European Commission

PERFoRM

Production harmonizEd Reconfiguration of Flexible Robots and Machinery

Horizon 2020 – Factories of the Future, Project ID: 680435

Deliverable 2.2

Definition of the System Architecture

Lead Author: IPB

Version: 1.0
Date: 28.09.2016
Status: Public

Revision	Date	Comments
0.1	23.03.2016	Deliverable structure, ToC
0.2	31/05/2016	Compilation and synthesis of requirements and previous successful projects in distributed control systems identified in WP1 deliverables, and preliminary description of the main principles of the system architecture
0.3	31/07/2016	Draft document considering the contributions from the different partners for the several chapters, the extension of the chapter related to mapping the generic system architecture for the 4 use cases, the revision of text, and the introduction of the list of acronyms
0.4	15/09/16	Improvement of the document considering the contributions from the different partners for the several chapters.
1.0	28/09/2016	Final revision of the document, harmonization of the text and removal of typos.

Author List:

Ambra Cala (Siemens)
André Hennecke (SmartFactory)
Arnaldo Pereira (IPB)
Benjamin Lee (SIEMENS)
Filippo Boschi (POLIMI)
Giacomo Angione (LOC)
Gregorio Iuzzolino (IFEVS)
Jeffrey Wermann (HSEL)
Johan Vallhagen (GKN)
José Barbosa (IPB)
Lennart Büth (TUBS)
Matthias Foehr (SIEMENS)
Olha Meyer (FhG-IPA)
Paola Fantini (POLIMI)
Paulo Leitão (IPB)
Pierluigi Petrali (WHIRLPOOL)
Ricardo Peres (UNINOVA)

Abstract

The world is transitioning to the fourth industrial revolution, with several domains of science and technology being strongly developed and, specially, being integrated with each other, allowing to build evolvable complex systems. Data digitization, Big Data analysis, distributed control, Industrial Internet of Things, Cyber-Physical Systems and self-organization, amongst others, are playing an important role in this journey.

This document considers the best practices from previous successful European projects addressing distributed control systems to design an innovative system architecture for the seamless production system reconfiguration that can be industrially deployed. For this purpose, the design process has considered the requirements and functionalities from various use cases and the best results from previous European projects.

The designed system architecture was mapped into four industrial use cases, which cover a wide spectrum of the European industrial force, and was also validated by analysing its compliance according to the use case requirements, as well as by analysing its alignment with the Industrie 4.0 principles.

This deliverable document provides input information for tasks T2.3, T2.4 and for WP3-10.

Table of Contents

List of Figures

List of Tables

Acronyms

Abbreviation	Explanation
AI	Artificial Intelligence
AM	Additive Manufacturing
ARUM	Adaptive Production Management
ASR	Architecturally Significant Requirements
ATAM	Architecture Trade-Off Analysis Method
B2MML	Business to Manufacturing Markup Language
CAD	Computer-Aided Design
CassaMobile	Flexible Mini-Factory for local and customized production in a container
CNC	Computerized Numerical Control
CPS	Cyber-Physical System
DB	Database
DCS	Distributed Control System
EMC^2-Factory	Eco Manufactured transportation means from Clean and Competitive Factory
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
FLEXA	advanced FLEXible Automation cell
FRAME	Fast Ramp-up and Adaptive Manufacturing Environment
GRACE	InteGration of pRocess and quAlity Control using multi-agEnt technology
HMI	Human-Machine Interface
HW	Hardware
ICT	Information and Communication Technology
IDEAS	Instantly Deployable Evolvable Assembly Systems
IIoT	Industrial Internet of Things
IMC-AESOP	ArchitecturE for Service-Oriented Process - Monitoring and Control
IoT	Internet of Things
IT	Information Technology
I-RAMP^3	Intelligent Reconfigurable Machines for smart Plug & Produce Production
KBF	Key Business Factors
KPI	Key Performance Indicator
M2M	Machine-to-Machine
MANUCLOUD	Distributed Cloud product specification and supply chain manufacturing execution infrastructure
MAS	Multi-Agent System
MES	Manufacturing Execution System
MPFQ	Materials, production Processes, product Functions, and product Quality
MRP	Material Requirement Planning
MTBF	Mean Time Between Failures
MTTR	Mean Time to Recover
OEE	Order Equipment Efficiency

OPC-UA	OPC Unified Architecture
OT	Operational Technology
PERFoRM	Production harmonizEd Reconfiguration of Flexible Robots and Machinery
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PRIME	Plug and produce intelligent multi-agent environment based on standard technology
R&D	Research and development
RAMI	Reference Architecture Model Industry 4.0
RE	Requirements Engineering
ReBORN	Innovative Reuse of modular knowledge Based devices and technologies for Old, Renewed and New factories
RT	Real-Time
SAAM	Software Analyses Architecture Method
SCADA	Supervisory Control and Data Acquisition
SE	Simulation Environment
Self-Learning	Reliable Self-Learning Production Systems Based on Context Aware Services
SelSus	Health Monitoring and Life-Long Capability Management for SELf-SUStaining Manufacturing Systems
SME	Small and Medium-sized Enterprises
SOA	Service Oriented Architecture
SOCRADES	Service-Oriented Cross-Layer Infrastructure for Smart Embedded Devices
SPC	Statistical Process Control
SW	Software
WP	Work Package
WS	Web Service
XML	Extensible Markup Language

1. Introduction

1.1 Contextualization

Globalization allows consumers to have easy access to producers around the world, being irrelevant where producers and consumers are located. Naturally, alongside with this market globalization, consumers have become more demanding in terms of product customization, quality and cost, forcing companies from different parts of the globe to compete without decreasing the product's quality and, many times, decreasing the production costs. In order to face this world-market re-shape, manufacturing companies demand new manufacturing paradigms, techniques and technologies, as well as new business models, for a complete process integration, sustained mainly by a production digitization, massive information exchange and processing [1].

To respond to these industry demands, many national and transnational programs, each one having its own research and innovation strategy, have emerged to support the research on key scientific and technological areas. The Industrie 4.0 initiative was established in Germany [2], and later on other European countries followed this vision by promoting similar initiatives, e.g., "Made in Sweden 2030" in Sweden, "La Nouvelle France Industrielle" in France, "Smart Industry" in Netherlands, "Industria Conectada 4.0" in Spain, "Innovate UK" in United Kingdom, "Fabbrica Intelligente" in Italy and "Industria 4.0" in Portugal. The European Commission, together with the private sector, has launched the Horizon 2020 program that includes several Public-Private Partnerships covering areas related to the manufacturing domain, namely Factories of the Future, Big Data Value and Robotics. Other countries from non-European continents are also providing similar roadmaps and funding schemes, particularly the "Industrial Internet" consortium in the United States [3] and the "Made in China 2025" program in China.

These initiatives and roadmaps are based on the adoption of similar technological strategies. Digitization is one of the cornerstones in future systems where the goal is the processing of the large amount of data collected from the system assets, namely resources, tools and supply-chain. This is being supported by using some disruptive technologies, namely the Industrial Internet of Things (IIoT) to collect massive sensorial data, Machine-to-Machine (M2M) protocols to support the interconnection between machines, Big Data analysis techniques for the data processing and cloud computing for storing huge amount of data and running powerful advanced data analysis algorithms.

Resources, while becoming smarter and pluggable, are able to communicate more effectively with each other, shifting the way these systems are designed, converting the traditional monolithic hierarchical systems into a distributed and horizontal structure, where the diverse components are cooperating and collaborating with each other. Commonly, the aforementioned features are wrapped under a common paradigm, designated as Cyber-Physical Systems (CPS) [4, 5] merging the physical part (i.e., the real world) with the logical part (i.e., the cyber counterpart). Several technological solutions are being advocated as promising to implement CPS solutions and to some extent being already applied. Multi-Agent Systems (MAS) [6, 7] are being used to provide distributed intelligence to the system's components while the Service Oriented Architecture (SoA) principles provide pluggability and seamless vertical and horizontal system integration. From a semantic point-of-view, the use of ontologies to define common data structures also supports a seamless integration.

The software applications located at higher level of the ISA95 automation pyramid [51], e.g., at MES (Manufacturing Execution System) and ERP (Enterprise Resource Planning) levels, must also be developed and improved in order to fully exploit this innovation. These tools must now be "connected" in the way where a true information exchange with other, not foreseen in classical approaches, must now be considered. This new consideration impose that the ISA95 pyramid is now dismantled and converted into an ecosystem of interoperable modules/tools following a cloud-based CPS approach.

In this context, and aligned with the technological trends and best industrial practices, the PERFoRM project aims to develop an innovative approach to handle the seamless production system reconfiguration, combining the plug-and-produce concept and the human role as a flexibility driver in future production systems. The proposed system also integrates advanced tools to enable the system operationality, namely scheduling, modelling, simulation and intelligent decision support. These concepts are aggregated by using a system-wide language that is compliant with legacy systems by using proper adapters. Having this in mind, this document, based on the best results from previous successful R&D projects in the field, aims to describe the main pillars of the PERFoRM system architecture.

1.2 Objective of the document

A flexible manufacturing system is a highly automated system consisting of processing stations, material handling and storage systems and controlled by a distributed computer system. As described in [14], and illustrated in Figure 1, the flexible and reconfigurable manufacturing concept can be characterized in three contexts: asset, process and architectural views.

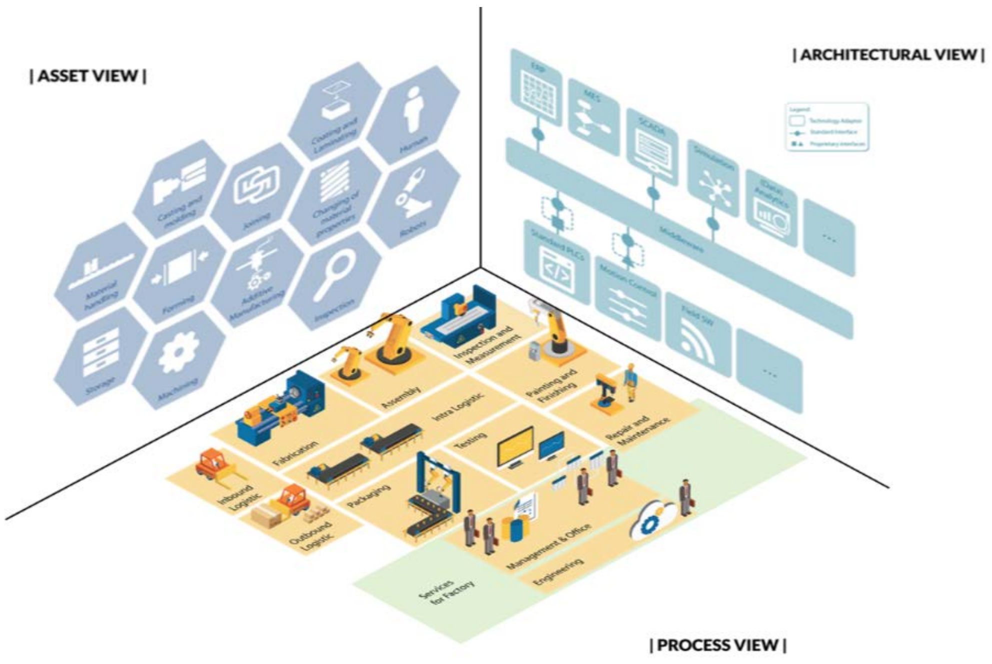


Figure 1 – Multi-views of the flexible and reconfigurable manufacturing system

The asset view represents the physical equipment of a flexible manufacturing system, which involves not only the workstations, e.g., machine operations, inspection facilities, assembly stations and material handling systems, but also the human resources. The process view depicts the shop floor and the office space and comprehends the several processes in a factory, namely inbound logistic, processing, assembling, inspection and maintenance. The architectural view represents the software provisions of a flexible manufacturing system, comprising the several IT systems and information and communication technologies (ICT) that exist in the factory or plant floor.

This deliverable contains the outcome of Task 2.2, entitled “System architecture design”, which addresses the architectural view and has as the main objective to design a system architecture based on smart production components, compliant with Industrie 4.0 principles, which is able to support the seamless system reconfiguration and enhance planning, simulation and operational features. Therefore, this document contains a sustained system design, its components, their functions and possible interactions between them.

The main assumption is not to develop the system architecture from scratch but instead to consider the results from previous successful R&D projects in the field of distributed automation control systems. For this purpose, as illustrated in Figure 2, this task considers the requirements defined in WP1.2 [8], WP2.1 [9], WP7.1 [10], WP8.1 [11], WP9.1 [12] and WP10.1 [13], and the information regarding previous approaches and technologies for distributed control systems identified and collected in WP1.1 [14] as inputs to address the established main objective.

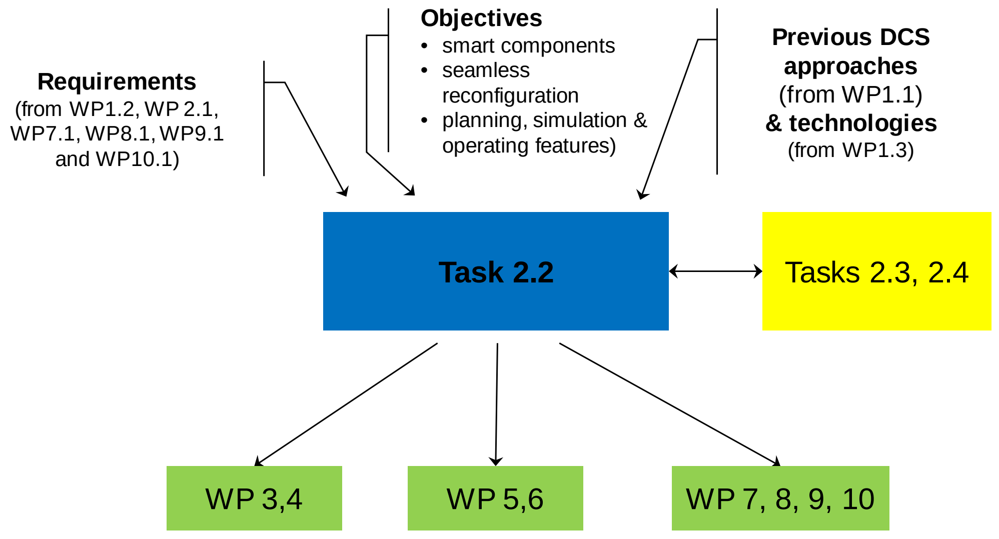


Figure 2 - Interconnection of the Task 2.2 with other WPs

The results of this task will be used in different WPs according to different granularity levels:

- The further specification and development of some architectural elements will be performed in T2.3 (namely data models and standard interfaces), T2.4 (namely the industrial middleware), WP3 (technological adapters and real-time information processing) and WP4 (advanced tools and mechanisms to support seamless reconfiguration, visualization, planning and simulation of operating processes).
- The integration of developed mechanisms and tools, as well the establishment of migration strategies will be performed in WP5.
- A pre-demonstration of the architecture system implementation will be performed in WP6.
- The demonstration at the use cases will be performed in WP7, WP8, WP9 and WP10.

The PERFoRM system architecture will cover the different layers of the automation pyramid defined by ISA-95, and particularly the layers L2 (Supervisory Control) and L3 (Manufacturing Operations Management), by providing an infra-structure and methodology to deploy the new generation of automation systems in the form of distributed cloud-based automation systems, following the principles sustained by Industrie 4.0 and the factories of the future. In fact, the PERFoRM system architecture should provide the means for both the vertical and horizontal system integration.

1.3 Structure of the document

The document is divided into 9 chapters. After this brief introduction, Chapter 2 summarizes the list of requirements established for the design of the distributed system architecture, taking into consideration the outcomes from WP1.2, WP2.1, WP7.1, WP8.1, WP9.1 and WP10.1. Chapter 3 overviews the related work in distributed control systems and particularly the existing successful approaches previously developed under R&D projects funded by EU, which best results can be re-used to design the PERFoRM system architecture. Chapter 4 is dedicated to describe the main principles of the system architecture addressing the identified requirements and aiming to achieve the seamless production system reconfiguration, and Chapter 5 details the functionalities of the architectural elements. Chapter 6 describes the mapping the generic system architecture into four industrial use cases, namely a large compressor producer, a micro-electrical vehicles producer, a home appliances producer and an aerospace components producer. Chapter 7 describes the preliminary validation of the designed system architecture and the Chapter 8 presents the compliance of the PERFoRM system architecture with the Industrie 4.0 platform and other similar initiatives associated to the factories of the future. Finally, Chapter 9 summarizes the conclusions of the document.

2. Analysis of Collected Use Cases Requirements

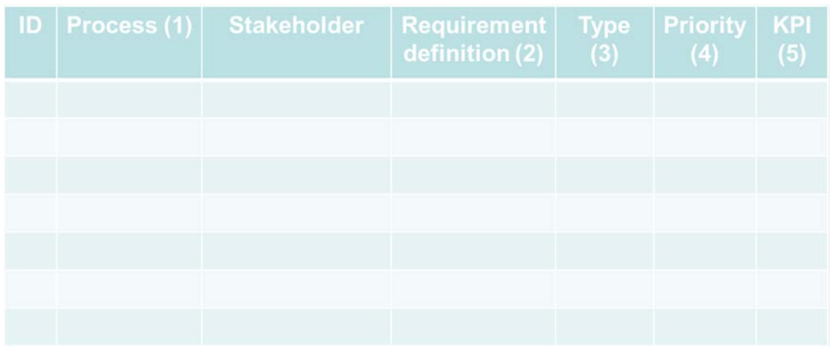
The validation of the PERFoRM system will be accomplished in 4 uses cases, covering a wide spectrum of the European industrial force, and ranging from home appliances to aerospace and from micro electrical vehicles to large compressor production. Several requirements were collected from the use cases in previous tasks of the PERFoRM, namely in WP1.2 [8], WP2.1 [9], WP7.1 [10], WP8.1 [11], WP9.1 [12] and WP10.1 [13], and are compiled in Table 5.

A deep analysis of the identified requirements has been carried out in deliverable D1.2 [8] in order to understand which are the main needs manufacturing industry has to face with when trying to achieve a flexible manufacturing environment based on rapid and seamless reconfiguration of machinery and robots. This find its relevance in ensuring the adequate level of flexibility and reconfigurability needed to introduce the new production concept based on plug and produce production systems and self-adjusting devices.

It has been worked out from D1.2, that a requirement describes a condition or capability of a system needed by a user to solve a problem or fulfill a specification and its documentation. Therefore, the identification of what is the problem (and not how it will be solved) and its documentation in an understandable form for all the stakeholders represent the first steps to be followed before realize a general and coherent analysis. For these reasons, a proper methodology based on RE (Requirements Engineering) has been carried out through an iterative exercise in order to first identify the problem, then specify the requirements and finally validate them. In particular, this methodology essentially consists in four main steps that allow to discover all the potential requirements (Elicitation phase), guarantee their quality level according to the solution (Analysis phase), notify them in a clear and understandable form for each stakeholder (Specification phase) and, finally, validate them in a second round (Validation phase).

The proposed methodology contributed to have a straight collaboration with each use case partner confirming that this approach could be indifferently used among different manufacturing sectors, for large companies, SMEs as well as for new and existing plants and, therefore, showing its wide applicability. The collected results come from the fulfillment of the following table that led to realize a focused analysis on the specific processes involved in the project, to identify the specific person, organization and department that have a potential interest on each process, to understand the problem that arises from the needs of stakeholders and, therefore, to sort out the final requirements and their performance indicators.

Furthermore, this methodology allows to classify on a relevance scale each requirement (through Priority taxonomy), identifying which requirements are really important according to the specific use case perspective.



ID	Process (1)	Stakeholder	Requirement definition (2)	Type (3)	Priority (4)	KPI (5)

Figure 3 – Second iteration template

Through this table is possible to figure out the relevant requirements that should be taken into account in a flexible and reconfigurable production system, clustering them in two classes: general requirements, which are necessary for a successful implementation of reconfigurable and flexible systems, and specific requirements, which represent the specific needs of each use case.

These results are depicted in Figure 4, where the summary of the overall requirements collected for each Use Case are figured out. In particular, the same framework utilized to represent the Use Case requirements has been provided in order to underline the same logic approach needed to understand it. In fact, coherently with D1.2, also in this case a bottom-up approach has been proposed aiming at showing that each level of requirements is led to reach the main relevant aspect of both flexibility and reconfigurability.

In these terms, considering the flexibility aspect, it is possible to realize, for example, that *the reduction* of both *cycle time* and *cycle cost* including the ability to *move different production units* allows to achieve the capability to *change raw materia*ls and *processes* and to obtain a *full process interaction*, leading to reach *agile production* and, therefore; leading to have the required flexibility level.

Applying the same approach to reconfigurability, it is possible to figure out that the awareness on production processes in terms of quality and health machine status is given by the ability to obtain feedbacks from robot system configuration, from production and from processes based on failure control which permit to have the *self-configuring system ability*, hence ensuring the *reduction of set-up time* and *reconfiguration cost*.

Finally, the last part of this analysis concerns the second class of requirements that are not directly involved in the flexibility or reconfigurability aspects but, although this, are considered necessary to reach them. In particular, Figure 4 shows, for example, that the full *traceability and identification of each product*, the ability *to simulate the global behavior in CPS environmen*t and the *integration and quick communication among different departments* are considered as the appropriate foundations to achieve flexibility and reconfigurability.

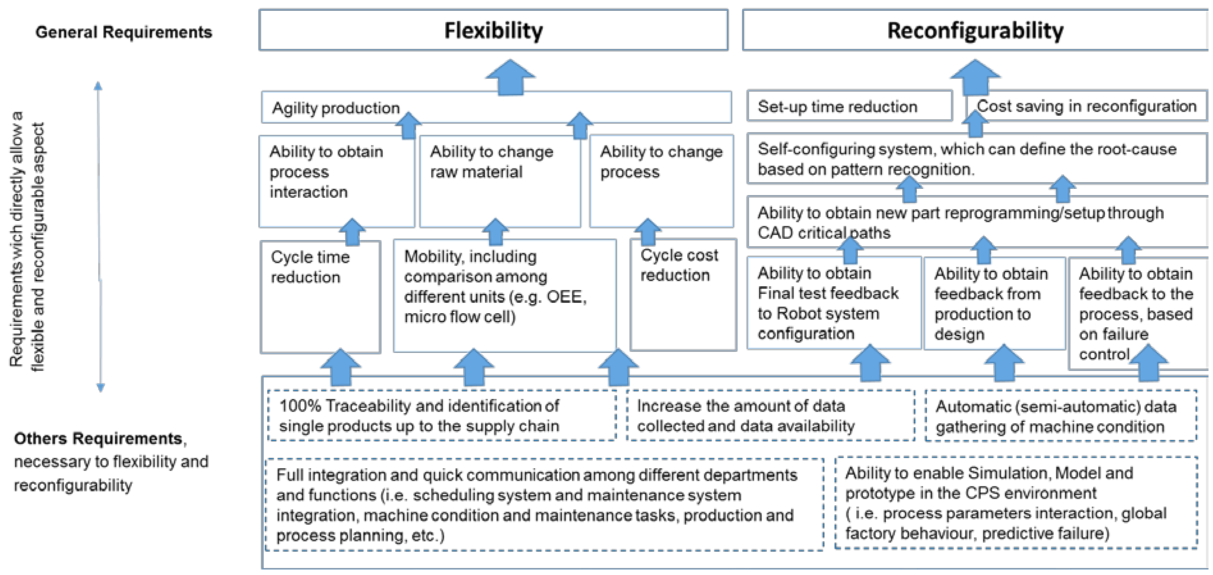

General Requirements
Flexibility
Reconfigurability
Requirements wich directly allow a flexible and reconfigurable aspect
Agility production
Ability to obtain process interaction
Ability to change raw material
Ability to change process
Cycle time reduction
Mobility, including comparison among different units (e.g. OEE, micro flow cell)
Cycle cost reduction
Set-up time reduction
Cost saving in reconfiguration
Self-configuring system, which can define the root-cause based on pattern recognition.
Ability to obtain new part reprogramming/setup through CAD critical paths
Ability to obtain Final test feedback to Robot system configuration
Ability to obtain feedback from production to design
Ability to obtain feedback to the process, based on failure control
Others Requirements, necessary to flexibility and reconfigurability
100% Traceability and identification of single products up to the supply chain
Increase the amount of data collected and data availability
Automatic (semi-automatic) data gathering of machine condition
Full integration and quick communication among different departments and functions (i.e. scheduling system and maintenance system integration, machine condition and maintenance tasks, production and process planning, etc.)
Ability to enable Simulation, Model and prototype in the CPS environment (i.e. process parameters interaction, global factory behaviour, predictive failure)

Figure 4 - General requirements: flexibility and reconfigurability overview

The outcome of this analysis provides the basis to develop the infrastructure able to support the correlation among different production components and their mutual communication guaranteeing the correct aspects of machinery and control systems plugability. In fact, providing the overall framework of functional requirements, it will possible to provide the guidelines for transforming existing production system into the new CPS environments based on the architecture aiming at ensuring the connectivity and the interoperability needed to support smart production components and their seamless reconfiguration capabilities.

3. Previous R&D Projects Focusing the New Generation of Automation Systems

The Industrie 4.0 platform [2] provides a vision to modernize the manufacturing sector towards the smart factories of the future addressing the current requirements of product customization, quality and cost, as well as the rapid and flexible reaction of manufacturing processes to condition changes in terms of product variability and fluctuation and process disturbances. In this perspective, consolidated along the last decade, the demanded new generation of automation control systems require new emergent and challenging features that are not anymore provided by the traditional hierarchical and monolithic ISA-95 automation systems. In fact, in this new distributed approach, as illustrated in Figure 5, the functionalities provided at the different ISA-95 levels [51] can be exposed as a collection of CPS services, which may exist in the traditional systems as well as the cloud, giving rise to a highly heterogeneous, dynamic, and adequately performant ecosystem of services [15].

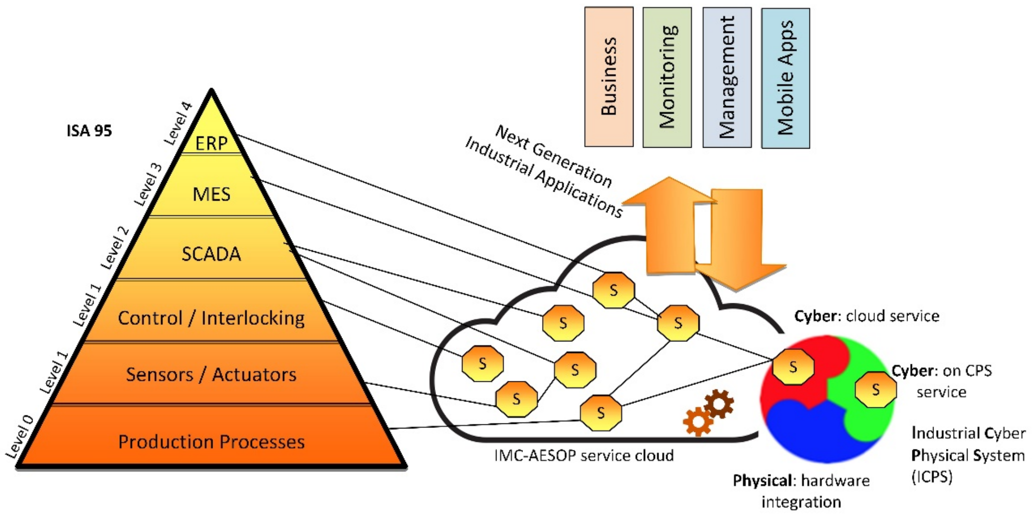


Figure 5 - Moving from centralized ISA-95 automation systems to the distributed cloud-based automation systems [16].

Aligned with this transformation from the centralized automation systems architecture into a distributed automation control system architecture, over the past few years, the EU FP7 programme supported several successful projects in the area of agile and plug-and-produce manufacturing, which have contributed to improve the state-of-the-art in the field. In spite of having created a sound conceptual basis, methods and technologies to achieve the true industrialisation, none of these projects was individually able to achieve sufficient technical maturity and critical mass to allow large scale industrial uptake of the agile, plug-and-produce system concept.

The IMC-AESOP (ArchitecturE for Service-Oriented Process - Monitoring and Control) project [17] envisioned a SOA-based SCADA (Supervisory Control and Data Acquisition)/DCS (Distributed Control System) infrastructure enabling the cross-layer service-oriented collaboration, i.e., not only at the horizontal level, e.g., among cooperating devices and systems but also at vertical level between systems located at different levels of an enterprise architecture. The SOCRADES (Service-oriented cross-layer infrastructure for distributed smart embedded systems) project [18] developed a design, execution and management platform for next-generation industrial automation systems, exploiting the SOA paradigm both at the device and at the application level.

Several of the projects present some kind of self-* features. In fact, under the GRACE (InteGration of pRocess and quAlity Control using multi-agEnt technology) [19] project, a MAS solution was implemented and deployed into a real industrial washing machine production line, integrating process control with quality control at local and global level, and featuring self-adaptation and self-optimization mechanisms. The MAS approach was also used by the IDEAS (Instantly Deployable Evolvable Assembly Systems) project [20] to create a fully distributed and pluggable mechatronic environment capable of self-configuring, self-organize and self-diagnose, allowing the dynamic control. The SelSus (Health Monitoring and Life-Long Capability Management for SELf-SUStaining Manufacturing Systems) project [22] proposes a new paradigm for highly effective, self-healing production resources and systems to maximize their performance over longer lifetimes through highly targeted and timely repair, renovation and upgrading. The Self-Learning (Reliable Self-Learning production systems based on context-aware service) project [23] was focused in developing highly reliable and secure service-based self-learning production systems aiming at merging the world of secondary processes (e.g., maintenance, energy efficiency, scheduling) with the world of control by using context awareness and data mining techniques. Finally, the PRIME (Plug and produce intelligent multi-agent environment based on standard technology) project [24] has developed a multi-agent architecture using plug-and-produce principles for configuring production systems through innovative human-machine interaction (HMI) mechanisms. PRIME developed a pluggable, high-scalable and distributed framework for the creation of new solutions regarding adaptive, self-aware, self-monitored and reconfigurable plug-and-produce systems.

Self-aware intelligent components are considered in the ReBORN (Innovative Reuse of modular knowledge Based devices and technologies for Old, Renewed and New factories) project [21], which aims to demonstrate strategies and technologies that support a new paradigm for the re-use of existing production equipment in factories. The ReBorn project is predominantly focusing on the development of hardware (e.g., modular reconfigurable assembly equipment, electric presses, and highly flexible resistance welding cells), but also touches the field of MES, e.g., software tools to retrieve and process maintenance relevant equipment status information.

The ARUM (Adaptive Production Management) project [25] aimed to improve planning and control systems for complex, small-lot products manufacturing, such as aircrafts and ships. The ARUM's approach is based on a new generation of service-oriented enterprise information platforms, a service bus integrating service-based architecture and knowledge-based multi-agent systems. The core objective of FRAME (Fast Ramp-up and Adaptive Manufacturing Environment) project [26] is the development of new control and human machine interaction strategies to enable future assembly systems to become self-aware, self-learning, and ultimately self-adapting during ramp-up and proactively react to disruptive events.

FLEXA (advanced FLEXible Automation cell) [27] was a case-study driven project whose main objective was to create the tools, methods, and technologies needed to define, prepare and validate an automated flexible cell that can manufacture a generic process chain allowing for safe human interaction and deliver quality assured parts for the European aerospace industry.

The MANUCLOUD (Distributed Cloud product specification and supply chain manufacturing execution infrastructure) project [28] aimed the development of a service-oriented IT environment as a basis for the next level of manufacturing networks by enabling production-related inter-enterprise integration down to shop floor level. For this purpose, self-descriptions for equipment, process and factory level manufacturing services and related mapping mechanisms were developed. The vision sustained in the I-RAMP^3 (Intelligent Reconfigurable Machines for smart Plug & Produce Production) project [29] is to enable the European manufacturing industry towards smart manufacturing systems in conventional production.

The EMC^2-Factory (Eco Manufactured transportation means from Clean and Competitive Factory) project [30] addressed the development of a radically new paradigm for cost-effective, highly productive, energy-efficient and sustainable factories. In particular, the project focused on main energy-intensive processes within three industrial sectors in Europe (automotive, rail and aerospace), developing tangible and industry relevant results to be easily implemented in manufacturing environments.

The main goal of the CassaMobile (Flexible Mini-Factory for local and customized production in a container) project [31] is to develop a new kind of local, flexible and environmentally friendly production system for highly customized parts based on a combination of different manufacturing processes like 3D-printing, CNC-milling and 3D-assembly technologies up to cleaning in clean room environment inside an enclosed unit such as a container.

Table 1 summarizes the evaluation of referred relevant projects taking into consideration some criteria regarding the used technologies and features [32].

Table 1 – Summary of Distributed Control System Approaches

Technology/ feature	GRACE	IDEAS	IMC-AESOP	SOCRADES	PRIME	ARUM	Self-Learning	FRAME	FLEXA	ReBORN	SelSus	MANUCLOUD	I-RAMP^3	EMC^2-Factory	CassaMobile
SOA			●	●		●	●		●		●	●			●
MAS	●	●			●	●									
Middleware	○	○	●	○	○	●	●	○	○		●	○	○		○
Standard Interfaces						●			●			●			
Human integration							●	●	●			●			●
Plug and play adapters					●					●			●		●

Schedulers & planners tools						●			●					●	
Self-* features	●	●		●	●		●	●		●	●	●			●
ISA-level	L2	L2	L1-4	L1-4	L2	L3-4	L2-3	L2	L2-4	L2-3	L2-3	L2-3	L3	L2-4	L2-3

Legend: ● covered; ○ partially covered; The ISA-95 levels are: Physical processes (L0), Automation Control (L1), Supervisory Control (L2), Manufacturing Operations Management (L3), and Business Planning and Logistics (L4).

Summarizing, SOA principles were embraced by several projects, namely SOCRADES, IMC-AESOP, ARUM, FLEXA, SelSus, MANUCLOUD and CassaMobile, with each one focusing on different ISA-95 layers, meaning that they can be applied successfully in industrial automation manufacturing. In a similar way, several projects used MAS as its main technological driver to achieve the decentralization of the control functions, namely GRACE, IDEAS, PRIME and ARUM projects.

Since several of these projects covered the distributed manufacturing systems, the topics of middleware, interfaces and adapters were also addressed by the majority of them. In particular, PRIME, ReBORN and I-RAMP[3] projects have focused on the pluggability by considering proper interfaces and adapters. On the other hand, the type of middleware used by the different projects span from Enterprise Service Bus (ESB), e.g., ARUM, to topologies nearest to point-to-point, e.g., FLEXA. In some approaches, e.g., GRACE, IDEAS and PRIME, an agent-based framework was used as middleware to support the interconnectivity among agents, covering only partially the middleware requirements (since they are only able to interconnect agent compliant tools).

The development and integration of high-level tools, e.g., focusing on strategic planning, scheduling and simulation, was touched by ARUM, FLEXA and EMC^2-Factory, some of them developed by using MAS technology and interconnected by using SOA principles, e.g., the agent-based planning systems developed in ARUM. In another perspective, several projects have considered self-* features, namely GRACE with self-adaptation and self-optimization, IDEAS with self-configuration and self-diagnosis, PRIME with self-monitoring, Self-Learning and FRAME with self-learning, ReBORN with self-awareness, SelSus with self-healing and finally, CassaMobile with self-description.

At the end, FLEXA, FRAME, PRIME MANUCLOUD and CassaMobile projects addressed the human integration by enabling human interaction.

4. Analysis of the PERFoRM System Architecture Principles

The design of the system architecture for new innovative production systems should take into consideration the requirements and functionalities defined for the use cases of the PERFoRM project, summarized in Chapter 2. An important assumption is to reuse the results from the previous successful R&D projects in the field, briefly described in Chapter 3, instead of developing a new architecture from scratch. This assumption also guarantees that the proposed architecture will be backward compatible and aligned with the current state-of-the-art approaches, increasing therefore its industrial adoption possibility. Furthermore, it is also crucial that the proposed architecture be aligned with the current trends, particularly with those being followed by the several international initiatives (e.g., the Industrie 4.0 platform and its Reference Architectural Model (RAMI) [49]).

The analysis of the identified requirements shows that the system architecture should:

i) Be based on smart and heterogeneous production components.

ii) Be able to support the seamless system reconfiguration.

iii) Be able to enhance planning, simulation and operational features.

iv) Be able to provide human operators with enhanced information and assistance.

Therefore, this chapter details the design of the system architecture principles taking into consideration these objectives.

4.1 Network of Smart Components

The objective of building a system architecture based on a community of smart and heterogeneous components demands the use of distributed control approaches instead of using the traditional centralized ones. In fact, these approaches are characterized to be rigid and monolithic structures that are not anymore able to face the levels of responsiveness and reconfigurability imposed by the factories of the future. For this purpose, several assumptions are established, each one being concretized using proper methods, approaches and technologies, as illustrated in Table 2.

Table 2 – Reaching Assumptions to Develop a System Based on Smart Production Components

Assumptions	How to reach
Distributed and heterogeneous HW devices and SW applications	• Use of service-oriented design principles • Aggregate and compose services (or skills) • Use of holonic design principles
Interconnectivity in an easy and transparent manner	• Use of standard interfaces • Use of adapters for legacy systems (existing interfaces) • Use of industrially adopted IoT technologies and M2M protocols
Some production components can be enriched with intelligence	• Use of artificial intelligence (AI) methods, and particularly MAS • Embed advanced data analysis
Integration of Humans in the loop	• Use HMI and mobile devices

	• Use of augmented reality technologies

The first assumption supports the idea of how to create a system based on a plethora of distributed and heterogeneous HW devices and SW applications. This can be reached by using service-oriented design principles, encapsulating their own functionalities as services, which are offered to the other components. This approach has been proved in SOCRADES, IMC-AESOP and ARUM as suitable for industrial automation. Complementary, this approach follows the IoT paradigm, being necessary to interconnect the production components in a transparent manner. In fact, in distributed, and particularly heterogeneous systems, the interconnectivity among these distributed components assume a critical role, being required to use standard interfaces (in terms of syntax and semantics) and industrially adopted M2M protocols, covering the several ISA-95 automation levels (see Figure 6).

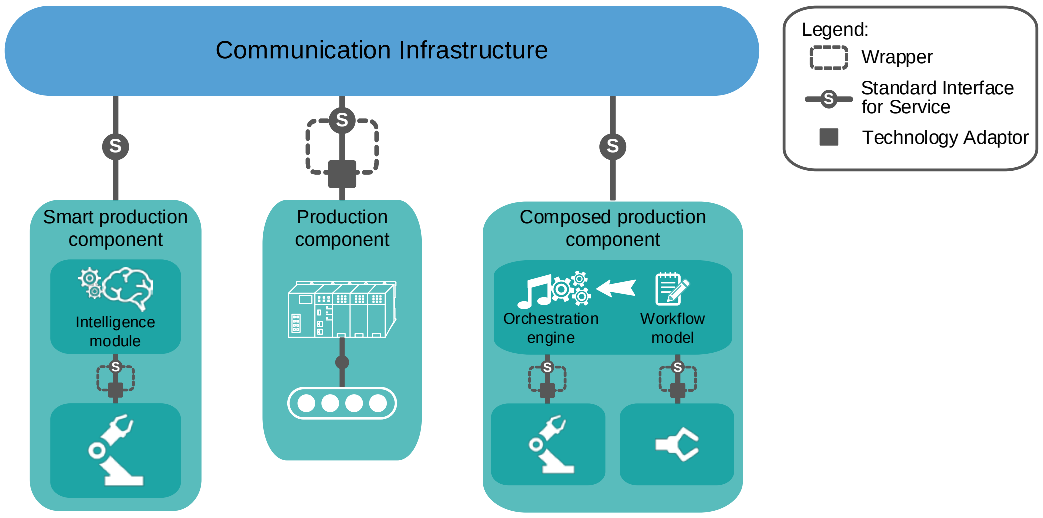

Communication Infrastructure
Legend:
Wrapper
Standard Interface for Service
Technology Adaptor
S
Smart production component
Intelligence module
Production component
Composed production component
Orchestration engine
Workflow model

Figure 6 - Distributed smart production components.

The creation of system of systems or holistic smart production components can be reached by using the aggregation and composition of atomic services (or skills). In fact, the service composition [33] is the combination of atomic services, and the interaction patterns between them, to create a composed service that is offered to the other entities. In this process, the service orchestration [34] is crucial to sequence and synchronize the execution of the atomic services according to a workflow that represents the business process, providing a high-level interface for such composed process. As an example, illustrated in Figure 6, consider an industrial robot offering the atomic service of "movement" and a gripper offering the atomic service "hold". Composing the two atomic services allows to create a new composed service "pick-and-place". The use of the service composition and orchestration was successfully tested and, thus, approved in the SOCRADES and IDEAS projects.

Also, as illustrated in Figure 6, the objective of building a network of distributed smart production components is achieved by considering the exposition of functionalities through the use of service interfaces, which should be standardized, as well the adoption of an industrial middleware (configured in a distributed structure) to support the easy interconnectivity among these components. For the cases where legacy systems exist, there's the need to convert the source native data structure into the system (in this case PERFoRM) language.

Some production components can be enriched with intelligence and self-* features to improve their behaviour during run time, e.g., embedding AI methods, and particularly MAS technology [6, 7]. In fact, MAS offers an alternative way to design these systems by distributing the automation control functions by several intelligent, autonomous and cooperative agents, providing flexibility, (since large-scale systems are built upon modular building blocks, i.e., the agents), robustness (since the system continues operating without perturbation even if one agent fails) and reconfigurability (since agents can be added, removed or even modified on-the-fly, i.e., without the need to stop, re-program and reinitialize the other agents). Advanced data analytics, for example considering data mining algorithms, can be easily integrated in production components, providing a smart layer that allows a fast response to condition changes and the identification of reconfiguration opportunities. Note that automation machines need different reaction times to data change, which requires in a certain of cases to embed local and real-time data processing aiming to filter the collected data as well as to run real-tie algorithms to process these data.

The human integration assumes an important issue as flexibility driver, which is forested by interfacing the humans with user friendly HMI, mobile devices (e.g., tablets and smartphones) and applying augmented reality technology, in order to provide relevant, up-to-date information and support the most appropriate intervention.

4.2 Seamless System Reconfiguration

The seamless system reconfiguration is a critical issue in the factories of the future, as identified in the Industrie 4.0 initiative. Table 3 summarizes how this objective can be reached in the PERFoRM system architecture.

Table 3 – Reaching Assumptions to Achieve the Seamless System Configuration

Assumptions	**How to reach**
Plug and play of production components on-the-fly	• Use of distributed approaches, e.g., MAS and SOA • Use of registry and discovery mechanisms • Use of standard interfaces • Consider a catalogue of adapters for existing interfaces (legacy systems) using self-descriptive and self-integrative concepts
On-the-fly reconfiguration	• Use of plug-and-produce concepts • Use self-* mechanisms, e.g., self-adaptation, self-organization and self-diagnosis • Consider reconfiguration boundaries and nervousness control

The seamless system reconfiguration requires the capability to add, remove or modify production components on-the-fly, i.e., without the need to stop, re-program and restart again the components. This can be reached by using distributed approaches, e.g., MAS or SOA. These approaches provide flexibility and robustness associated to their decentralized and distributed nature, in opposite to the traditional centralized control approaches, which are built up upon a central node.

Aiming to reach a truly system reconfiguration, the plug-and-produce concepts should be considered, taking insights, e.g., from the results achieved in PRIME, GRACE and I-RAMP[3]. For this purpose, the plug-and-play ability of production components can be simplified by using registry and discovery mechanisms, which are inherent to SOA approaches. The use of standard interfaces to describe these services in a transparent manner and adapters to convert the existing interfaces to the standard interfaces language, ensure the transparent interconnectivity of these components.

Self-organization mechanisms also play an important role for the system reconfiguration, namely considering the behavioural and structural perspectives, which provides different scopes and time response to evolution [35]. In this field, the reconfiguration boundaries, nervousness and chaos control should be considered, allowing to keep the system under control during plug-and-produce.

Other self-* mechanisms may be considered to improve the system behaviour, namely the self-adaptation, self-diagnosis and self-learning already testes in GRACE, IDEAS, PRIME Self-Learning, FRAME, Re-Born and SelSus, as well as self-description of system modules already tested in MANUCLOUD.

4.3 Enhancing Planning, Simulation and Operational Features

Existing legacy systems focusing planning, simulation and operational features must be integrated and also co-exist with advanced tools taking advantage of powerful computational algorithms and technologies. Table 4 summarizes how this objective can be achieved in the design of the PERFoRM system architecture.

Table 4 – Reaching Assumptions to Enhance Planning, Simulation and Operational Features

Assumptions	**How to reach**
Integrate legacy systems, such as MES, SCADA and databases	• Use of standard interfaces • Use M2M and ESB technologies addressing backbone level • Consider a catalogue of adapters for existing interfaces • GUI as human interaction enabler
Integrate advanced planning, simulation applications	• Use MAS and advanced optimization methods • Use advanced simulation frameworks • Use cloud technologies • Use of standard interfaces • GUI as human interaction enabler
Seamless data representation and exchange schema	• Consider standards for the representation of industrial data models • Use gateways for data transformation (interconnecting backbone and machinery levels) • GUI as human interaction enabler

The integration of legacy systems, such as databases, ERP (Enterprise Resource Planning), MES (Manufacturing Execution Systems) and SCADA, is simplified by using standard interfaces, and ESB platforms to implement industrial middlewares addressing the higher ISA-95 levels. Adapters are commonly used to interconnect these legacy systems by transforming their internal data models into the standard interfaces data model. Advanced planning, scheduling and simulation applications, e.g., developed using the MAS technology, may also be integrated, but in this case without the need to use adapters since they already follow the PERFoRM standard interfaces. As described, the integration of agent-based planning and scheduling systems using ESB platforms was successfully tested in ARUM. These applications, as well as advanced simulation applications, could run in cloud platforms to take advantage of ubiquity and computational power.

The seamless data representation and exchange schema is reached by considering industrially adopted data models, e.g., IEC 62264 B2MML (Business to Manufacturing Markup Language), which is a XML implementation of the ISA-95, for the backbone environment, and OPC-UA or IEC 62714 AutomationML data models for the machinery environment. The implementation of gateways that interconnect data models from backbone and machinery levels are also required to ensure a proper data transformation. Note that for a potential industrial adoption, the development of new ontologies for the PERFoRM data model should be avoided and instead the use of industrial data models should be encouraged.

5. The PERFoRM System Architecture

As result of considering the assumptions established for the initial four objectives described in the previous chapter, the system architecture for the seamless production system reconfiguration is based on a network of distributed HW devices and SW applications, addressing different ISA-95 levels, which exposes their functionalities as services following SOA principles, and are interconnected in a transparent manner by using an industrial middleware, as illustrated in Figure 7.

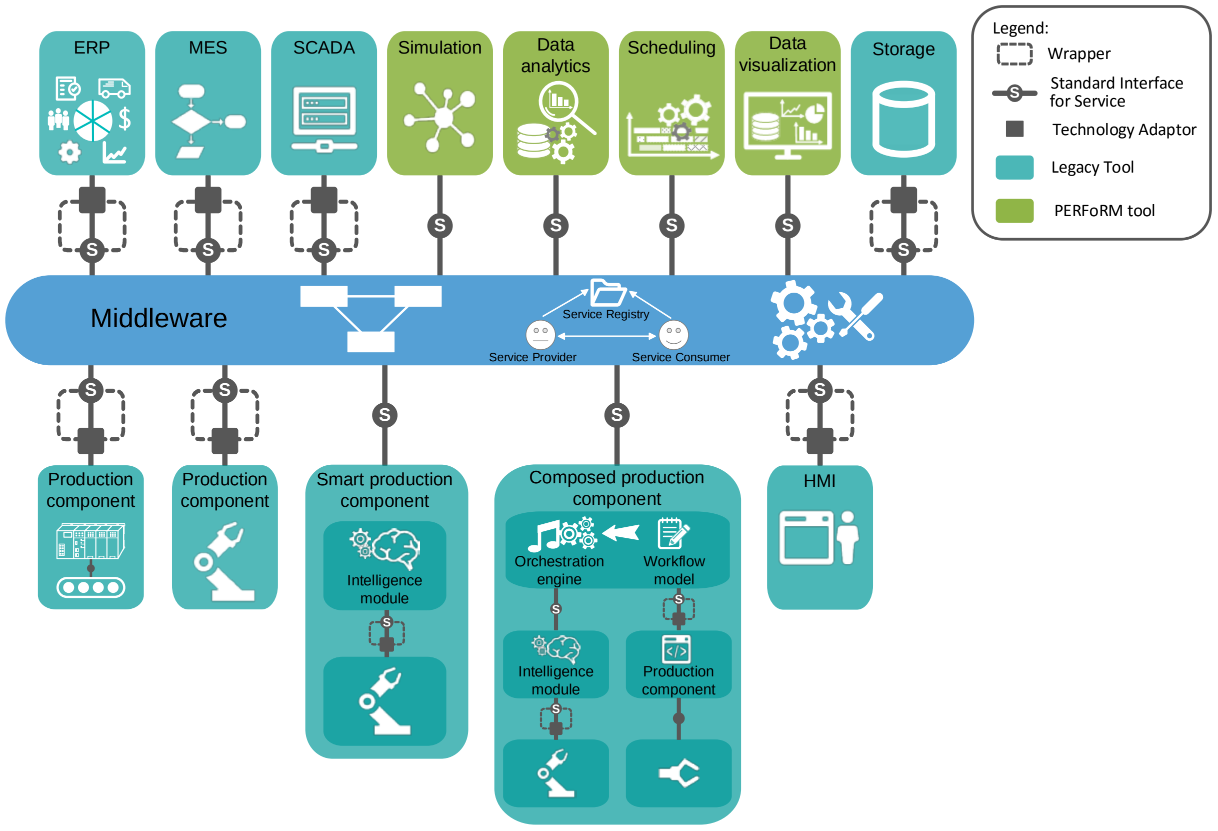


Figure 7 - Overall PERFoRM system architecture.

This chapter depicts the core architectural elements, in a more technical perspective, allowing the fulfilment of the aforementioned requirements and functionalities.

5.1 Industrial Middleware

A key role in this system architecture is performed by the industrial middleware that is a distributed service-based integration layer that aims to ensure a transparent, secure and reliable interconnection of the diverse heterogeneous hardware devices (e.g., robotic cells and Programmable Logic Controllers (PLCs)) and software applications (e.g., MES and SCADA) presented at the PERFoRM ecosystem. An important innovation of this integration layer is its distributed and cloud approach, instead of the centralized ones that can be mostly found nowadays and can act as a single point of failure as well as a limitation for the system scalability. For this purpose, this distributed integration layer handles the interconnection of these heterogeneous production components by following the service-orientation principles, i.e., each one is exposing their functionalities as services, which will be discovered and requested by the other components.

The common definition of a middleware is a system or software component, which is used to connect different applications or systems. The target is to be able to establish a communication between these applications without them having to know about each other's inner structure. Middleware systems are getting increasingly important in the industrial world, where a huge amount of systems with different tasks are involved and have to work together to keep the production running. This reaches from low level sensors, actuators and controllers (PLCs) to management systems for ERP.

As Figure 7 is showing, the PERFoRM Middleware will act as a way for industrial hardware and other manufacturing level devices to communicate with higher level Software applications. Both vertical (e.g. from PLC to MES) and horizontal (e.g. simulation to data analytics or PLC to HMI) communication should be supported.

An important innovation of the PERFoRM middleware will be to follow a more distributed approach, where the middleware itself can be a distributed system running in a cloud platform, instead of the centralized ones that can be mostly found nowadays. The problem with the latter can be that they act as a single point of failure as well as that the scalability of such a middleware system can be limited. Furthermore, the middleware should follow a service-oriented approach, where the connected heterogeneous software and hardware components are exposing their functionalities as services, which will be discovered and requested by the other components.

Legacy production systems will need to use adapters, which translate their own data model into the standard interfaces defined in PERFoRM, to be compliant to this approach. Within the PERFoRM architecture, the necessary standard interfaces for this communication are defined as well (see Chapter 5.2). As far as these interfaces and data models are used, the middleware will act as a platform to connect to whenever a certain service needs to be used.

The middleware will provide a service registry, so called Yellow Page system, which is storing the different services available in the whole system. Additionally, it will take care of linking components which need to communicate and – if necessary – translating the data in a transparent manner. These basic functionalities will be the core of the middleware, as it is planned to be a more lightweight and not too complex system. Any kind of sophisticated intelligence, such as orchestration engines for the services, marshal use of redundant services and services life-cycle monitoring, are not included in the basic concept of the middleware. Instead the middleware is designed to have open interfaces to add more functionalities and intelligence as e.g. Plug-Ins or Add-Ons.

As PERFoRM is looking for solutions which are directly applicable for industrial use, an important focus of the middleware design will be the integration of industrially accepted existing solutions. Multiple vendors already provide middleware systems, namely SIMATIC WinCC Open Architecture, IBM Integration Bus and Red Hat JBoss Fuse, which follow some of the requirements set for the project. These solutions already provide some required functionalities, such as service registration and discovery, monitor and control the routing of message exchange, prioritization of messages delivery, data transformation, mapping/protocol translation (transformers and converters), security or exception handling, and data persistence. These solutions will be evaluated and - if they fit - included to be the core engine of the PERFoRM middleware. In this case, they can be complemented with pluggable modules focusing the missing or advanced features.

5.2 Standard Interfaces

Aligned with the general vision for the Industrie 4.0 platform, one of the key challenges that the PERFoRM architecture aims to tackle is the aspect of interoperability in real industrial environments, dealing with the representation and seamless exchange of data originating from a wide array of entities, often from different, albeit related, actions levels. The interconnection of heterogeneous legacy HW devices, e.g. robots and the respective controllers, and SW applications, e.g. databases, SCADA applications and other management, analytics and logistics tools, is one of the main goals currently being pursued in this vision.

To this effect, the PERFoRM architecture employs the adoption of standard interfaces as the main drivers for pluggability and interoperability, aiming at enabling the connection between such devices and applications in a seamless and transparent manner. These interfaces should support the devices, tools and applications with the means to fully expose and describe their services in a unique, standardized and transparent way to enhance the seamless interoperability and pluggability, fully specifying the semantics and data flow involved in terms of inputs and outputs required to interact with these elements. Therefore, from the system point of view, the standard interface specification and development abstracts the underlying function operation making transparent the way how the several architectural modules interact and operate.

These interfaces should provide a set of functionalities related to a standardized service invocation, i.e.:

- The definition of the list of services to be implemented by the interface.
- The contract implementation of each service (i.e., the name, input parameters and output parameters)
- The definition of the data model handled by the services.

Note that an important requirement for the design of standard interfaces is the usage of service-orientation to expose the device/system functionalities as services.

For this purpose, a common data model is also adopted, serving as the data exchange format shared between the PERFoRM-compliant architectural elements. This data model covers the semantic needs associated to each entity, which in the particular case of industrial automation, means that the requirements related to each ISA-95 layer and their respective needs are considered. In this context, two particular data abstraction levels are taken into account, more specifically the machinery level, covering mainly layers L1 (automation control) and L2 (supervisory control), and the data backbone level, which covers layers L3 (manufacturing operations management) and L4 (business planning and logistics).

The specification of the data model composed by the standard interfaces will be performed in the task 2.3 of the PERFoRM project, but the preliminary study considers a joint solution for the data exchange format by IEC 66264 B2MML and IEC 62714 AutomationML [36]: B2MML implements the ANSI/ISA-95 family standards, but lacks of low-level data (PLC signals, I/O and control sequences) which are instead covered in AutomationML.

As such, full interoperability and harmonization of data at a system of systems level is achieved by coupling the standard interfaces with the data model for a common representation of data and system semantics. However, taking into account the integration of legacy devices and their own individual data models and semantic requirements, the addition of technology adaptors is also required in order to enable the translation and mapping of legacy data into the common PERFoRM representation, allowing for these devices to be conferred additional intelligence and integrated into the cyber-physical paradigm. This aspect will be addressed in further detail in the next section.

5.3 Technological Adapters

As previously referred, manufacturing companies are usually characterized by the use of legacy and heterogeneous systems for the management and the execution of their production process. At machinery level (L1 and L2 layers of ISA-95 standard) example of these systems are robots, CNC machines, PLCs and HMIs; at backbone level (L3 and L4 layers of the ISA-95 standard) examples of these systems are ERP, MES and production databases. The innovative architecture proposed in the PERFoRM project can be industrially accepted and really adopted only if the possibility to integrate the legacy systems is presented. For this reason, technology adapters are key elements to connect legacy systems to the PERFoRM middleware and to transform the legacy data model into the standard interface data model defined in Task 2.3 of the project (i.e. masking the legacy systems’ data/functionalities according to the PERFoRM standard interfaces). In this way, the technological adapters are only necessary when there is the need to connect a legacy component (e.g., an existing DB or robot) to the PERFoRM system.

Although at its essence the adapter function is to convert legacy data (non-PERFoRM compliant) into the PERFoRM standardized data model, one can further detail the adapter function according with its scope. As depicted in Figure 8, three different kinds of scopes are considered in the WP3 and tackled along its tasks, namely adapters to interconnect standard legacy systems, real-time legacy systems and HMI legacy systems. These adapters respond to the different types of legacy systems which can be found in a production environment and are able to seamlessly connect these systems with the industrial middleware and the higher level of the enterprise network (ERP, MES, etc.).

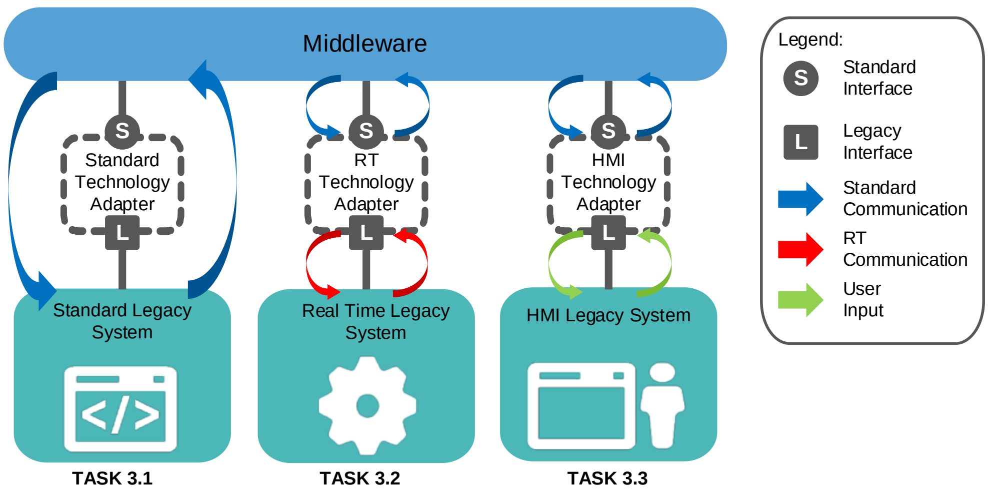


Figure 8 - Types of technology adapters

Real-Time (RT) constraints are particularly important when considering CNC machines and robotic cells as they may need quick adjustments and corrections according to the data acquired from low-level sensors locally installed in the production resource (e.g. vibration analysis of spinning spindles). Currently, no hard real-time constraints were identified in the use cases. However, should they arise, this aspect will be tackled within Task 3.2, as depicted above, where local and fast data processing is to be executed. HMIs, instead, can be used not only for monitoring and controlling the production resource but also for capturing human expert knowledge and support following human activities from past experience (e.g. change over and ramp-up operations can be supported by policies derived from past cases).

According to the outcomes from WP1, a list of legacy systems in each use case needs to be connected with industrial middleware, e.g., SAP APO ERP system, Oracle DB and Siemens PLC IM-151. The integration of these hardware equipment and software applications requires the use of proper technological adapters to transform the native data format into the data model defined by PERFoRM. Moreover, these adapters involve the HW and/or SW development and their implementation is strongly dependent of the selected technology for the industrial middleware (to be done in the Task 2.4 of the project). For example, Siemens WinCC OA (Open Architecture) provides a direct PLC interface which greatly simplifies the implementation of adapters for such kind of hardware equipment. The PLC integration is particularly important for the IFEVs and GKN use cases where PLCs are used to control the robotic cells. Another example of middleware technology considered in the PERFoRM project is the IBM Integration Bus. This solution offers, among other interesting features, a Database Input node that permits to retrieve updated data directly from a database: it creates a message flow that quickly reacts to changes to application data held in the database. Database connection and integration is particularly important for the Siemens and Whirlpool use cases where databases contain the information needed for feeding the predictive maintenance system (Siemens use case) and the KPI monitoring systems (Whirlpool use case).

5.4 Human Integration

The integration of the human in the loop is seen as a key factor to improve flexibility. The analysis of possible scenarios for human integration in flexible production systems within the PERFoRM industrial use cases has been undertaken in T2.1 and the results have been reported in D2.1 [9].

The results included some recommendations with implications for the planning and designing the human-machine and human-human interfaces, addressing two different levels: at strategic level, e.g., supporting decision-makers to take strategic decisions, and also at operational level, e.g., supporting operators or maintenance engineers to perform their tasks. In particular, the following functionalities should be considered and supported in PERFoRM:

- Consultation among team members and or with other experts, supporting virtual presence and/or the possibility to share the view, the screen, the information, voice and chatting space. These interfaces would allow to show a detail of a part, machine, etc. to share the information displayed on a screen, to attract the attention on a particular sound, so that other colleagues can be consulted, also if they are not physically present on the shop-floor.
- Acquisition of commands and data concerning human task execution (e.g., gesture recognition) and provision of feedback and alerts in case of errors.
- Delivery of condition-based instructions/on-the job-training, using multi-medial interfaces, offered in mobility and, in some cases they should provide augmented reality functionalities.
- Attract the attention and alert the operators in case of unexpected/anomalous events or behaviours, problems in the manufacturing processes and systems.
- Provide mobile turn-by-turn guidance to navigate the factory to retrieve tools, equipment, spare parts.
- Support visual inspection by providing information gathered by sensors.
- Display the dynamics and results of simulations and rescheduling, allowing the intuitive representation of alternatives and trade-offs, the comparison and evaluation of multiple objectives and performances, and facilitating collaboration and negotiation within and among teams.

It is important to underline that the above mentioned study was performed in the first few months of the project when the use cases were still not very well defined and the scenarios only outlined, therefore the recommendations for human integration and their implications in terms of human and human-machine interfaces have not to be considered as binding for the definition of the architecture but as possible requirements for the evolution of the systems.

In this architecture, the HW devices (e.g. tablets, smart phones, glasses or gesture devices) providing HMI functionalities (e.g., collecting data from the user or providing data to the user) are interconnected to the PERFoRM ecosystem through the industrial middleware by using proper adapters and standard interfaces.

5.5 Local Intelligence to Build Smart Production Components

Machines need, and are indeed becoming, to be smarter. To achieve this, robots and automation machinery need to be empowered with intelligence and higher processing capabilities to run more complex algorithms allowing them to process higher amount of data, producing a valuable analysis to be used when needed (i.e. in a real-time basis) and also supporting the seamless reconfiguration of the system and the achievement of self-* properties, e.g., self-adaptation, self-diagnosis.

In fact, the number of data being generated in shop floor plants is increasing at a very high rate. The proper analysis of the collected data assumes a crucial aspect, generating new knowledge that can be used to detect trends, deviations and possible problematic situations beforehand and in a timely manner. This data analysis can be divided into two different levels, as for where the data analysis is made as also regarding the scope of this data analysis. Data analysis more close to the machine is needed for a fast (real-time) data analysis, allowing a fast reaction to local machine situations. At a more global level, data analysis is needed for a more long-term optimization considering a wider set of information sources.

Alongside with this, machines are being equipped with an increasingly higher number of sensors, generating enormous streams of raw data. Different data sources are commonly collected, each one having its own particularities, such as current, voltage or power, temperature and vibration, tool wearing and others. This naturally disjunctive data need to be analysed and correlation patterns detected, identifying problems beforehand.

Sending these streams of raw data over the network has the problem of overloading the network with *non-meaningful* data. Therefore, by analysing this data locally and closer to the machine (where it is actually needed) has numerous potentialities, particularly by reducing the communication latency times, enabling faster reaction to events or trend deviations and by reducing greatly the network overload. Achieving this in a cloud environment would be harder due to the communication latency.

This PERFoRM feature is also aligned to what is commonly known as “fog-computing” [52] or “edge-computing”, being complementary to cloud-computing (in terms of data processing). In fact, fog-computing offers a faster, real-time response, to some extend less reliable information processing (due to the intrinsic nature of the processing), while the cloud computing offers a more reliable information processing. Naturally, this fog-computing paradigm brings additional requirements such as the need to connect/embed higher processing units near the machines or by the need to develop node-to-node coordination mechanisms (note that edge computing refers to the computation embedded in the machines and fog computing refers to situation where the computation is near the machines, e.g., in the routers).

In the PERFoRM architecture, the real time info processing module is located, if applicable, between the production components and the PERFoRM middleware, as illustrated in Figure 9, forming the smart production component. This intelligent module will react in a real-time basis due to operation/functional deviations and interact with the rest of the system as needed.

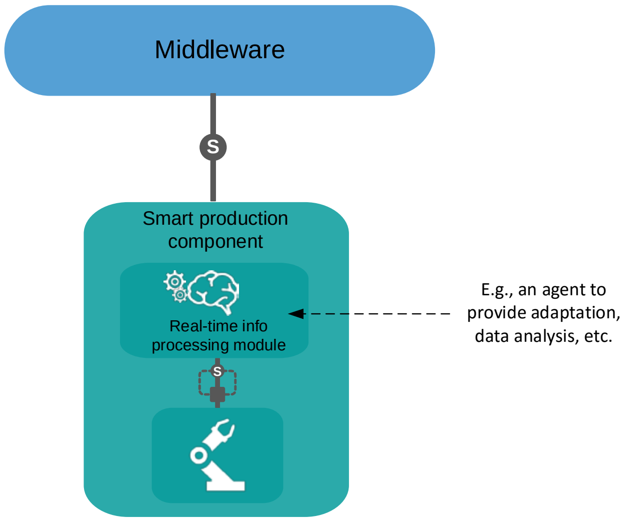

Middleware
S
Smart production component
Real-time info processing module
S
E.g., an agent to provide adaptation, data analysis, etc.

Figure 9 – Local data processing as a mean to achieve responsiveness.

The collection of data is not sufficient and needs to be accompanied with appropriate diagnostic algorithms. At this stage, it is crucial to not only understand the data to collect, but also what could be the information that the data could be used for. Additionally, and depending on the machine's type, it is important to adapt the processing algorithms to the machine's reality. Since the older machines are not equipped with the appropriate computational power, this local processing assumes a crucial aspect when the number of machines present at a given system is considerable, avoiding the transmission of raw data over the network. Furthermore, when this data needs to be sent over the network into a cloud environment for processing, latency might compromise real time needs. Therefore, local processing assumes a critical aspect due to avoid communication overload and ensure control real-time needs.

Alongside with this machine empowerment there is the need to cover, the many time overlooked feature, of bridging the Operational Technology (OT) and the Information Technology (IT) worlds. In fact, both worlds have been, for natural reasons in the past, separated apart by building islands around them. If in the past, shop-floors were composed by proprietary, hard to connect with lack of interoperability devices, this is being overcome by a wide dissemination of standardized protocols (take for instance OPC-UA) that are able of seamlessly interconnect devices. Due to this (as also with other technologies, namely the Internet of Things), there is an advent of a greater data being generated at the shop-floor level. In order to take fully advantage of the generated data, it is mandatory that higher levels of ISA-95 (normally those found in the IT) have access to these data. PERFoRM architecture empowers this vertical data interchange by enabling a transparent data-flow independently of where the data is located.

In another perspective, some production components can offer composed services based on the aggregation and orchestration of existing atomic services provided by individual smart production components. E.g., in Figure 10, a new service "*pick-and-place*" is composed by considering the atomic service "*transfer*" offered by the robot device and the atomic services "*open*" and "*close*" offered by the gripper device. For this purpose, the embedded orchestration engines should synchronize the execution of the atomic services according to a process workflow, being possible to be implemented by using different formalisms [39].

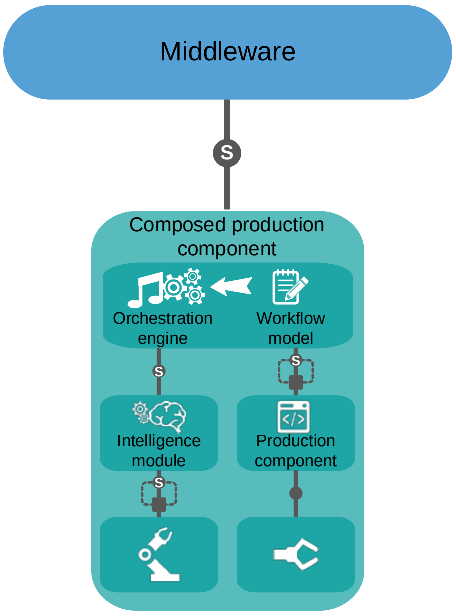


Figure 10 – Orchestration of services to support the creation of Composed production components.

During the SOCRADES project, a kind of Petri nets [40] engine tailored for formalizing orchestration mechanisms was developed and prototyped, embedding the engine into smart embedded I/O devices of the SOA-technology provider Schneider Electric Automation [41].

5.6 Advanced Tools for Planning, Scheduling and Simulation

Tools particularly designed with advanced algorithms and technologies to support the production planning, scheduling and simulation may improve the system performance and reconfigurability. These tools should be PERFoRM compliant, i.e. following the service orientation and using the PERFoRM native interfaces. The generalized advanced tools are developed within WP4. The development is clustered in three different groups: simulation solutions (later on addressed in D4.1), planning logic solutions (later on stated within D4.2 and D4.4), and intelligent decision support and visualization solutions (later addressed in D4.3).

Within the simulation cluster, different technologies are planned to be developed. The Simulation Environment (SE) is covering different interfaces to ensure the seamless integration of specific simulation models to dynamically acquire and provide data. Four different interfaces were identified:

1. Getting data from the middleware into the simulation environment.
2. Sending data from the simulation environment to the middleware.
3. Getting data from control planning logic into the simulation environment.

4. Internal interface within the simulation environment for the execution of the simulation model itself (actual simulation tools are Plant Simulation and AnyLogic).

Furthermore, the actual simulation tools will be enabled to model flexible and reconfigurable production systems, e.g. with agent based approaches within a classical discrete event simulation tool. As a last technology, validation methods will be developed to ensure the connection between the physical and the cyber level harmonize seamless.

The planning logic cluster includes solutions regarding to the dynamic production (re-) scheduling, planning and reconfiguration mechanisms, supporting flexible and reconfigurable manufacturing systems, e.g., used by the production planning for including reconfigurable robot cells. Furthermore, these solutions include approaches for multi-objective planning and agent-based approaches. It will be possible for the results of these tools, e.g. potential operation schedules and reconfiguration proposals, to be evaluated using the simulation environment. In such a way, these tools can be harmonized with the production system and used to guide its optimal operation.

The intelligent decision support and visualization cluster is concerned with data extraction and deriving predictions. This includes predictive maintenance tools, using e.g., regression analysis, Bayesian networks, clustering, time-frequency analysis and other methods. Moreover, what-if games, the link to the planning logic and simulation cluster provide decision support on the manufacturing system level. In the visualization sub-cluster, a generic tool is being developed to enable the visualization of different possibilities of visualizations utilizing data available from the middleware as well as from other advanced tools. Possible visualizations include value stream maps, KPIs, work orders, worker specific data, topologies and possible what-if scenarios. The visualization could be realized with open technologies including JavaScript, node.js and OPC UA.

In PERFoRM architecture, these tools are inter-connected to the system by means of the middleware and are able to connect to the data available in the system, either for gathering data for the simulation or by generating new data to be used elsewhere by other tools. Naturally, these tools will use the PERFoRM native language. In principle, these tools will not need the adapter since they are to be developed specifically for the project. Nevertheless, legacy modelling and simulation tools are pluggable by means of the use of proper adapter and standard interface. WP4 will determine the landscape of existing modelling and simulation tools within the four use cases, and determine the need for these tools to be directly supported by the PERFoRM framework, or whether they will continue to be operated independently.

Two exemplary process flows are provided to illustrate the data flow when interconnecting these kind of tools (note that the actual process flow of similar tools is dependent of the tool implementation strategy itself, since the PERFoRM architecture doesn't bind to a particular process flow strategy). Figure 11 illustrates the interconnection of a simulation tool to the PERFoRM ecosystem to access to the production database.

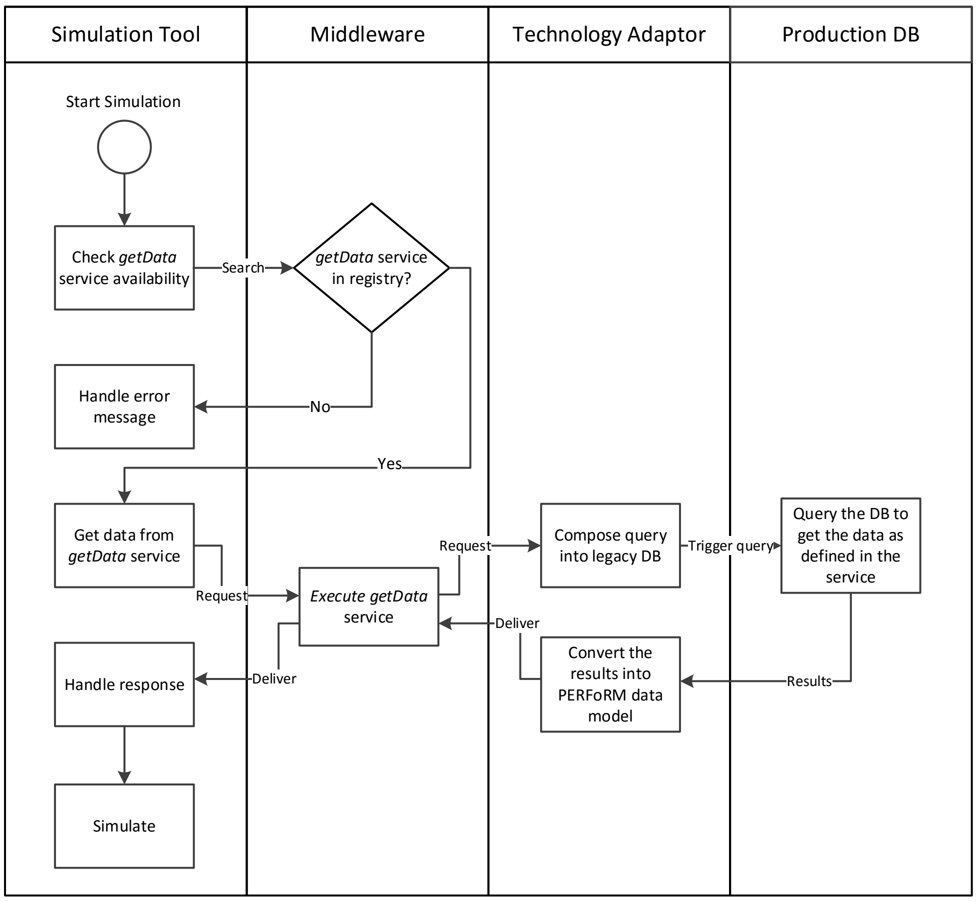

Simulation Tool
Middleware
Technology Adaptor
Production DB
Start Simulation
Check *getData* service availability
Search
getData service in registry?
Handle error message
No
Yes
Get data from *getData* service
Request
Execute getData service
Request
Compose query into legacy DB
Trigger query
Query the DB to get the data as defined in the service
Deliver
Convert the results into PERFoRM data model
Results
Handle response
Deliver
Simulate

Figure 11 – Exemplary flow diagram for a simulation tool integrated in the PERFoRM ecosystem.

Initially, the simulation tool, after knowing the available services registered in the ecosystem, queries the middleware for the availability of production historical data by requesting the execution of the "*getData*" service offered by the production DB. If a positive response is returned, the middleware will trigger the necessary actions for the data retrieval, namely using a data adapter (in the case where the data source is a legacy). After gathering the requested data, the tool will proceed with the simulation process.

Note also, that this process flow could be more complex by inserting multiple queries to different data sources. Additionally, the process could end by the generation of new data as result of the simulation process and/or by triggering further steps e.g., by using scheduling embedded in the simulation process or scheduling to use the simulation results.

Figure 12 depicts a process flow used by the KPI monitoring & visualization tool to get data parameters to support the KPIs monitoring by using a subscribe/notification procedure. For this purpose, the tool initially subscribes the notification of a parameter change by requesting the *subscribeParameter* service, offered by a PLC located at the shop-floor. Since this is a legacy PLC, the request must undertake a data conversion by means of the use of an appropriate technology adapter, converting a non-PERFoRM into a PERFoRM data structure. In the case the PLC is not available or not provide the desired service, the system middleware should reply accordingly.

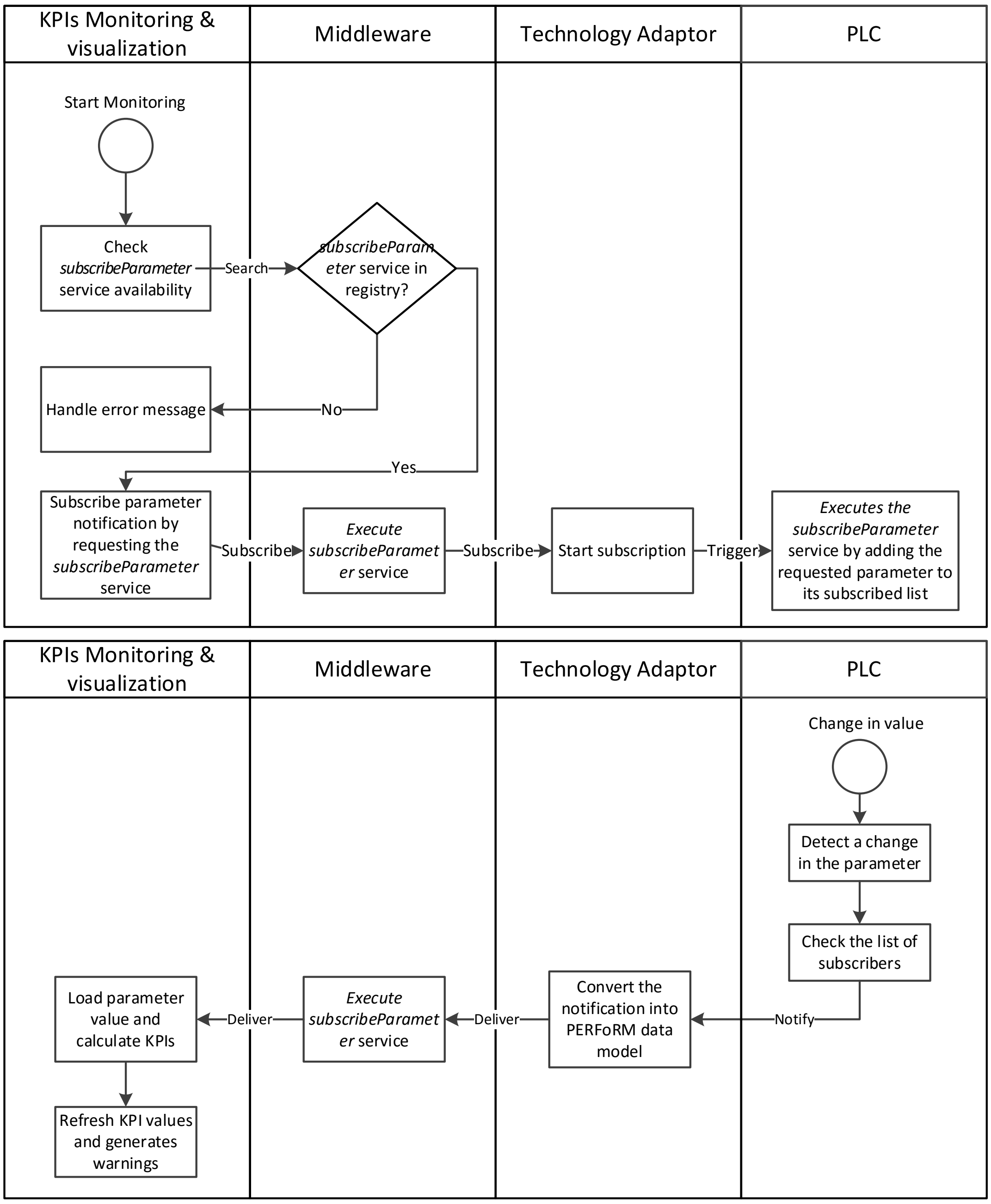

KPIs Monitoring & visualization
Middleware
Technology Adaptor
PLC
Start Monitoring
Check *subscribeParameter* service availability
Search
subscribeParameter service in registry?
No
Handle error message
Yes
Subscribe parameter notification by requesting the *subscribeParameter* service
Subscribe
Execute subscribeParameter service
Subscribe
Start subscription
Trigger
Executes the subscribeParameter service by adding the requested parameter to its subscribed list
Change in value
Detect a change in the parameter
Check the list of subscribers
Notify
Convert the notification into PERFoRM data model
Deliver
Execute subscribeParameter service
Deliver
Load parameter value and calculate KPIs
Refresh KPI values and generates warnings

Figure 12 – Exemplary flow diagram for a KPI monitoring tool integrated in the PERFoRM ecosystem.

When the PLC detects a change in the subscribed parameter, it checks the list of subscribers and notifies them of such occurrence. Naturally, the KPI monitoring and visualization tool can subscribe several parameters, possibly from different sources, according the correlation to calculate the desired KPIs, which are analysed and displayed in a user dashboard

As previously referred, these tools use advanced algorithms and technologies, some of them using MAS technology. MAS [6, 7] is a suitable approach to provide flexibility, robustness and responsiveness by decentralizing the control over distributed, autonomous and cooperative intelligent control nodes. In spite of these important benefits, the real time constraints and the emergent behaviour in industrial environments can be pointed out as weaknesses. However, the

MAS tools developed in PERFoRM don't face soft or hard real-time restrictions since these tools are placed at strategic and/or tactical planning and control levels. Additionally, the emergent behaviour should be seen as a potential benefit and not as a problem, since boundaries can be used to ensure stability during the emergency process. In fact, several commercial planning and scheduling solutions are already operating in big companies (see for example the MAS solutions developed by Smart Solutions [37]), as well as the Gartner's Strategic Technology Trends for 2016 report that predicts the use of MAS technology as a base for numerous mobile applications by 2020 [38].

5.7 Mechanisms for the Seamless Reconfiguration and Pluggability

All the aforementioned PERFoRM system architecture features wouldn't be fully exploited if the architecture is not enriched with appropriate mechanisms for the seamless system reconfiguration as also the introduction of plug-and-produce concepts for a proper "modularity" approach.

In PERFoRM, the seamless system reconfiguration is achieved by using the features commonly used in the development of distributed systems, namely those under the technological umbrella of MAS and SOA, particularly service- registry, discovering and composition, which also enhances the plugability, and the proper design, development and deployment of self-* mechanisms, particularly those targeted for improving the system adaptability and reconfiguration.

In service-oriented design, entities that want to offer their functionalities, encapsulated as services, should publish these services in a registry repository that acts as a "yellow pages" functionality. In PERFoRM, the HW devices and SW applications need to register themselves into the system and particularly their catalogue of services (i.e. by means of the "yellow pages" registry). The plug-in of new services in the system is easily discovered by the other entities through the use of a service discovering mechanism, potentiating the cooperation/collaboration between different system components leading to the seamless system reconfiguration. For example, as illustrated in Figure 13, consider a system comprising the "Process A" and "Process B" that are interacting with an industrial "Robot". In case of system reconfiguration, through replacing the "Process A" by "Process C", the intelligence (e.g., an agent) of the "Process A" should de-register its service from the service registry and the intelligence of the "Process C" should register its service. Automatically, and on-the-fly, the intelligence of the industrial robot discovers the new service and adapts its internal behaviour to start interacting with the new plugged "Process C".

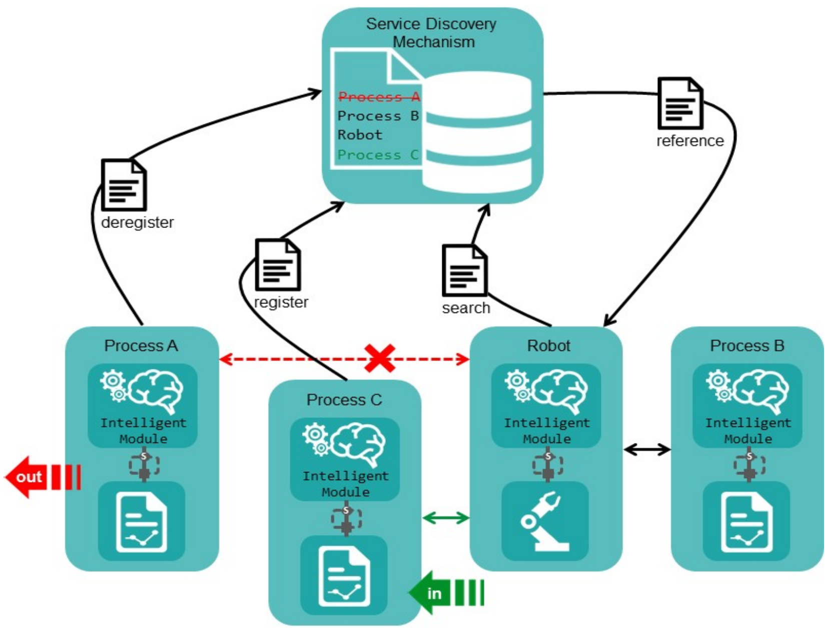


Figure 13 - Service registry and discovering.

After the discovering process, the services can be requested to be executed in a smoothly manner and following a proper process flow.

It is also worthy to note that in order to have a fully functional system, the service registration must be accompanied with the necessary information regarding the tool that offers them, e.g., its location or service quality. Therefore, the service- registry and discovering, offered by the middleware, must follow the best practices of similar MAS and SOA systems already in use. The pluggability of production components, supported by the registry and discovering features, and already successfully tested in SOCRADES and IDEAS projects, enables the seamless, online and on-the-fly reconfiguration.

The consideration of intelligence modules attached to the production devices allows to transform the traditional devices into smart production components. The use of the real time info processing module is one example, but these modules, e.g., implemented using MAS technology, can support the implementation of the seamless system reconfiguration. In particular, they can reason about the best opportunities to proceed with the reconfiguration and decide the best way to implement the reconfiguration. For this purpose, the aforementioned service-orientation features, and particularly service registry and discovery provided by the industrial middleware platform, are used as a mean of performing the reconfiguration on-the-fly, i.e. without the need to stop, reprogram and restart the system. Usually, the reconfiguration might happen when a resource is plugged-in/out or when the performance parameters of a resource are changed. This reconfiguration “block”, illustrated in Figure 7, differs from the previously described classic tools and is rather a set of procedures that will support the reconfiguration process, which will be also deeply designed in WP4.

In this perspective, several self-* properties can be identified in the PERFoRM system architecture, namely self-adaptation, self-diagnosis, self-optimization and self-organization.

Self-adaptation and self-diagnosis are provided by individual smart production components embedding the real time info processing modules and self-optimization is provided by the different advanced tools for planning, scheduling and simulation running in the PERFoRM cloud environment. The self-organization leading to the system reconfiguration emerges from the interaction of the smart production components that have embedded local and global driving self-organization forces implemented using MAS technology.

Self-* properties allow the seamless adaptation and reconfiguration of the system; however, in some circumstances the human role is impacted. In these cases, the operators have to be involved in the re-organization process.

6. Mapping the System Architecture Specification for the Industrial Case Studies

The PERFoRM architecture will be fully developed and instantiated into four industrial use cases, presenting different and specific requirements and covering a wide spectrum of production domains, namely a highly specialized large compressor production facility, a highly customizable producer of small-size electric vehicles, passing by a producer of home appliances and a producer of components to the aerospace industry. The mapping of the generic architecture into these use cases is described in the following sections, aiming to structurally validate the designed system architecture and also identify missing links and misunderstandings during the design phase.

This section instantiates the PERFoRM architecture to each of the four industrial use cases [48].

6.1 Siemens Case Study

Brief Description of the Case Study

The first use case is related to a factory producing industrial compressors and gas separators. It is characterized by highly complex systems of several tens of thousands components, which are typically only produced once on a customer specific basis. At the same time single parts can be very heavy and big, requiring special machining stations. As these stations are typically quite expensive they cannot be set up multiple times within a factory. Together with machining times of several days up to 2 weeks, this produces critical delays and costs in case of machine failures and breakdowns.

Currently, maintenance activities are done only re-actively and on separated IT-systems, namely: a) the maintenance scheduling and b) failure reporting. Thus, maintenance tasks are sometimes recognized late and cannot be scheduled accordingly.

The objective of this use case process is to integrate the separate systems supporting the identification of disturbances in the production as early as possible and to deliver to all involved stakeholders (e.g. maintenance, operation, scheduling, logistics) as much information as possible. Basically, three different scenarios can be distinguished when detecting early the disturbances:

1. The machine can still operate (maybe with limited capabilities) and a future maintenance should be planned. The work does not need to be rescheduled.
2. The machine cannot operate, but a repair can be done right away. The work does not need to be rescheduled.
3. The machine cannot operate and maintenance will need to be carried out as soon as all necessary material and resources are available. In this case, the work should be rescheduled to another machine.

This integration ensures a faster elimination of disturbances, and a reduction of delays in production and machine downtimes by introducing a proactive maintenance system.

Architectural Mapping

Considering the particularities of the described use case, the general system architecture is mapped as illustrated in Figure 14, where the generic blocks have been replaced by the legacy hardware devices and software applications installed at the plant or to be developed throughout the project. Namely the maintenance tools and machines are the source of failure reporting and machine condition monitoring. Their information will be stored in the Order Equipment Efficiency (OEE) database for further elaboration. Additionally, human operators can open up maintenance tickets, informing the maintenance staff about disturbances on the shop floor within the maintenance database.

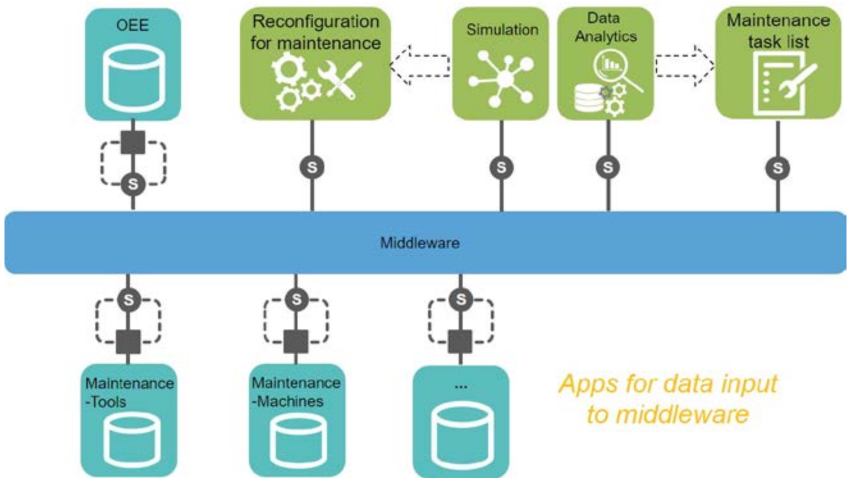


Figure 14 - Architecture mapping for the Siemens use case.

Since they are legacy systems, proper technological adapters are required to transform their proprietary interfaces in the standard interfaces defined by PERFoRM.

In addition, several new tools are considered to provide advanced features related to the analysis of the data gathered from the existing systems and now integrated. In particular, the data analysis tool aims to analyse the collected data and to identify in advance possible disturbances, generating warnings for the maintenance task list. The simulation tool aims to create several What-If? scenarios which can be compared by KPI's, allowing the selection of the best maintenance schedule.

6.2 IFEVS Case Study

Brief Description of the Case Study

The second use case considers a factory plant dedicated to produce micro-electrical vehicles. At the moment, the production line is actually operated completely manually with a welding operator in each island with also multi- skilled competences. The line is being automatized to support the production's efficiency and to permit the necessary flexibility for the production of different type of vehicle configurations (i.e. the easy switch from one vehicle configuration to another one).

This use case aims to enable a high quality production line for micro-electric vehicles, despite the throughput. For this purpose, the seamless integration of modular stations (each one composed by welding robots and Programmable Logic Controllers (PLCs)) is crucial to achieve flexibility and reconfiguration in the production system in order to allow the production of low amounts of micro-cars in an economical manner.

Architectural Mapping

Considering the particularities of the described use case, the general system architecture is mapped as illustrated in Figure 15, where generic blocks were updated by the legacy HW devices ad SW applications covered by the use case.

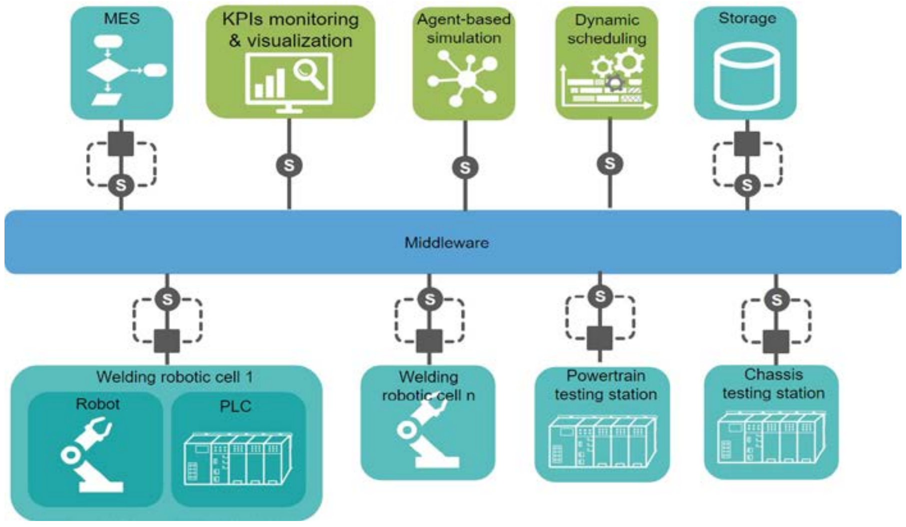


Figure 15 - Architecture mapping for the IFEVs use case.

Several welding robotic cells, as well as the powertrain testing station (rolling bench test to check the functionalities of the motorized axle frame) and the chassis testing station (geometrical test of the chassis to check if all the assembly complies with the design), are interconnected to the MES system through the industrial middleware. In addition, agent-based simulation and dynamic scheduling tools are considered to increase the system performance and flexibility.

These hardware devices and particularly the software application can access the data stored in the database also by using the middleware. The integration of these hardware equipment and software applications will require the use of proper technological adapters to transform the native data format into the data model defined by PERFoRM.

HMIs are related to the KPIs monitoring and visualization, and also used as a graphical interface between the operator and the system for the production's traceability.

6.3 Whirlpool Case Study

Brief Description of the Case Study

The third use case is related to a factory plant dedicated to produce microwave ovens. The current factory continuous adaptation and medium term reconfiguration mechanism is based on a set of processes (Factory Master Plan, Profit Plan, Cost Deployment) which aim at improving key performance indicators through modification of factory assets and organization described by key business factors. Of course KPI are mainly driven by shop floor data of each single facilities and departments. The behaviour of these facilities is monitored in order to meet middle- and long-term goals to benefit. Currently, the data gathered at shop-floor level lack of uniformity (different formats and source of data) and, moreover, correlation (i.e. each data is treated and analysed without a model or a tool able to describe how each KPI is linked with others KPI or input factors or KBF)

The use case aims on installing a real-time shop floor data acquisition to be able to react immediately on reasonable requirements in production planning. For this purpose, the data acquired and collected from the shop floor should be analysed and correlated to extract in advance the KPIs and also to detect earlier possible disturbances or performance degradation. This analysis can be complemented with simulation facilities.

Architectural Mapping

Considering the particularities of the described use case, the general system architecture is mapped as illustrated in Figure 16, where generic blocks were updated by the legacy HW devices ad SW applications covered by the use case.

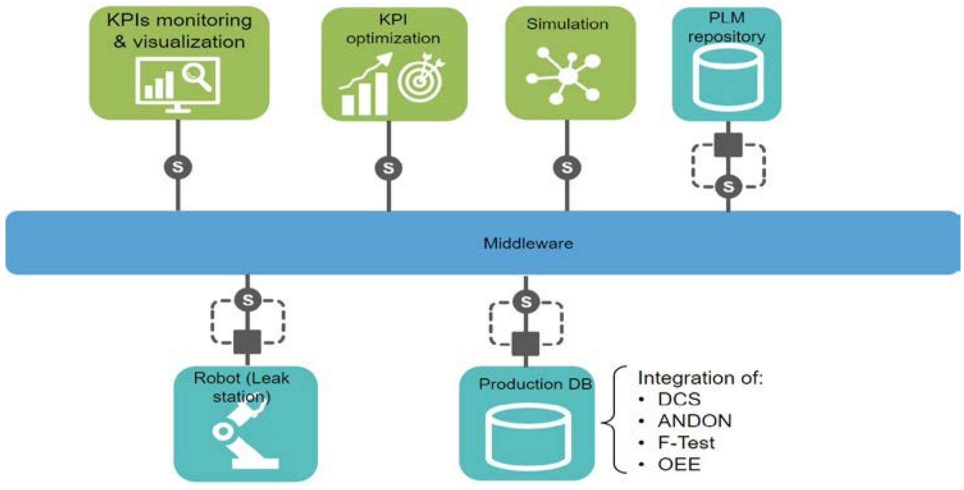


Figure 16 - Architecture mapping for the Whirlpool use case.

As shown, the data acquired from the shop floor and currently collected by several databases, will be integrated in the PERFoRM ecosystem by the middleware and in a PERFoRM database. The integration of these legacy databases will require the use of proper technological adapters to transform the native data format into the data model defined by PERFoRM.

Several new tools are considered to provide the required advanced features, namely the monitoring and visualization system to support the on-line visualization of KPIs, the KPIs optimization tool that uses two different models (MPFQ-K model and Value Stream Map) to identify strategies to improve KPIs and also considers a simulation tool having what-if game functionality to support the analysis of the impact of several degrees of freedom in these KPIs.

The human-machine interaction is mainly reserved for key decision-makers (e.g., production and industrial engineering managers) that will use the monitoring and visualization tool, as well as the proper designed user interfaces for the simulation tool, to understand the current system performance and study how KPIs can be optimized.

6.4 GKN Case Study

Brief Description of the Case Study

The forth use case considers a factory plant that manufactures complex, high value jet engine components with very stringent quality characteristics. The production system extends a functional workshop with standalone work centers and a mix of dedicated and common resources. The level of automation is usually rather low and based on separate process automation cells with low level of process flow integration. The production system has to cope with a large variety of different components, low volumes and varying demands.

All data (master data) is stored and managed through the SAP ERP system, which provides the functions for production planning, scheduling and MRP and it also collects different kind of documentation from the processing and inspection in order to guarantee traceability. Some additional data and information is generated and made available in the PLM system (Team Center Engineering and Manufacturing). The information flow in the typical day-to-day process is a straight communication between different IT systems to CNC-machines, robots and other equipment whereas the near term production planning and scheduling is done by shop floor planners based on ERP data for long term order scheduling and customer demands.

The main objective is the improvement of the flexibility to demonstrate more agile and automated production using an integrated system that can complete a short sequence of common operations in the value adding process chain. The approach aims to develop a modular production cell concept that can be reconfigured with different automated or semi-automated processes. The cell and process modules should be easily and quickly changed depending on current production demands aligned with the ideas of plug-and-produce for cyber-physical systems. The business oriented criteria are to reduce lead times and increase the level of automation as well as equipment utilization. For this purpose, mechanisms for the seamless reconfiguration of the production process should be addressed by plugging-in/-out modular processes in robotic stations.

Several technical and organizational challenges need to be addressed, namely interfaces for process modules that allow simple and short change over time, methods for production cell planning and scheduling to maximize throughput, decision support to identify when to reconfigure the cell to have the best impact on cost and production lead time, and procedures to coordinate the human intervention in the reconfiguration phase.

Architectural Mapping

Considering the particularities of the described use case, the general system architecture is mapped as illustrated in Figure 17, where generic blocks were updated by the legacy HW devices and SW applications covered by the use case.

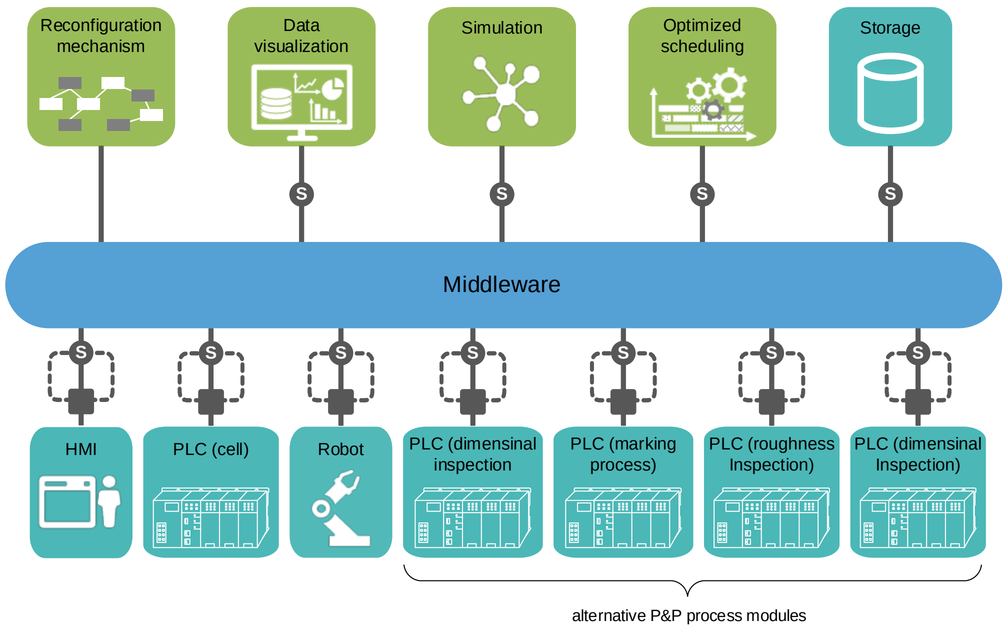

Reconfiguration mechanism
Data visualization
Simulation
Optimized scheduling
Storage
S
Middleware
HMI
PLC (cell)
Robot
PLC (dimensinal inspection
PLC (marking process)
PLC (roughness Inspection)
PLC (dimensinal Inspection)
alternative P&P process modules

Figure 17 - Architecture mapping for the GKN use case

The modular "micro-flow-cell" concept has a base configuration and components to host, control and coordinate the production on the process modules. The cell functions are a PC, PLC for communication and control, a robot system for part handling and processing and the safety system for the cell. Each of the different process modules, that can be replaced with short lead time – plug and produce – have their own PLC running the local control system.

All process related code/process parameters are downloaded from a central database, through the industrial middleware, to guarantee the full control of the configuration and versions of programs. In a similar way, the data generated from the executed processes and/or inspections are uploaded to the ERP system or databases and related systems for analysis and visualization (e.g., OEE/Stop time analysis), SPC / quality data).

The production planning and scheduling is planned to be supported by the optimized scheduling and simulation tools that will ensure the short-term and long-term scheduling, as well will trigger the need for use of the cell flexibility and reconfigurability, i.e. when to make the change-over in the cell.

The human interaction will be performed in two manners: i) at strategic level, a data visualization tool will provide to the cell manager a plethora of information related to the current status of the cell, namely current operating processes and tools, as well KPIs, and ii) at production cell level, a HMI supports the operator during its tasks. Note that all operations or operation sequences need to be started by an operator, which also confirms and finish/report the job in the ERP system before parts are shipped to the next station.

7. Analysis of Compliance of the System Architecture According to the Use Case Requirements

This chapter aims at matching the system architecture specifications according to the requirements established in WP1 for generic use and for the four industrial use cases considered in the project. In order to reach this goal two main steps, have to be considered:

1. Firstly, an appropriate methodology is required to verify the system architecture compliance to the given requirements. Thus, this step will include a research on appropriate software architecture analysis methods and their selection. As a result, it will provide matching algorithms, scenarios and crucial evaluative criteria, which will make an analytical background for the next step.
2. Lastly, a critical evaluation of the system architecture is to be conducted with respect to the achievements of the previous steps. This will include the following phases:
   a. Review of the system components and their connections;
   b. Review of the selected PERFoRM functional and non-functional requirements;
   c. Use of the matching algorithm and evaluation criteria selected in the first step and mapping the outcomes towards the review results.

7.1 Overview of Software-based Architecture Evaluation

Architectures are the key assets of each software-intensive system and are an important contributor to the success of a project. But "*How can you be sure whether the architecture chosen […] is the right one? How can you be sure that it won't lead to calamity but instead will pave the way through a smooth development and successful product?*" [42]. The process of architecture evaluation tries to cover these questions by analysing architectures to identify risks, to show the suitability for a given task and to verify that the quality requirements for the system have been addressed in the design.

Starting from the early 90's, system designers and architects recognized the necessity to evaluate the developed architecture before it is implemented, in consequent, a wide variety of evaluation methods have been developed by different research institutes up to today [43]. These methods can be divided in four main categories: Experience-based, Simulation-based, Mathematical modelling and Scenario-based [44]. While the Simulation and Mathematical approach are quite accurate, the effort for developing models and implementing parts of the architecture is very high and for the most projects not applicable in an early project phase. Hence, the Scenario-based method together with experienced people who have the necessary domain knowledge is the mainly used approach for evaluating architectures.

One of the first scenario based methods which have been accepted and successfully used in the industry, are the Software Analyses Architecture Method (SAAM) [45] and its successor the Architecture Trade-Off Analysis Method (ATAM) [46], which are still the leading methods for analysing architectures in the context of quality attributes and functional coverage, hence these two methods will be considered as the evaluation method for the PERFoRM architecture.

Before providing a short introduction into the two methods and determine the approach for evaluating the architecture of this project, it is important to understand what scenario-based evaluation means, what quality attributes can be inspected and what the expected output is.

7.1.1 Background of scenario-based evaluation methods

The evaluation of architectures will tell if the architecture is suitable or problematic with respect to its goals. These goals can be in conflict or at least, they will be prioritized. This means, the evaluation will not tell if the architecture is "correct" or "incorrect", it will also not precisely rate architectures, e.g., on a 10-points rate. But what it does is it tells if architectural decisions have risks or not regarding to the quality requirements of the system and if the architecture is suitable to the given tasks. With this, the evaluation process gives the following outputs [42]:

- **Prioritized statement of quality attributes requirements and the mapping to scenarios:** produces a mapping that shows how the architectural approaches achieve (or fail to achieve) the desired quality attributes and prioritize the quality requirements.
- **Suitability of the architecture:** the architecture is suitable if the system will run predictably and fast enough to meet its performance requirements, if it is modifiable in the planned way, if it provides the required behavioural function and the security level which it is required and if the system can be built with the resources in hand.
- **Overview of risks and non-risks:** "Risks" are potentially problematic architectural decisions or decisions which have not been made, while "Non-Risks" are good decisions that rely on assumptions that are frequently implicit in the architecture.

So how to get these outputs? Scenario-based methods consist mainly of three steps, which is shown more detailed in Annex A for SAAM and ATAM:

1. Describe the architecture in sufficient detail and in a way which is suitable for mapping the quality attributes.
2. Develop and prioritize relevant scenarios.
3. Evaluate and classify the scenarios.

Beside the description of the architecture, which is an important task in the evaluation process, but will be mainly done by the system architect, the developing and the evaluation of scenarios is a critical step. Naturally the scenario is the core of each scenario-based method; hence it is necessary to know what a scenario is.

Scenarios

The first job of an architecture evaluation is to elicit the specific quality requirements. In a perfect world, there is a requirements document where all quality goals are specified and described in an unambiguously way. Often, this document is missing or if, it is often not well written, does not include quality requirements and is not unambiguously in the early or middle stage of a project [46]. This is why it is necessary to concretize the actual meaning of a quality requirement in a scenario with the help of the requirement engineers and the stakeholders - "*A* *scenario is a short statement describing an interaction of one of the stakeholders with the system*" [42] and concretize the actual meaning of the attributes. Scenarios represent the stakeholders' interest, they help to anticipate the use of the system and the change to the system, they describe the stimulus and the response of the system and give details about the environment conditions.

Basically, scenarios can be differentiated in three types [46]:

- **Use-Case Scenarios,** which describe the intended interaction between an actor and the system. E.g., *the system should predict machine issues which can occur in the next 10 minutes to the next two weeks.*
- **Growth Scenarios,** which describe typical future changes to the system. Often, these changes have ramifications to different quality attributes. E.g., *double the observed machines without increasing the data latency.*
- **Exploratory Scenarios,** which describe changes to the system, which are not expected in the future, but are realistic assumptions. E.g., *change the ERP system solution from SAP to Oracle.*

Quality Attributes

Scenarios are strongly related to quality attributes of the system requirements. But what attributes are necessary to estimate the accuracy and suitability of an architecture? [42] and [47] proposed the following: Usability, Performance, Reliability, Availability, Security, Functionality, Modifiability, Portability, Variability, Subsetability, Testability, Conceptual Integrity, Building simplicity, Cost and Time to market.

Not all of these quality attributes can be evaluated, and it isn't true that one can tell if the architecture meets the quality requirements, just by looking on it. A main concern is that attributes influence each other, e.g., *Performance* and *Security*, which leads to a trade-off depending on the prioritization of the attributes.

Other quality attributes, e.g., *Usability*, which is certainly an architecture issue, have dependencies which are not in the scope of the architecture, which makes it hard for the evaluation or there are attributes which are often depending on implementation specific parameters, e.g., *Availability* and its related implementation specific parameters MTBF (Mean Time Between Failures) and MTTR (Mean Time to Recover).

The possible quality attributes to evaluate are also depending on the specific scenario-based method.

7.1.2 Approach for Evaluating the PERFoRM Architecture

Like above mentioned, there exists a variety of different evaluation methods. The most of them are based on the scenario evaluation methods SAAM and ATAM, hence these two methods are briefly described in the Annex A, forming the foundation for the project solution.

The PERFoRM system architecture is a high level description of a reconfigurable system for the manufacturing industry, which describes how assets and IT-Systems can be harmonized. Hence, for this first evaluation step, the architecture will be evaluated against more general requirements - a detailed architecture evaluation which includes the tools and other activities will not be in the scope of this evaluation.

The evaluation approach for the PERFoRM architecture will be a combination of ATAM and SAAM which is depicted in Figure 18. In contrast to ATAM and SAAM some steps will be skipped or adapted:

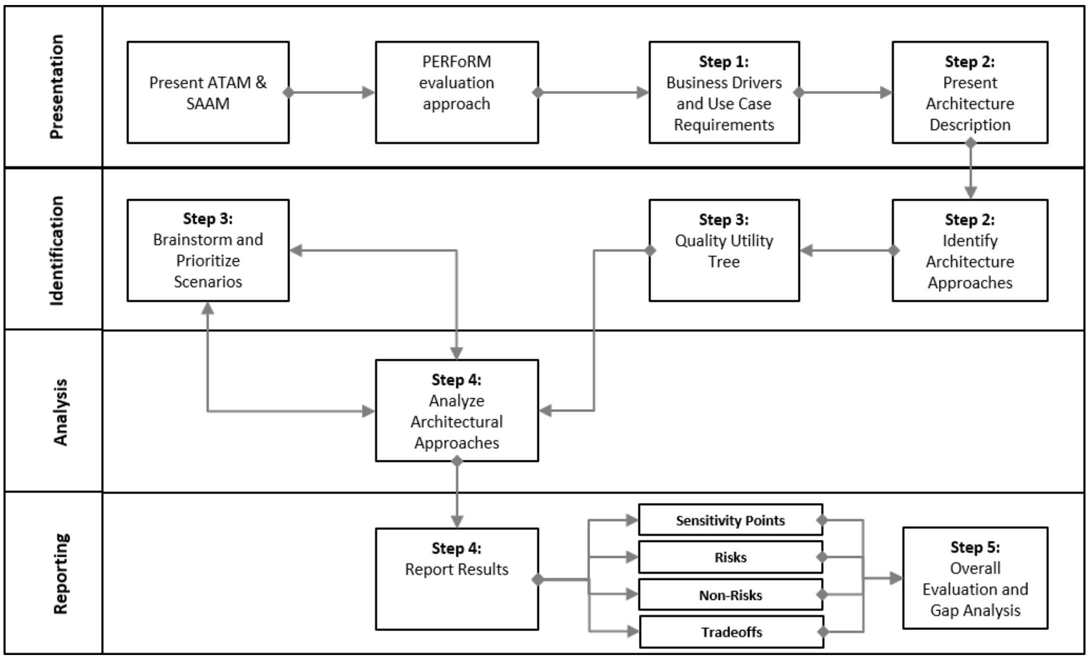

Presentation
Identification
Analysis
Reporting
Present ATAM & SAAM
PERFoRM evaluation approach
Step 1: Business Drivers and Use Case Requirements
Step 2: Present Architecture Description
Step 3: Brainstorm and Prioritize Scenarios
Step 3: Quality Utility Tree
Step 2: Identify Architecture Approaches
Step 4: Analyze Architectural Approaches
Step 4: Report Results
Sensitivity Points
Risks
Non-Risks
Tradeoffs
Step 5: Overall Evaluation and Gap Analysis

Figure 18 - The PERFoRM evaluation process

Step 1: Business Driver and Use-Case Requirements: in this step the business drivers and use-case requirements will be described. Also this step is part of the evaluation process; it will not be performed here. There is already a comprehensive business driver description in WP1 resp. in the use case work packages WP 7-10. Also the results of the requirements engineering from Task 1.3 will be used to define the scenarios. A summary of the requirements can be found in chapter 2 of this deliverable.

Step 2: Architecture Description: the architecture and the architecture approaches will be used to analyse the scenarios. For this, the description of the system architecture, performed mainly in chapters 4 and 5, will be used. If specific views are missing for the evaluation, they will be directly added to these chapters during the evaluation process.

Step 3: Scenarios Definition: an important task in the evaluation process is the definition of scenarios. Basically, the same methodology, as it is described in the step 5 of ATAM method, will be used. Hence, the quality attribute utility tree is used, which has already been shown in

Figure 21 of Annex A, to map the requirements to scenarios and to prioritize the scenario importance and its feasibility risk.

Step 4: Analyse Architecture: in this step, the architecture is analysed in the same manner as it is described in ATAM. Therefore, the template shown in Table 10 of Annex A is used to identify the Risks, Non-Risks, Trade-offs and Sensibility-Points.

Step 5: Overall Evaluation: finally, an overall evaluation and gap analysis based on the identified scenario types, risks, trade-offs and sensibility points will summarize the PERFoRM evaluation.

7.2 Matching the System Architecture Specification According to the Requirements

The scenarios represented in Table 5 are classified according to the general requirements results described in D1.2 and will be used as the foundation to setup the utility tree. Note that, in the following, Step 2 is skipped, because is already done in chapters 4 and 5.

Step 1 - Derive ASRs from the use case requirements or business drivers

The following table shows the derived ASRs, referring to the related use case requirements collected from WP1.2. Those use case requirements which don't have an impact on the architecture or are already covered by other ASRs will not be considered further in the architecture analysis.

Table 5 - Deriving Architecture Significant Requirements (ASR) from existing Use Case Requirements

N*	Use Case	Requirement	ASR
1.	Siemens	Scheduling systems able to handle maintenance tasks	Has no impact on the architecture
2.	Siemens	Scheduling system needs to interact with ERP system	The scheduling system should be able to interact with the ERP system to receive production tasks and to report the active schedule
3.	Siemens	Scheduling system needs to interact with Maintenance system (collection of maintenance tasks)	The scheduling system can interact with the maintenance system to gather new maintenance tasks.
4.	Siemens	Scheduling system should dynamically schedule the mix of production and maintenance tasks manually	A human should be able to schedule the mix of production and maintenance tasks in the scheduling system manually
5.	Siemens	Scheduling system should dynamically schedule the mix of production and maintenance tasks semi-automatically	The scheduling system should be able to schedule the mix of production and maintenance tasks semi-automatically
6.	Siemens	Scheduling system should dynamically schedule the mix of production and maintenance tasks automatically	Has no impact on the architecture
7.	Siemens	Condition detection system / method	Has no impact on the architecture

8.	Siemens	Conditions observations by humans should be included / given to the system	Machine conditions can be reported by a human/operator to the condition observation system
9.	Siemens	Condition observations by machines/sensors should be automatically gathered	Machine conditions should be automatically gathered by sensors or machine controllers and be saved in a database.
10.	Siemens	System should automatically elaborate the different states and input data (to calculate the machine conditions)	The predictive maintenance system can access all machine condition data, saved in a database, to elaborate the machine state
11.	Siemens	System should be able to detect the current machine state based on (e.g. knowledge based systems, SPC, …)	Has no impact on the architecture
12.	Siemens	System should be able to create maintenance tasks based on the observed machine conditions	Has no impact on the architecture
13.	Siemens	System should handle maintenance information (e.g. time between scheduled maintenance)	Has no impact on the architecture
14.	Siemens	the system should be able predict failures / breakdowns	Has no impact on the architecture
15.	Siemens	the system should be able to correlate machine conditions to predicted failures in order to create a new maintenance task	a. Has no impact on the architecture b. The maintenance tasks should be easy to maintain
16.	Siemens	the system should be able to decide if the maintenance task can be done internally or needs to be processed by an external supplier (e.g. machine supplier)	a. Has no impact on the architecture b. The decisions correlation should be easy to maintain.
17.	IFEVs	Process iteration	Has no impact on the architecture
18.	IFEVs	Process change	The system should provide a modular architecture and be able to integrate a decision tool to manage seamless process change.
19.	IFEVs	Feedback from production to design	The system should support feedback from the production systems to the design system.
20.	IFEVs	Robot welding and eventual re-work by an expert welder	The system should support the interaction between an operator and the process, to re-work parts if necessary
21.	IFEVs	Final test feedback to robot system configuration	Has no impact on the architecture
22.	IFEVs	100% Traceability and identification related to single product up to the supply chain	The system architecture should provide a service for tracing and saving data or a support tool to save data in DB; communication protocols should exist
23.	IFEVs	Simulation and prototype on the CPS environment	Integration of a Digital Twin of the CPS environment.

24.	IFEVs	Process interactions	The communication protocols between processes should exist and be provided by the system; processes should be modular designed
25.	IFEVs	Reducing both downtime and time to repair	A service or a tool should exist or be integrated within the system architecture to trace downtimes and time to repair
26.	IFEVs	Predictive maintenance	see 14
27.	IFEVs	Reconfigurability	see 17, 18
28.	IFEVs	Cost saving in reconfiguration	The system architecture should enable an analytical service or a tool integration to trace and analyse costs
29.	IFEVs	Cooperation between welder and robot	The system architecture should enable GUI integration for manual operations
30.	IFEVs	Frequent material or process changes	see 18
31.	IFEVs	One piece flow	Has no impact on the architecture
32.	IFEVs	On Time Delivery (OTD)	Has no impact on the architecture
33.	Whirlpool	(Relevant) Data Availability	The system should be able to manage, save, and transfer data; integration of measuring devices; data flows
34.	Whirlpool	Simulation	see 23
35.	Whirlpool	Process interactions	see 24
36.	Whirlpool	Model, i.e. process parameters interaction	Has no impact on the architecture
37.	Whirlpool	Process parameters definition	Has no impact on the architecture
38.	Whirlpool	Reconfigurability	see 17, 18
39.	Whirlpool	Cost saving in reconfiguration	see 28
40.	Whirlpool	Cost monitoring	The system architecture should enable an analytical service or a tool integration to monitor costs
41.	Whirlpool	Overall Equipment Effectiveness (OEE)	see 33
42.	Whirlpool	Identification and traceability (e.g. products and components)	see 22
43.	Whirlpool	Reducing both downtime and time to repair	see 25
44.	Whirlpool	Continual improvement (Ind Eng)	The architectural design should support reconfigurability and ongoing improvement features
45.	Whirlpool	User friendly model	The system architecture should enable integration of monitoring systems and interoperability between the tool and simulation system
46.	Whirlpool	Work In Process (WIP) optimization	The system architecture should enable services or tools integration to supports WIP optimization
47.	Whirlpool	On Time Delivery (OTD)	Has no impact on the architecture

48.	Whirlpool	Increasing the number of data collected	The system architecture should be able to manage, save, and transfer data; integration of measuring devices
49.	Whirlpool	Mobility, including comparison among different units	Has no impact on the architecture
50.	Whirlpool	New part reprogramming/setup through CAD critical paths	See 34
51.	Whirlpool	Connection between PLM and new testing programs	The system architecture should enable communication protocols and interfaces between PLM system and testing program; integration of both systems
52.	Whirlpool	Self-configuring system	The system architecture should enable a service or a tool integration to support the definition process of a root-cause based on the pattern
53.	Whirlpool	Meta-layer for different robot interface	The system architecture should provide a meta-layer for adapters and interfaces to integrate robot systems
54.	Whirlpool	Feedback to the process, based on failure control	The system architecture should enable communication with the processes in case of detected failures
55.	Whirlpool	Efficiency	Has no impact on the architecture
56.	GKN	Improve flexibility	The communication protocols between process should exist; processes should be modular designed
57.	GKN	Reduce cycle time	The system architecture should enable modular process exchange functionality
58.	GKN	Reduce cycle cost	The system architecture should enable modular process exchange functionality
59.	GKN	Reduce set-up time	The system architecture should enable modular process exchange functionality
60.	GKN	Improve production optimization	Has no impact on the architecture
61.	GKN	Improve machine utilization rate	Has no impact on the architecture
62.	GKN	Improve scheduling activity	The system architecture should be able to integrate a service or a scheduling tool
63.	GKN	Improve delivery performance	The system architecture should be able to integrate a service or a scheduling tool
64.	GKN	Improve reconfigurability	The system architecture should provide a flexible interface to a micro-cell (e.g. with other factory)
65.	GKN	Improve up-time machine	The system should reduce downtimes

Step 3 – Utility Tree

The Utility tree describes the ASRs with their importance and maps them to quality attributes. As quality attributes are the main drivers for architectural decisions, this is an important step to understand architectural decisions to fulfil a specific requirement and their related quality attributes. This step also adds the importance and the risk to achieve the ASR, where H stands for high, M for medium and L for low. Further details about the utility tree description are available in Annex A.

For each quality attribute refinement, one ASR is finally analysed in Step 4, where the importance is rated high.

Table 6 – PERFoRM Quality Attribute Tree

	Quality Attribute	Attribute Refinement	ASR	Use Case
Utility	Interoperability	Accessibility by humans	***(H,L)*** *Operator can access the scheduling system to schedule the mix of production and maintenance tasks manually or semi-automatically.*	***Siemens***
			(H,M) *Operator should be able to report machine conditions manually to the condition observation system*	
		Interaction of PERFoRM and legacy components	***(M,H)*** *ERP and scheduling system should communicate with each other to share new production tasks and to report the active schedule*	
			(H,L) *Scheduling and maintenance system should interact with each other*	
			(H,M) *Automatically and manually gathered machine conditions should be saved in a database*	
			(H,L) *Predictive maintenance system should be able to access all machine data saved in the database*	
	Reconfigurability	Operator-defined changes	***(M, L)*** *Tasks should be easy to modify/maintain. E.g. adding new tasks*	
			(L,L) *It should be possible to modify the decision correlation*	
	Interoperability	Accessibility by humans	***(H,M)*** *Operator should be able to interact with the process to be able to re-work parts if necessary*	***IFEVS***
			(H,H) *Operator (Welder) and robot should cooperate*	
		Interaction of PERFoRM and legacy components	***(M,M)*** *Integration of a Digital Twin of the CPS environment.*	
			(H,H) *Communication between process modules*	
	Reconfigurability	Process change	***(H,H)*** *Decision tool should be able to manage seamless process change*	
		Low costs	***(H,H)*** *Reconfiguration should be low on costs*	

	Traceability	Tracing products	***(H,H)*** *Single products must be traced, without exception during the process chain*	
	Reliability	Low downtimes	***(H,H)*** *The system should reduce downtimes*	
	Availability	Availability of Data	***(H,M)*** *All relevant data should be available to the system*	***WHR***
	Reconfigurability	System changes and Improvements	***(H,H)*** *The architectural design should support ongoing improvements / reconfigurability*	
			(M,H) *Process should be adapted based on the quality-check*	
	Interoperability	Accessibility by humans	***(H,L)*** *The system architecture should integrate monitoring systems*	
		Interaction of PERFoRM and legacy components	***(H,M)*** *The system architecture should support integration of simulation tools*	
			(H,M) *Integration of tools to handle CAD files and can analyze critical paths*	
			(L,H) *Integration of PLM system and CAD analyzation tool*	
			(H,H) *Integration of tools to support the definition of quality-check process.*	
			(H,H) *System should provide a meta-layer to integrate different robot interfaces*	
			(H,H) *Integration of tools to support WIP optimization*	
	Interoperability	Interaction of PERFoRM and legacy components	***(H,H)*** *Communication between modules should be possible*	***GKN***
			(H,L) *Architecture should support the integration of a scheduling tool*	
			(H,H) *Architecture should provide a flexible interface to a micro-cell, to be able to integrate it into other factories*	
	Reconfigurability	Modular design	***(H,M)*** *Process should be modular designed*	
			(H,H) *Architecture should enable modular process exchange functionality*	
	Reliability	Low downtimes	***(H,H)*** *The system should reduce downtimes*	

Step 4 – Analysation of the architecture approaches

Here the evaluation is examined for the highest rated ASRs, one for each quality attribute to reveal Risks, Non-Risks, Sensitivity and Trade-off points in the architecture. After the analyzation a comprehensive list of all points can be compiled (see Annex B) which can be used to identified the key architectural design decisions and their risks to archive these.

The architectural decisions are gathered from chapter 4 and 5 and are also summarized in Annex C.

ASR	*Scheduling and maintenance system should interact with each other*			
Quality Attribute	*Interoperability / Interaction of components (PERFoRM and Legacy)*			
Environment	*Normal operation / Maintenance*			
Stimulus	*A maintenance task for a machine is recommended*			
Response	*The scheduling system receives the new maintenance task*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Integration of both systems with service technologies		*S5*		*NR1*
Use of standard interfaces	*R5*	*S1*	*T2*	
Use of adapters for legacy systems			*T4*	*NR2*
Use of industrially adopted M2M protocols and IoT technologies	*R9*			
Use ESB (Enterprise Service Bus) for backbone	*R1*		*T3, T5*	*NR3*
Using standards for the representation of industrial data models	*R2*			*NR4*

ASR	*Operator should be able to report machine conditions manually to the condition observation system*			
Quality Attribute	*Interoperability / Accessibility by humans*			
Environment	*Normal operation*			
Stimulus	*An operator reports a machine condition or an abnormality of the system manually*			
Response	*The reported condition/abnormality is send to and received by the condition observation system*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Use of HMI and mobile devices	*R4*	*S3*	*T1*	
Communication between Operators HMI and condition system with services		*S5*		*NR1*
Use of adapters to add legacy HMI system			*T4*	*NR2*
Use ESB as backbone for the communication	*R1*			*NR3*

ASR	*Tasks should be easy to modify/maintain. E.g. adding new tasks*			
Quality Attribute	*Reconfigurability / Operator-defined changes*			
Environment	*Maintenance*			
Stimulus	*A new maintenance task is supported for the machine*			
Response	*The new maintenance task can be easily integrated into the system*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Using services which accept modification requests	*R9*	*S2, S4*	*T1*	*NR1*
Use of standard interface	*R5*	*S1*	*T2*	*NR4*

ASR	*Decision tool should be able to manage seamless process change*
Quality Attribute	*Reconfigurability / Process change*
Environment	*Normal operation*
Stimulus	*The process has changed and must be reconfigured*
Response	*The process can be changed seamlessly with the support of the decision tool*

Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Use of service-oriented design principles		*S4*	*T1*	*NR1*
Use of holonic design principles	*R15*			*NR8, NR12*
Use of adapters for the integration of legacy modules of the process	*R9, R3*		*T4*	*NR2*
Use of registry and discovery mechanism				*NR5*
Use of plug-and-produce concepts	*R7*			*NR6*
Use of self-* mechanisms	*R6, R8*			
Use of ESB as a backbone for the communication	*R1*	*S7*		*NR3*
Reconfiguration boundaries and nervousness control		*S6*	*T7*	

ASR	*Reconfiguration should be low on costs*
Quality Attribute	*Reconfigurability / Low reconfiguration costs*
Environment	*Normal operation*
Stimulus	*Reconfiguration of the process*
Response	*The process is reconfigured with low efforts and costs*

Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Service oriented design to support easy reconfiguration		*S4*	*T1*	*NR1*
Use of standard interfaces	*R5*	*S1*	*T2*	*NR4*
Use of registry and discovery mechanisms to find and use the services after reconfiguration		*S2*		*NR5*
Aggregate and compose services		*S5*	*T6*	

ASR	*The architectural design should support ongoing improvements / reconfigurability*
Quality Attribute	*Reconfigurability / System changes and Improvements*
Environment	*Design Time / Maintenance*
Stimulus	*Changing the system for improvements*
Response	*System can be changed*

Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Service oriented design to support easy reconfiguration		*S4*	*T1*	*NR1*
Use of distributed approaches e.g. MAS	*R13, R14*			*NR7*
Use of self-* mechanisms	*R8*			
Reconfiguration boundaries to stabilize the system		*S6*	*T7*	

ASR	*Architecture should enable modular process exchange functionality*			
Quality Attribute	*Reconfigurability / Modular design*			
Environment	*Normal operation*			
Stimulus	*Change parts/modules of the process*			
Response	*Parts or modules of the process can be easily changed and the process is still working*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Service oriented design with registry and discovery mechanisms	*R9*	*S2, S4*	*T1*	*NR1*
Use of distributed approaches e.g. MAS	*R13, R14*			*NR7*
Use of plug-and produce concepts	*R7*			*NR6*
Reconfiguration boundaries to stabilize the system		*S6*	*T7*	

ASR	*Single products must be traced, without exception during the process chain*			
Quality Attribute	*Traceability / Tracing products*			
Environment	*Normal operation*			
Stimulus	*Request the process steps for a specific product*			
Response	*All process steps for a produced can be requested*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
ESB as a backbone between the shop floor and the IT level	*R1*	*S7*		*NR3*
Services for all process steps	*R3*	*S2, S4*	*T1*	*NR1*
Database for saving the product data and the process steps are connected with the system via the ESB.	*R11, R12*	*S8*	*T8*	*NR9, NR10*

ASR	*The system should reduce downtimes*			
Quality Attribute	*Reliability / Low downtimes*			
Environment	*Normal operation*			
Stimulus	*Hardware or software failure*			
Response	*System is till operable*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Use self-* mechanisms for self-reconfiguration	*R8*			
Use of holonic design	*R15*			*NR8,NR12*

ASR	*All relevant data should be available to the system*			
Quality Attribute	*Availability / Data*			
Environment	*Normal operation*			
Stimulus	*A specific data is requested from somewhere in the system*			
Response	*The data is send to the requester*			
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**	**Non-Risks**
Use ESB as a communication backbone	*R1*	*S7*		*NR3*
Connect database to the backbone integration layer	*R11, R12*	*S8*	*T8*	*NR9, NR10*
Use cloud technologies	R10		T9, T10	NR11

Step 5 – Overall Evaluation and Gap Analysis

The gap analysis is a typical activity which maps various architectural requirements to find a balance for the future system architecture and proceed with the software and hardware design.

Quality attributes play a very important role in this process. To a large extend, these depend on the system requirements and determine the core qualities, which help to fulfil the required architectural design conditions, i.e. the overall architecture design as well as the implementation level. However, it is known that some of the quality attributes cannot co-exist or support the system architecture without leading to serious conflicts. Some of these conflicts occur, if the system architecture registers contradictive requirements, e.g. reconfiguring the system at the run-time (reconfigurability and performance) or tracing a product or a process activity during system reconfiguration (traceability and reconfigurability). Consequently, trade-offs should be taken into account during the analysis to avoid possible future risks.

During evaluation of the PERFoRM Quality Attribute Tree (see Table 6) and taking into consideration the importance of ASR and possible risks (“High”, “Medium” and “Low”), the following first gaps could be identified as follows:

1. 50% of collected ASRs are rated with the high risk and point out that the system is hard to achieve. These difficulties are expected conflicts between the two main quality attributes: high flexibility in reconfiguration of the PERFoRM system, system components and processes, on the one side, and a high degree of interoperability, which is also expected for the most of the system components, on the other. The main reason for the conflicting co-existence is that these quality attributes depend on each other to a large extend and influence each other’s performance during the run-time.
2. A large number of ASR includes the interoperability quality. This means that there is a high degree of integration of various components and tools in PERFoRM system and their interaction. Therefore, there is a high expectation of smooth communication and data exchange between all components and an easy integration procedure while connecting to the implemented legacy components and standard interfaces of the PERFoRM system.
3. Reliability, traceability and availability quality attributes are to a large extent dependent on the availability of data. These attributes rely on the functionality of the integrated data base and message exchange mechanism or data flows. As already described in the example above, a conflict can be expected while reconfiguring the PERFoRM system or exchange processes during the run-time and triggering or saving the updated data at the same time (e.g. tracing a specific product; analysing downtimes; etc.).
4. Finally, human factor has an influence on the reconfigurability of the PERFoRM system and may lead to unexpected conflicts. Thus, the attributes describe operator-defined changes, such as modify the decision correlation or adding new tasks, which may contradict with the execution or performance of the components during the reconfiguration processes.

The analysis of the architecture in Step 4 summarizes forty architectural decisions (see Annex C), which, again, point out next possible gaps. The most of the worked out decisions are based on the required technologies and their capabilities, which are classified according to five main groups. Although generally the architectural decisions are aligned with the best practices and covers the identified requirements for a seamless system reconfiguration, each group includes a thorough analysis of system functionalities, risks, sensitive points, possible gaps. Furthermore, the evaluation outlines the gaps and discusses the existing possibilities of how to fill them out or points out issues and problems for a future study.

Group 1

The first group regards the introduction of a service bus platforms, which is recommended to be applied in most of the cases (eleven cases) to verify the ASRs. Service-bus platforms, e.g., ESBs, are seen as the main functional backbone for communication between shop floor and high level IT-services. This goal also includes the usage of registry and discovery mechanisms for plug-and-play or plug-and-produce processes in order to find and use the services after registration as well as the integration of various condition systems, HMI devices, etc. with existing services.

The main gaps and possible risks for this group include the absence of the service bus drivers or badly specified plug-and produce mechanisms, which can result in a very generic architecture and less supportability for smooth process integration unacceptable for industrial needs. Above this, a high interoperability, which should be provided by an service bus platform, may influence the performance or interfere with the choice of applicable adapters and, thus, cause unexpected conflicts.

To close the gaps, several cloud technology solutions (Non-Risks) could be already specified in the previous chapters (see Chapters 4 and 5). Thus, the Non-Risks include the integration of the industrial middleware to reflect the service bus functionalities and provide a high interoperability by introducing service technologies. These solutions will be introduced to cover the plug-and-produce concepts, to verify the registration procedure and discovery mechanisms. Additionally, in order to overcome these gaps and risks, PERFoRM architecture considers the use of standard interfaces (to be defined in T2.3), considers the use of unified data models based on industrial data model standards (to be defined in task 2.3), and considers the use of technological adapters. In this way, PERFoRM enables the connection with legacy systems, contributing for a smooth migration in industrial environments.

A close analysis shows that the main gaps, which are related to the architectural decisions in this group, can be filled out by using the proposed Non-Risks decisions and, therefore, eliminate possible risks. Here a one-to-one correlation can be observed, i.e. each possible gap has a correspondent Non-Risk equivalent.

Group 2

The second important group could be detected in the next eleven cases and specifies the use of service-oriented principles, which are supposed to support easy reconfiguration of the processes and make modifications at run-time. Keeping in mind the main characteristics of the cloud technologies, the service-oriented design should enable aggregation of services of various types as well as support reconfigurability, reconfiguration and self-* mechanisms.

There are several gaps in this group. These include several risks concerning the implementation and usage of self-* mechanism solutions, a high degree of service integration and their orchestration, centralized/decentralized control mechanisms and secure communication between services or their integration within a middleware solution. As a result, a very generic architecture, tangled service integration instruments or, simply, break the communication flows during run-time can be expected.

The proposed Non-Risks solutions do not cover all the gaps. Firstly, the proposed self-* mechanisms for self-reconfiguration and ongoing system improvements are currently a research topic and, therefore, cannot be judged as highly reliable. Secondly, the open standards, i.e.

services, interfaces, communication protocol, which are planned to be used to verify the service-oriented principles are often less secure than proprietary standards. Although this can be debatable, the use of open standards adds an undeniable value, related with the possibility to eliminate the existent technological automation islands, traditionally present at the production shop-floors, by allowing the connection of heterogeneous components. Furthermore, this risk is naturally minimized by its confination to the internal shop-floor IT infra-structure.

Group 3

Other seven decisions concern the connectivity and the integration level of the PERFoRM system. Here belong such decisions as the use of adapters for legacy systems (e.g. to add legacy HMI system or integrate legacy modules of the processes) and integration of standard interface technologies, which help to support industrially adopted M2M protocols and integrate CPS devices.

Too generic interfaces and the use of IoT-technologies, which might not fulfil the hard industrial requirements (e.g. reliability or real-time) could be identified as possible risks for this group (for instance the incorrect selection of the IoT technology, e.g. Client/Server vs Publish/Subscribe approaches). The use of adapters can influence the communication performance and detain integration processes. To cover these gaps, it is planned to include standard interface technologies to integrate tools and CPS devices into the middleware solution.

Though, the Non-Risks ensure that the adapters will be developed for each legacy tool and, thus, connect the legacy tool to the system, the possibility of a flexible connection or exchange of other currently undefined tools in a flexible manner is not taken into consideration.

Group 4

Self-* mechanisms, the use of holonic principles, MAS and defining the reconfiguration boundaries are recommended to be applied in other seven cases to support the reconfiguration of the system and its components during the run-time and set the boundaries to stabilize the system and execute the nervousness control.

The main gaps of these group include difficulties concerning the introduction of self-* mechanisms (see Group 2) and realization of holonic principles and setting up the reconfiguration boundaries, which are supposed to stabilize the system. The possible risks show that a full distributed system design, which uses autonomous agents able to recognize themselves (e.g. to reduce downtimes) is hard to achieve. Therefore, the PERFoRM architecture considers not a fully distributed system design using agents but instead follows an holonic approach where the use of agents, if applied, are contained as tools that are connected to the PERFoRM ecosystem using standard interfaces. This solution contributes to reduce the complexity in the design and implementation of the system as well its reconfiguration.

Furthermore, though the proposed nervousness control mechanisms are able to make the PERFoRM system more robust, the system may lose sensitivity and skip important changes.

And finally, there is a possibility of defining too narrow reconfiguration boundaries, which may need solution.

Group 5

The last four architectural decisions cover the rest of various important goals, i.e. the use of standards to represent industrial data models, integration of data base for managing data, i.e. product data, process steps, process events and messages, etc. and integration of HMI devices.

The overload of data base and the loss of backup mechanisms are main gaps, which can have impact, as in any other system, on the PERFoRM architectural solution if taking industrial data models into consideration. Database is able to influence the performance of the PERFoRM system and provide failures or unexpected error in database schemes. In this way, and as the industrial best practices suggests, a proper storage and retrieval design approach is needed.

Another possible risk is the missing of data or data types of standard information model, which is minimized in the PERFoRM system design by considering an extensible and industrial-ready information data model (e.g., AutomationML and B2MML).

To cover these gaps such Non-Risks solutions can be taken into consideration. Firstly, the ESB solution and its supportability can be used to maintain a high variety of different database technologies, which can be easily accessible by other systems. Secondly, the use of standard interfaces should help to integrate selected database technologies in a “home-made” middleware environment.

A special concern should be detected towards a requested secure integration mechanism of the database technologies and shop floor devices into the middleware solution. In order to properly handle this gap, PERFoRM considers the use of industrial adopted middleware solutions (that are being consider in task T2.4).

8. Alignment of the system Architecture with Industrie 4.0

The concepts currently being developed within the PERFoRM project are aligned with the current state-of-the-art and road-map trends. Several matches can be devised from the “regulatory” documentation, namely [2, 49, 50].

From [2] several scattered key concepts can be seen. CPS are at the cornerstone of the Industrie 4.0, and PERFoRM addresses this by setting its foundation in the CPS concepts, particularly by promoting the symbiotic use of "digital" and "physical" layers of the manufacturing world and also considering its interconnection and interoperability. Optimized decision-making is also refereed in [2] and PERFoRM is addressing this by promoting a set of different tools that will allow the decision-makers, and particularly each of the use cases, to early detect deviations and performance degradation, allowing to take better, more accurate and timely decisions. Integration, either vertical and/or horizontal, is also mentioned to be crucial. PERFoRM also addresses this by promoting the use of a common PERFoRM data model, covering data needs from lower levels into higher levels as also from different domains within the same level (e.g., considering different data needs of devices) and by considering a distributed and interoperable middleware. Finally, and considering only a few key concepts, the Industrie 4.0 working group also recommends the development of a reference architecture. This is currently being developed by the Industrie 4.0 platform, named “Reference Architectural Model Industrie 4.0 (RAMI 4.0)” [49], but from the initial developments, PERFoRM is aligned with what is being considered in RAMI 4.0.

Aligned with what is aforementioned, the International Electrotechnical Commission white paper on Factories of the future also makes several remarks and recommendations [50]. The "connectivity and interoperability" is covered through the development of a distributed and interoperable middleware alongside with the design of a common and cross-layers data model. The “*seamless factory of the future system integration*” is accomplished by the connection of several information data sources as also the consideration of the human as a valuable data source itself. The “*integration of existent systems*” is also managed by the development of HW and SW adapters, adapting the native information language into the PERFoRM ecosystem. Modelling and simulation are also envisioned as crucial building blocks of future systems. Therefore, the PERFoRM architecture considers the use of such tool domains, particularly allowing beforehand to foresee future problems and solutions to these as to allow to optimize production processes. Other key concepts are located around the human operator and in its role in future production systems. PERFoRM considers this by moving the human to the centre of the architecture and to consider him as a flexibility driver in future systems. Therefore, a special emphasis is being devoted to the study of its integration and interaction to/from the system.

One of the most critical aspects being pointed out by all the reference documents is the use of standards and the standardization process promotion. PERFoRM also considers this issue as a major road-blocker breaker and enabler for the future industrial adoption of the system architecture, and is promoting the use of standardized approaches and technologies, e.g., using OPC-UA or AutomationML data models.

Table 7 summarizes the mapping of the PERFoRM system architecture considering the desired features established by Industrie 4.0 platform and IEC white paper on factories of the future.

Table 7 – Required features and mapping to relevant documents.

	Industrie 4.0 platform [2]	**IEC White paper [50]**	**PERFoRM**
CPS compliant, namely the integration of heterogeneous HW and SW	✓	✓	🟢
Optimized decision making (using advanced data analytics)	✓	✓	🟢
Connectivity and interoperability (vertically and horizontally)	✓	✓	🟢
Integrating existing (legacy) systems	!	✓	🟢
Modelling and simulation	✓	✓	🟢
Human-system integration (or human-centric design)	✓	✓	🟢
Industrial standards compliance	✓	✓	🟢
Migration strategies	✓	✓	🟢
Modularity	✓	✓	🟢
Service orientation to encapsulate functionalities	✓	✓	🟢
Security and safety	✓	✓	🟡
IoT and M2M communication	✓	✓	🟡
Cloud-based application infrastructure	!	✓	🟢

Legend: 🟢 - accomplished; 🟡 - partially accomplished; 🔴 - not accomplished

Finally, the smooth, secure and efficient migration from the traditional centralized structures and legacy systems, currently running in industrial environments, to the emergent distributed, agile and plug-and-produce systems, requires a special attention (note that newer devices and/or applications will co-exist with remaining existing systems). In fact, as also stated in [2], "*the journey towards Industrie 4.0 will be an evolutionary process*", and also reinforced in [50], the migration process from legacy systems is also crucial, particularly in the future adoption of such innovative systems. This issue may be simplified with the definition of migration methodologies and guidelines, and the adoption of industrial middlewares, standard interfaces and repository of wrappers.

Therefore, the PERFoRM architecture is also complemented and accompanied with a set of migration guidelines that allow the successful deployment into both legacy and new production systems, which will be studied and developed during the WP5.

9. Conclusion

This document describes the design of the modular system architecture developed under the H2020 R&D PERFoRM project, covering all the different layers in the production process identified by ISA95 automation model, being able to respond in a promptly manner to nowadays requirements as aligned with state-of-the-art visions, such as those advocated by the industry 4.0 initiative.

For this purpose, initially, the document has depicted several proposals for distributed control systems where concepts related with CPS and new manufacturing architectural trends were used. From these, the document has analysed the current best practices and lessons learned, deriving several requirements to be considered in the design of the PERFoRM system architecture. From the end user perspective, several requirements were also considered and accounted, complementing the previous architectural ones.

Merging these two distinct perspectives, the PERFoRM requirements were listed, culminating in architectural principles, such as the use of distributed smart components, seamless system reconfiguration and integrated planning, simulation and operational tools. From a technological point of view, several technologies and approaches were also envisioned to be part of the system architecture, namely:

- The use of service-orientation to expose the system functionalities as services.
- The use of a common platform for information exchange, i.e. a middleware.
- The use of a common language for the specification of standard interfaces.
- The compliance with legacy systems by means of technology adapters.
- The use of the human as a flexibility driver.
- The development of advanced planning, simulation and operational tools.

The designed system architecture was also validated by analysing its compliance according to the use case requirements, as well as by analysing its alignment with the Industrie 4.0 principles.

Recalling the evaluation of the architecture, performed in Chapter 7, some risks were identified, mainly related with technology aspects that don’t directly compromise the designed system architecture. Some of the identified risks will be overcome in posterior WPs, namely in the task 2.3 concerning the design of the data model and standard interfaces, the WP3 concerning the development of technological adapters and real-time information processing modules and WP4 related to the development of advanced tools and mechanisms to support seamless reconfiguration, visualization, planning and simulation of operating processes.

References

[1] H. Bauer, C. Baur, G. Camplone, and et. al., "Industry 4.0: How to Navigate Digitization of the Manufacturing Sector", Technical report, McKinsey Digital, 2015.

[2] H. Kagermann, W. Wahlster, and J. Helbig, "Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0", Technical report, ACATECH – German National Academy of Science and Engineering, 2013.

[3] S.-W. Lin, B. Miller, J. Durand, R. Joshi, P. Didier, and A. C. et al., "Industrial Internet Reference Architecture", Technical report, Industrial Internet Consortium (IIC), June 2015.

[4] E. A. Lee, "Cyber Physical Systems: Design Challenges", Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'08), p. 363–369, 2008.

[5] P. Leitão, A. W. Colombo, and S. Karnouskos, "Industrial Automation based on Cyber-Physical Systems Technologies: Prototype implementations and Challenges", Computers in Industry, vol. 81, pp. 11-25, 2016.

[6] J. Ferber, Multi-Agent Systems, "An Introduction to Distributed Artificial Intelligence", Addison-Wesley, 1999.

[7] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," Engineering Applications of Artificial Intelligence, vol. 22, pp. 979–991, Oct. 2009.

[8] Deliverable D1.2, "Requirements for Innovative Production System Functional Requirement Analysis and Definition of Strategic Objectives and KPIs", PERFoRM project, 29th March 2016.

[9] Deliverable D2.1, "Guidelines for Seamless Integration of Humans as Flexibility Driver in Flexible Production Systems", PERFoRM project, 11th March 2016.

[10] Deliverable D7.1, "Siemens Description and Requirements of Architectures for Retrofitting Production Equipment", PERFoRM project, March 2016.

[11] Deliverable D8.1, "Micro Electric Vehicles Description and Requirements of Architectures in View of Flexible Manufacturing", PERFoRM project, 29th March 2016.

[12] Deliverable D9.1, "Description of Requirements and Architecture Design", PERFoRM project, May 2016.

[13] Deliverable D10.1, "Use Case goals/KPIs and Requirements Defined", PERFoRM project, September 2016.

[14] Deliverable D1.1, "Report on Decentralized Control & Distributed Manufacturing Operation Systems for Flexible and Reconfigurable Production Environments", PERFoRM project, 30th March 2016.

[15] S. Karnouskos, V. Somlev, "Performance Assessment of Integration in the Cloud of Things via Web Services", Proceedings of the IEEE International Conference on Industrial Technology (ICIT'13), 2013.

[16] S. Karnouskos, A.W. Colombo, T. Bangemann, K. Manninen, R. Camp, M. Tilly, M. Sikora, F. Jammes, J. Delsing, J. Eliasson, P. Nappey, J. Hu, M. Graf, "The IMC-AESOP Architecture for

Cloud-based Industrial CPS", in Industrial Cloud-based Cyber-Physical Systems: The IMC-AESOP Approach, Springer, pp. 49-88, 2014.

[17] A. W. Colombo, T. Bangemann, and S. Karnouskos, "IMC-AESOP Outcomes: Paving the way to Collaborative Manufacturing Systems", Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN'14), pp. 255-260, 2014.

[18] A. Colombo and S. Karnouskos, "Towards the Factory of the Future: a Service-oriented Cross-layer Infrastructure", ICT Shaping the World: a Scientific View, European Telecommunications Standards Institute (ETSI), pp. 65-81, 2009.

[19] P. Leitão, N. Rodrigues, C. Turrin, and A. Pagani, "Multi-agent System Integrating Process and Quality Control in a Factory Producing Laundry Washing Machines", IEEE Transactions on Industrial Informatics, vol. 11, no. 4, pp. 879-886, 2015.

[20] M. Onori, N. Lohse, J. Barata and C. Hanisch, "The IDEAS Project: Plug & Produce at Shop-Floor Level," Assembly Automation, vol. 32, no. 2, pp. 124-134, 2013.

[21] P. Ferreira, S. Doltsinis and N. Lohse, "Symbiotic Assembly Systems - A New Paradigm", Procedia CIRP, vol. 17, pp. 26-31, 2014.

[22] M. S. Sayed, N. Lohse, N. Søndberg-Jeppesen, and A. L. Madsen, "Sel-Sus: Towards a Reference Architecture for Diagnostics and Predictive Maintenance Using Smart Manufacturing Devices", Proceedings of the 13th International Conference on Industrial Informatics (INDIN'15), pp. 1700-1705, 2015.

[23] Giovanni Di Orio, Gonçalo Cândido, José Barata, Sebastian Scholze, Oliver Kotte, Dragan Stokic, "Self-Learning Approach to Support Lifecycle Optimization of Manufacturing Processes", Proceedings of the 39th Annual Conference of the IEEE Industrial Electronics Society (IECON'13), pp. 6946-6951, 2013.

[24] A. Rocha, G. D. Orio, J. Barata, N. Antzoulatos, E. Castro, D. Scrimieri, S. Ratchev, and L. Ribeiro, "An Agent Based Framework to Support Plug and Produce", Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN'14), pp. 504-510, 2014.

[25] C. A. Marín, L. Mönch, P. Leitão, P. Vrba, D. Kazanskaia, V. Chepegin, L. Liu, and N. Mehandjiev, "A Conceptual Architecture Based on Intelligent Services for Manufacturing Support Systems", Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'13), pp. 4749-4754, 2013.

[26] Stefanos C. Doltsinis, Svetan Ratchev and Niels Lohse, "A Framework for Performance Measurement during Production Ramp-up of Assembly Stations", European Journal of Operational Research, vol. 229, n. 1, pp. 85-94, 2013.

[27] P. Webb and S. Asif, "Advanced Flexible Automation cell," Proceedings of the 6th Innovation for Sustainable Aviation in a Global Environment, pp. 296-301, 2011.

[28] I. Mezgar and U. Rauschecker, "The Challenge of Networked Enterprises for Cloud Computing Interoperability", Computers in Industry, vol. 65, no. 4, p. 657-674, 2014.

[29] G. Gonçalves, J. Reis, R. Pinto, M. Alves, and J. Correia, "A Step Forward on Intelligent Factories: A Smart Sensor-oriented Approach", Proceedings of the IEEE International Conference on Emerging Technology and Factory Automation (ETFA'14), p. 1-8, 2014.

[30] G. May, B. Stahl and M. Taisch, "Energy Management in Manufacturing: Toward Eco-factories of the Future – A Focus Group Study", Applied Energy, vol. 164, pp. 628-638, 2016.

[31] S. Scheifele, J. Friedrich, A. Lechler and A. Verl, "Flexible, Self-configuring Control System for a Modular Production System", Procedia Technology, pp. 398-405, 2014.

[32] P. Leitão, J. Barbosa, A. Pereira, J. Barata, A.W. Colombo, "Specification of the PERFoRM architecture for seamless production system reconfiguration", Proceedings of the 42nd Annual Conference of IEEE Industrial Electronics Society (IECON'16), October 24-27, Firenze, Italy, 2016.

[33] C. Peltz, "Web Services Orchestration", Hewlett Packard, Co., 2003.

[34] F. Jammes, H. Smit, J. L. M. Lastra, and I. Delamer, "Orchestration of Service-oriented Manufacturing Processes", Proceedings of the 10th IEEE International Conference (ETFA'05), pp. 617–624, 2005.

[35] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic Self-organization in Holonic Multi-Agent Manufacturing Systems: The ADACOR Evolution", Computers in Industry, vol. 66, pp. 99-111, 2015.

[36] R. S. Peres, M. Parreira-Rocha, A. D. Rocha, J. Barata, "Selection of a Data Exchange Format for Industry 4.0 Manufacturing Systems", Proceedings of the 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 2016.

[37] Smart Solutions, http://smartsolutions-123.ru/en/ (accessed on 30th June 2016).

[38] "Gartner Identifies the Top 10 Strategic Technology Trends for 2016", October, 2015, http://www.gartner.com/newsroom/id/3143521/ (accessed on 30th June 2016).

[39] A.W. Colombo and S. Karnouskos, "Towards the Factory of the Future: a Service-oriented Cross-Layer Infrastructure", ICT shaping the World: a Scientific View, European Telecommunications Standards Institute (ETSI), Wiley, New York, pp 65–81, 2009.

[40] T. Murata, "Petri Nets: Properties, Analysis and Applications," IEEE, vol. 77, no. 4, pp. 541-580, 1989.

[41] J.M. Mendes, A. Bepperling, J. Pinto, P. Leitão, F. Restivo, A.W. Colombo, "Software Methodologies for the Engineering of Service-oriented Industrial Automation: The Continuum Project", Proceedings of the 33rd Computer Software and Applications Conference (COMPSAC'09), pp. 452-459, 2009.

[42] P. Clements, R. Kazman, M. Klein, "Evaluating Software Architectures: Methods and Case Studies", Boston: Addison-Wesly, 2002.

[43] M. T. Ionita, D. K. Hammer, H. Obbink, "Scenario-Based Software Architecture Evaluation Methods: An Overview", Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering, 2002.

[44] M. Mattsson, H. Grahn, and F. Mårtensson, "Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability", Second International Conference on the Quality of Software Architectures (QoSA 2006), June 27-29, 2006, VÃsterås, Sweden (short paper).

[45] R. Kazman, G. Abowd, L. Bass, P. Clements, “Scenario Based Analysis of Software Architecture”, IEEE Software, vol. 13, n. 6, pp. 47-55, 1996.

[46] R. Kazman, M. Klein, P. Clements, “ATAM: Method for Architecture Evaluation”, Technical Report, Aug. 2000.

[47] M. Barbacci, M. Klein, T. Longstaff, C. Weinstock, “Quality Attributes”, Technical Report, 1995.

[48] P. Leitão, J. Barbosa, M. Foehr, A. Calà, P. Perlo, G. Iuzzolino, P. Petrali, J. Vallhagen, A.W. Colombo, “Instantiating the PERFoRM System Architecture for Industrial Case Studies”, Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing (SOHOMA’16), Lisbon, Portugal, 6-7 October, 2016.

[49] ZVEI, “Industrie 4.0: The Reference Architectural Model Industrie 4.0 (RAMI 4.0)”, http://www.zvei.org/Downloads/Automation/ZVEI-Industrie-40-RAMI-40-English.pdf (accessed on 30th June 2016).

[50] IEC, “White Paper: Factory of the Future”, International Electrotechnical Commission, ISBN 978-2-8322-2811-1, 2015.

[51] B. Scholten, “The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing”, International Society of Automation, 2007.

[52] Cisco Systems, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are", http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf (accessed on 20th September 2016).

Annex A - SAAM and ATAM Methods

This appendix describes the SAAM and ATAM scenario-based evaluation methodologies for system architectures.

SAAM

The Software Analyses Architecture Method was the first scenario-based evaluation method developed in the Software Engineering Institute (SEI) [45] published in 1993. It is used to compare different architectures solutions with a focus on the quality attribute "Modifiability", but can also address other attributes or a single architecture.

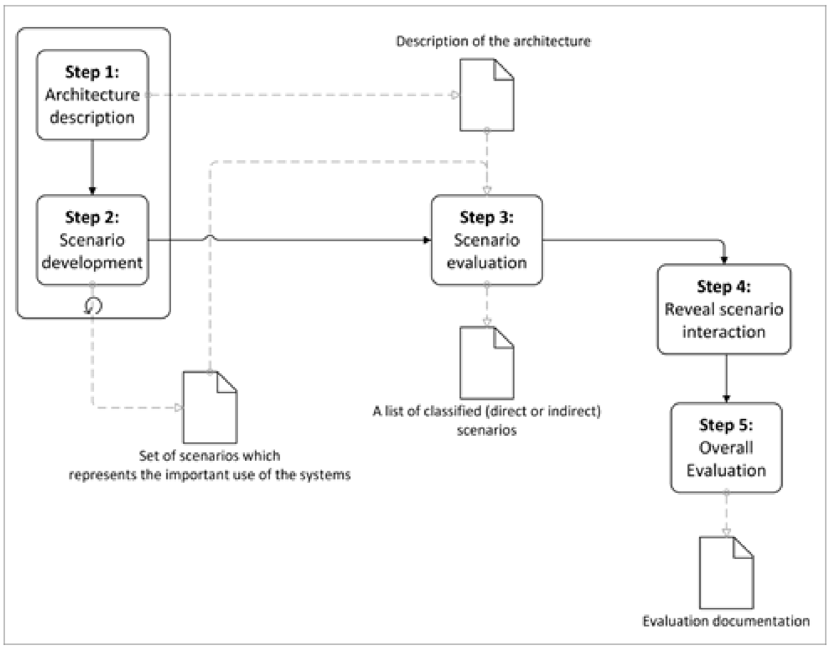

Step 1: Architecture description
Step 2: Scenario development
Description of the architecture
Step 3: Scenario evaluation
Step 4: Reveal scenario interaction
Step 5: Overall Evaluation
A list of classified (direct or indirect) scenarios
Set of scenarios which represents the important use of the systems
Evaluation documentation

Figure 19 - The Software Analyses Architecture Method Process

The SAAM method consists of 5 steps as it is depicted in Figure 19:

1. **Architecture Description:** this step describes the architecture(s) in a way which can be understood by all participants of the evaluation team. It is necessary to describe all evaluation relevant views of the architecture, e.g., systems component, relationships and behaviour.
2. **Scenario Development:** develop and prioritize use case scenarios which the system must support representing relevant roles, like customer, administrator, maintainer and developer
3. **Scenario Evaluations:** evaluate the scenarios developed in step 2 and determine whether they can be executed directly or changes are needed (indirect). In terms of indirect scenarios estimate the cost of modification which means effort to introduce or change of components and interfaces/connections. A template for the scenario evaluation is shown in Table 8.

Table 8 - Template for the scenario evaluation [45]

Scenario	Description	Type	Changes
<Scenario number>	*<Description of the scenario>*	*<Direct or Indirect>*	*<Necessary changes to fulfil the scenario>*

4. **Reveal Scenario** Determine indirect scenarios which need modifications to the same component or connection. This scenario interaction is an indicator for the architectures cohesion and component coupling. Table 9 shows a template for the scenario interactions recap.

Table 9 - Template for scenario interactions by module [45]

Component	Number of changes
<Component or connection in the system>	*<Number of all changes due to an indirect scenario>*

5. **Interaction and Overall Evaluation** Perform the overall evaluation depending on the priority of the scenarios

ATAM

The successor of SAAM is the Architecture Trade-Off Analysis Method (ATAM). While SAAM has its main focus on the "Modifiability" attribute by checking the type of the scenarios and identify how many changes are needed for an indirect scenario, ATAM is a much heavier evaluation method but producing also more sophisticated results, like the discovery of risks and non-risks, and the trade-offs and sensitivity points of architecture decisions.

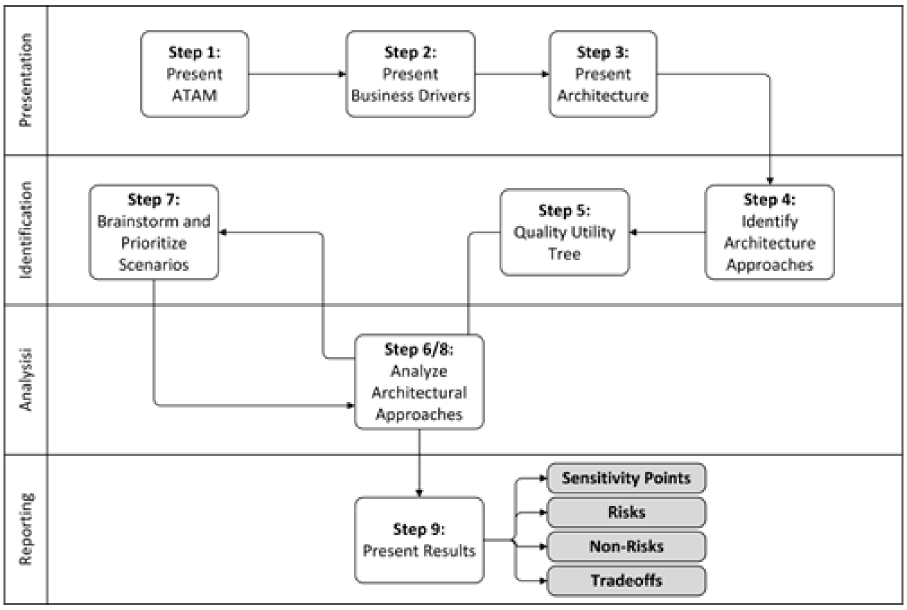

Presentation
Identification
Analysisi
Reporting
Step 1: Present ATAM
Step 2: Present Business Drivers
Step 3: Present Architecture
Step 4: Identify Architecture Approaches
Step 5: Quality Utility Tree
Step 7: Brainstorm and Prioritize Scenarios
Step 6/8: Analyze Architectural Approaches
Step 9: Present Results
Sensitivity Points
Risks
Non-Risks
Tradeoffs

Figure 20 - The ATAM Process

The ATAM method consists of 4 phases and 9 steps which are depicted in Figure 20 [46]:

Step 1: Presentation of ATAM: the steps and the outputs of the ATAM process will be explained briefly to all participants.

Step 2: Present Business Drivers: the project manager presents the system overview of a business perspective, so that all and particular the evaluation team understands the context of the system.

Step 3: Present Architecture: the architecture team presents the architecture decisions in an appropriate level of detail.

Step 4: Identify Architectural Approaches: in this step, the architecture approaches, meaning the use of patterns and tactics, are collected.

Step 5: Generate Quality Attribute Utility Tree: this step identifies, prioritize and refines the system's most important quality attributes requirements which were named in step 2 with the help of a quality attribute utility tree and the scenario mechanism.

Figure 21 shows a sample utility tree. The node of the tree is "Utility", the quality attributes are defined in level 1, level 2 gives a refinement of the quality attribute and the leaves are the most important ones, they represent the scenarios with a 2-tuple prioritization which includes the importance of the scenario and the difficult/risk to achieve it.

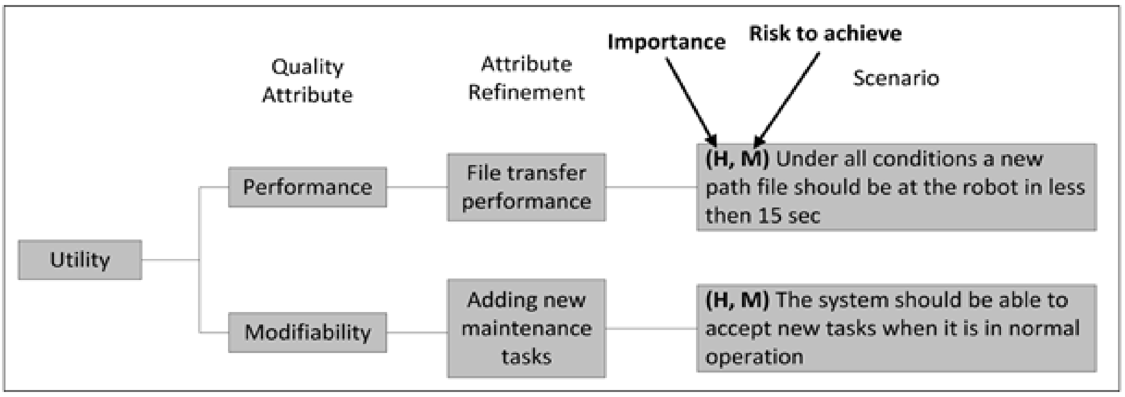

Importance
Risk to achieve
Quality Attribute
Attribute Refinement
Scenario
Utility
Performance
File transfer performance
(H, M) Under all conditions a new path file should be at the robot in less then 15 sec
Modifiability
Adding new maintenance tasks
(H, M) The system should be able to accept new tasks when it is in normal operation

Figure 21 - Utility Tree Example

Step 6: Analyse Architectural Approaches: the evaluation team can now probe the scenarios which were identified in the utility tree against the architecture decisions to identify the following evaluation points:

- **Risk** is an alternative that might create future problems in terms of a quality attribute or an architecture decision which has not been made.
- **Non-risk** is a decision that helps to realize the quality attribute requirements.
- **Sensitivity point** is a property for which a little change is likely to have a large effect on the system.
- **Tradeoff** is a property and a sensitive point, which affect more than one quality attribute.

A template for the documentation of the analysis of the architecture approach for a scenario is depicted in Table 10.

Table 10 - Template for an Architecture Approach Analysis [46]

Scenario	*<a scenario from the utility tree or from the brainstorming>*		
Quality Attribute	*<the related quality attributes>*		
Environment	*<relevant assumptions about the environment of the system>*		
Stimulus	*<a precise statement of the quality attribute stimulus embodied by the scenario>*		
Response	*<a precise statement of the quality attribute response>*		
Architectural Decisions	**Risk**	**Sensitivity**	**Trade-off**
<list of architectural decisions which affect the scenario>	*<risk ref. #>*	*<sens.point ref. #>*	*<trade-off ref. #>*
Reasoning:			
<Rationale why the architectural decisions help to meet the quality attributes response>			

Step 7: Brainstorm and Prioritize Scenarios: all stakeholders are part of the next steps. They were asked to identify (all) relevant scenarios which were not identified in the utility tree in a brainstorming and to prioritize these afterwards.

Step 8: Analyse Architectural Approaches: this step is identical with Step 6. The architect is asked to explain how the architecture supports the identified scenarios.

Step 9: Present Results: this step groups the identified risks and compares them with the business drivers from step 2. All evaluation information will be presented.

Annex B – Architecture Evaluation Points

This annex summarizes the list evaluation points to be used in the evaluation process.

Table 11: Risks and Non-Risks

N*	Risks	N*	Non-Risks
R1	Backbone ESB has no drivers for all needed technologies	NR1	Service technologies are well accepted by the industry and leads to a high interoperability
R2	Standard information model is missing data or data types and can't be extended	NR2	Adapters are developed for each legacy tool, hence it should be able to connect the legacy tool to the system
R3	No adapter for the legacy system available or adapters are too generic	NR3	ESBs as an integration layer all well accepted in the IT, hence there is a high probability that an ESB will also help in industrial context
R4	Mobile devices might be not accepted on the shop floor	NR4	Integration of tools and devices should be very easy with a standard interface
R5	Standard interface might be to generic for a specific task	NR5	Discovery with a registry is more reliable than using broadcasts for the discovery
R6	New process can't be described with the existing information models	NR6	Plug and produce concepts like discovery mechanism and standard hardware / software interfaces will help to dynamical reconfigure the process
R7	There is no real definition for plug and produce concepts, hence this is a very generic architecture decision which can lead to more risks	NR7	Autonomous agents would help to reduce the complexity of the reconfiguration.
R8	Self-* mechanism are a research topic and not very reliable	NR8	Autonomous agents can reorganize themselves to reduce downtimes
R9	IoT technologies might not fulfil the hard industrial requirements of e.g. reliability or real-time	NR9	Databases are very reliable and almost all systems can access and work with databases
R10	Cloud systems aren't accepted by the industry	NR10	ESBs do support a high amount of different databases out of the box
R11	Database is overload	NR11	Cloud technology makes the system very robust
R12	Database can be a single point of failure without backup mechanisms	NR12	Holonic system combine the advantages of centralized and distributed control
R13	Full distributed design is hard to archive		
R14	Agents are not accepted for control systems in the industry. Missing methods and software tools		
R15	Switch algorithms for centralized and decentralized control are necessary		

Table 12: Sensitivity and Trade-off Points

N*	Sensitivity	N*	Trade-offs
S1	Definition of information model	T1	Open Standards (Services, interface, communication protocols) are often less secure than proprietary standards
S2	Different service technologies can't be easily combined	T2	Defined information model vs. flexibility of system in terms of changing data and datatypes
S3	Operating system of the mobile device	T3	High interoperability provided by an ESB can influence the performance
S4	data format of a maintenance/production task	T4	Adapters will influence the communication performance
S5	if a centralized control is necessary it might not be easy to combine it within a holonic multi agent design	T5	Decision of the ESB technology might have an influence on the needed adapter
S6	Reconfiguration boundaries are to narrow	T6	Using one service protocols vs flexibility with different protocols
S7	Different adapters are needed depending on the ESB and it's capabilities	T7	Nervousness control mechanisms will make the system more robust but also less sensible to changes
S8	Database schema	T8	Database may influence the performance of the system
		T9	Cloud technology has normally no real-time capability
		T10	Public clouds are less secure in comparison to local servers

Annex C – Architecture Decisions

This annex summarizes the possible architectural decisions, which are recommended to apply to fulfil the specified ASRs.

N*	Architecture Decision
AS1	Use of service-oriented design principles
AS2	Aggregate and compose services (or skills)
AS3	Use of holonic design principles
AS4	Use of standard interfaces
AS5	Use of adapters for legacy systems (existing interfaces)
AS6	Use of industrially adopted IoT technologies and M2M protocols
AS7	Use of artificial intelligence (AI) methods, and particularly MAS
AS8	Embed advanced data analysis
AS9	Use Human-Machine Interface (HMI) and mobile devices
AS10	Use of augmented reality technologies
AS11	Use of distributed approaches, e.g., MAS and SOA
AS12	Use of registry and discovery mechanisms
AS13	Use of plug-and-produce concepts
AS14	Use self-* mechanisms, e.g., self-adaptation, self-organization and self-diagnosis
AS15	Consider reconfiguration boundaries and nervousness control
AS16	Use M2M and ESB (Enterprise Service Bus) technologies addressing backbone level
AS17	Consider a catalogue of adapters for existing interfaces
AS18	Use MAS and advanced optimization methods
AS19	Use advanced simulation frameworks
AS20	Use cloud technologies
AS21	Consider standards for the representation of industrial data models
AS22	Use gateways for data transformation (interconnecting backbone and machinery levels)
AS23	GUI as human interaction enabler