tr>
<tr>
<td>VCS</td>
<td>Version Control System</td>
</tr>
<tr>
<td>WSA</td>
<td>Wiper System Assembly</td>
</tr>
</tbody>
</table>

Appendix B

Table B4 briefly presents the HORSE pilot cases and the open call experiments and their projects. More detailed information can be found on http://horse-project.eu/Pilots and http://horse-project.eu/Experiments.

Table B4
HORSE Pilots and Open Call Experiments.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Short Description</th>
<th>Manufacturing company (end-user)</th>
<th>Sector</th>
<th>Challenges</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>BOS</td>
<td>Robotic based quality (visual) inspection and human-robot co-manipulation</td>
<td>Automotive</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Robert Bosch España, Fàbrica de Castellet, Barcelona, Spain</td>
<td></td>
<td>• Product quality with required cycle time</td>
<td>Automated packaging of wiper systems, including artificial visual quality check of parts, to replace current situation (manual)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Customized packaging of wiper systems</td>
<td>• Augmented Reality assistance for manually checking points on potentially assessed faulty parts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Operator’s health conditions</td>
<td>• Orchestration and monitoring of robotic and human tasks,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(continued on next page)</td>
</tr>
</tbody>
</table>
Below are example of Test Cases (TCs). The first one tests a simple task assignment message from MPMS to FlexBe, while the second one extends the first task assignment message with more data.

**TC-EP-004**

**Short name:** Task assignment - simple communication

**Key components:** MPMS, Broker, FlexBe

**Objective:** Validation of FlexBe ability to receive task assignments from MPMS

**Notes:** The payload of the message has predefined values.

**Precondition:**

1. MPMS is started and "Dummy Task Assignment" workflow is loaded
2. FlexBE is started
3. MPMS and FlexBE are registered with the Broker (as in TC-EP-001)

**Steps:**

1. MPMS sends a task assignment over the Broker
2. FlexBE sends a receipt acknowledgment message to MPMS
3. The ACK message is processed by MPMS

**Postcondition (expected results):**

1. Orchestration, monitoring and planning support to enable flexible and effective production
2. Augmented Reality assistance on production tools assembly
1 The payload message with the assignment is displayed on the Broker console
2 The receipt of the assignment message is displayed on Flexbe state machine
3 The ACK message is displayed on the Broker console
4 The ACK message is displayed on MPMS

Related requirements: FRQ-31, FRQ-32, FRQ-33 (as per [29])
TC-EP-007
Short name: Task assignment - advanced
Key components: MPMS, Broker, FlexBe
Objective: Validation of entire workflow of assigning tasks to FlexBe
Notes: 
Precondition:

1 Start MPMS and load the test workflow
2 Start FlexBE and load the test behavior
3 Register MPMS and FlexBE with the Broker (as in TC-EP-001)

Steps:
1 The actor starts the MPMS workflow via the MPMS UI
2 MPMS retrieves the PO details (TC-EP-005)
3 MPMS sends a task assignment over the Broker
4 FlexBE sends a receipt acknowledgment message to MPMS
5 FlexBe retrieves the Task ID from the message
6 FlexBe retrieves the Task details for this ID (TC-EP-106)

Postcondition (expected results):
1 The payload message with the assignment is displayed on the Broker console
2 The receipt of the assignment message is displayed on Flexbe state machine
3 The ACK message is displayed on the Broker console
4 The ACK message is displayed on the MPMS console
5 The Task details are displayed on FlexBe console

Related requirements: FRQ-32, FRQ-33, NRQ-12 (as per [29])

Appendix D

Below are the definitions of the validation quality attributes:

- Functional suitability
  - Functional completeness: Concept completeness – Degree to which the system functionalities cover the functionalities that the user expects
  - Functional appropriateness: Functional adequacy – Degree to which the system functions are appropriate for the task accomplishment

- Usability
  - Appropriateness recognizability: Completeness of description – Degree to which the user understands the system and the system messages
  - Operability: Operational Consistency – Degree to which the user finds the system functions consistent
  - User interface aesthetics: Attractive interaction – Degree to which the user finds it comfortable and pleasant to interact with the system

- Effectiveness: Task accuracy – Degree to which the accuracy and completeness of the automated task meets the user needs

- Efficiency: Task performance – Degree to which the task is performing or functioning in the best possible manner with the least waste of time and effort by using the component

- Satisfaction:

- Usefulness:
  - Usefulness – Degree to which the user finds the system useful

- Task satisfaction – Degree to which the task is executed effectively and efficiently

- Overall Satisfaction – Overall satisfaction of the user

The questions related to each of these attributes are the following:

- Concept completeness:
  - 1a. The functionalities of the component cover my expectations for this task.
  - 1b. Please refer the functionalities that do not totally cover your expectations.

- Functional adequacy:
  - 2a. The functionalities of the component seem to be adequate for the accomplishment of this task.
  - 2b. Please refer the functionalities that seem not to be adequate for the accomplishment of this task.

- Concept completeness
  - The functionalities of the component facilitate the accomplishment of this task.

- Completeness of description:
  - 4. It is easy to understand the functionalities and recognize if the component is appropriate for my needs.
  - 5. The system messages, alerts and indications accompanied with the component are easy to be understood.

- Operational Consistency:
  - 6. The component operation and control are consistent, without presenting unexpected and inconsistent messages.

- Attractive interaction:
  - 7. The user interface (i.e., how information is presented and how input is given, like buttons, screens etc.) of the component is pleasant and satisfying for me.

- Task accuracy:
  - 8. I am satisfied with the accuracy and completeness of task accomplishment by using the component.

- Task performance:
  - 9. I am satisfied with the amount of time spent to complete this task by using the component.
  - 10. The procedural steps are performed in an understandable and straightforward way, without the need for more user intervention and effort.

- Usefulness:
  - 11. It is easy to use the component and complete this task.
  - 12. I feel comfortable to complete my work by using the component.

- Task satisfaction: Summarizes all questions for each component (the average value of questions 1a, 2a, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12).

- Overall satisfaction: Summarizes all questions for all components together

Appendix E

The following diagrams present the evaluation scores of the MPMS component, as assessed by the initial three pilots (BOS, TRI, OPSA) of the HORSE project.

See also Figs. E34, E35, E36
**Appendix F**

Table F5 shows per pilot (numbered from Table B4) the deployed HORSE components, listed both in their logical software form and in their technical software realization. The logical components belong to either aggregation level 3 or level 4 of the architecture, depending on their software realization grouping.

**Table F5**

<table>
<thead>
<tr>
<th>HORSE Components</th>
<th>HORSE Pilots and Open Call Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P1</td>
</tr>
<tr>
<td>Process/Agent Design</td>
<td></td>
</tr>
<tr>
<td>MPMS</td>
<td>X</td>
</tr>
<tr>
<td>Safety Designer</td>
<td>X</td>
</tr>
<tr>
<td>MPMS</td>
<td>X</td>
</tr>
<tr>
<td>Global Execution</td>
<td></td>
</tr>
<tr>
<td>Intervention Manager</td>
<td>X</td>
</tr>
<tr>
<td>Shift Manager</td>
<td></td>
</tr>
<tr>
<td>Global Awareness</td>
<td></td>
</tr>
<tr>
<td>Worker Monitoring System</td>
<td>X</td>
</tr>
<tr>
<td>UWB Asset Localization</td>
<td></td>
</tr>
<tr>
<td>WebSocket Message Bus</td>
<td>X</td>
</tr>
<tr>
<td>HORSE-Kafka bridge</td>
<td>X</td>
</tr>
<tr>
<td>Abstraction layers</td>
<td></td>
</tr>
<tr>
<td>Node-RED bridge</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL interface</td>
<td></td>
</tr>
<tr>
<td>EtherCAT interface</td>
<td></td>
</tr>
</tbody>
</table>

(continued on next page)
### References


### Table F5 (continued)

<table>
<thead>
<tr>
<th>HORSE Components</th>
<th>HORSE Pilots and Open Call Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Component</td>
<td>P1 P2 P3 E1 E2 E3 E4 E5 E6 E7</td>
</tr>
<tr>
<td><strong>Task/Step Design</strong></td>
<td>FlexBe</td>
</tr>
<tr>
<td>Safety Configurator</td>
<td>X X X X</td>
</tr>
<tr>
<td>Programming by demonstration with hybrid force-control</td>
<td></td>
</tr>
<tr>
<td>Programming by demonstration for coating applications</td>
<td>X</td>
</tr>
<tr>
<td>OPC-UA</td>
<td>X</td>
</tr>
<tr>
<td>ROS Messaging</td>
<td>X X X X X</td>
</tr>
<tr>
<td>ROS MoveIT</td>
<td>X</td>
</tr>
<tr>
<td><strong>Local Execution</strong></td>
<td>ROS Kinect</td>
</tr>
<tr>
<td>nVidia CUDA</td>
<td>X</td>
</tr>
<tr>
<td>KUKA-Sunrise</td>
<td>X</td>
</tr>
<tr>
<td>BOS-adapter</td>
<td>X</td>
</tr>
<tr>
<td><strong>Local Exec.</strong></td>
<td>Drag&amp;Bot</td>
</tr>
<tr>
<td><strong>Local Perception</strong></td>
<td>Human Tracking</td>
</tr>
<tr>
<td>Device Manager</td>
<td>X</td>
</tr>
<tr>
<td>Worker sensing</td>
<td>X</td>
</tr>
<tr>
<td><strong>Sensing Supervisor</strong></td>
<td>Laser-based quality inspection</td>
</tr>
<tr>
<td>Recognition &amp; Localisation based on 3D CAD models</td>
<td>X</td>
</tr>
<tr>
<td><strong>Local Safety Guard</strong></td>
<td>Each robot has its integrated module</td>
</tr>
<tr>
<td><strong>Deviotion Monitor</strong></td>
<td>Deviation Monitor</td>
</tr>
<tr>
<td>Compliant Control</td>
<td>X</td>
</tr>
<tr>
<td>AR for task instructions</td>
<td>X</td>
</tr>
<tr>
<td><strong>Augmented Reality</strong></td>
<td>AR for quality control</td>
</tr>
<tr>
<td>Quality visual check</td>
<td>X</td>
</tr>
<tr>
<td>Ligh Guide System</td>
<td>X</td>
</tr>
</tbody>
</table>