SHOP4CF

D3.5 SHOP4CF Architecture 4

Grant Agreement No.	873087
Project Name	Smart Human Oriented Platform for Connected Factories (SHOP4CF)
Work Package No.	WP3
Lead Beneficiary	PSNC
Delivery Date	31/12/2023
Author(s)	Michał Zimniewicz (PSNC)
Contributor(s)	Zuzanna Domagała-Schmidt (TUE), Paul Grefen (TUE), Konstantinos Traganos (TUE), Matteo Pantano (SAG), Adam Olszewski (PSNC), Genessis Perez (FZI), Pieter Becue (IMEC), Diego Francisco Carvajal Flores (UPM), Soubarna Banik (TUM), Karolina Dostatnia (PSNC)
Editor(s)	Michał Zimniewicz (PSNC)
Reviewer(s)	Pieter Becue (IMEC), Blazej Banaszewski (DTI), Raphael Prabucki (UO), Aske Bach Lassen (DTI), Julie Skødt Clausen (DTI)
Nature	Report
Dissemination Level	Public

This Project has received funding from the European Union’s Horizon 2020 research and innovation program under grant agreement No. 873087. Neither the European Commission nor any person acting on behalf of the Commission is responsible for how the following information is used. The views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect the views of the European Commission.

Document Revision History

Version	Date	Modification Reason	Modified by
0.1	18/12/2020	Initial version of the deliverable	Michał Zimniewicz (PSNC)
0.2	12/01/2021	Internal review	Blazej Banaszewski (DTI)
0.3	14/01/2021	Internal review	Raphael Prabucki (UO)
1.0	19/01/2021	Final version of the deliverable	Michał Zimniewicz (PSNC)
1.1	06/12/2021	Initial version of Deliverable D3.3 Changes summarized in Section 1.3	Michał Zimniewicz (PSNC)
1.2	09/12/2021	Internal review	Aske Bach Lassen (DTI)
1.3	13/12/2021	Internal review	Raphael Prabucki (UO)
2.0	20/12/2021	Final version of the deliverable	Michał Zimniewicz (PSNC)
2.1	01/12/2022	Initial version of Deliverable D3.4 Changes summarized in Section 1.3	Michał Zimniewicz (PSNC) Matteo Pantano (SAG)
2.2	08/12/2022	Internal review	Raphael Prabucki (UO)
2.3	12/12/2022	Internal review	Julie Skødt Clausen (DTI)
3.0	13/12/2022	Final version of the deliverable	Michał Zimniewicz (PSNC)
3.1	30/11/2023	Initial version of Deliverable D3.5 Changes summarized in Section 1.3	Michał Zimniewicz (PSNC)
3.2	10/12/2023	Internal review	Raphael Prabucki (UO)
3.3	12/12/2023	Internal review	Julie Skødt Clausen (DTI)
4.0	19/12/2023	Final version of the deliverable	Michał Zimniewicz (PSNC)

Abbreviations

AGV	Automated guided vehicle
EDC	Eclipse Dataspace Connector
FSTP	Financial Support to Third Parties
IDS	International Data Spaces
IoT	Internet of Things
K4+1	The Kruchten 4+1 architecture framework
KPI	Key performance indicator
MES	Manufacturing execution system
MQTT	Publish-subscribe network protocol for message queue service
PCB	Printed circuit board
ROS	Robot Operating System
SQL	Structured Query Language
UML	Unified Modeling Language
UT5	The Updated Truijens 5 Aspect Framework
WoT	Web of Things
WPx	Work package no. x of the SHOP4CF project

In addition, the acronyms of SHOP4CF components are listed in Section 5.

Executive Summary

This document presents the SHOP4CF framework architecture that aims at ensuring coherence and interoperability of the SHOP4CF software components under development. This framework architecture (reference architecture) serves as a guiding tool and provides a common template for concrete systems under design.

Requirements for the SHOP4CF architecture are defined by the pilot scenarios from the prior project deliverables and by the existing SHOP4CF components. Therefore, top-down (scenarios) and bottom-up (components) approaches to the architecture design were combined. In addition, the architecture conforms to the reference architectures: ISA-95, RAMI 4.0, FIWARE Smart Industry, and International Data Spaces, as well as to the architectures of prior research projects.

Methodology of undertaken design decisions as well as how they are organized and presented in this document are all based on established architectural standards and frameworks. The SHOP4CF architecture focuses mainly on the logical view of the Kruchten 4+1 architecture framework that provides a structure and specifies functionality of abstract modules. The document addresses not only software but also the platform and data aspects, as defined in the Updated Truijens 5 aspect framework.

A high-level logical view on the software aspect is designed as a set of six subsystems supporting manufacturing processes in three phases (design, execute, analyze) and at two levels (global, local). Coherence of the components is addressed by positioning of components within these subsystems and organizing connections between them.

Interoperability of the components is addressed with logical views on the platform and data aspects that together facilitate and standardize communication among the components. Separately, interoperability of the architecture with existing standards is discussed.

Logical views on the platform aspect define the organization of software and hardware that are necessary to deliver high-level functionalities. They focus mainly on the FIWARE middleware, and on how middleware components support connections among software components and with IoT.

Logical views on the data aspect ensure the uniform structure and meaning of data items exchanged by various modules as well as conformity to existing standards. This data architecture maps to relevant parts of the software and platform architectures to emphasize cross-dependencies between these aspects.

In addition, the document highlights possible areas for further design decisions, such as extending of the functionality or the middleware.

A special focus is put on designing the SHOP4CF architecture as an open architecture to facilitate its future extensions, even without the involvement of the SHOP4CF consortium.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose

The SHOP4CF project aims at providing technical means for manufacturing companies to find the right balance between automation and involvement of human workers. The technical means are based on functionalities of the SHOP4CF components that facilitate digitalization of factories [1].

This document presents the architecture of the SHOP4CF framework that ensures coherence and interoperability of software components, which are developed in the project and beyond.

The term "framework architecture" (also known as "reference architecture") refers to its role as a guiding tool. This allows for the design and deployment of concrete systems, which may involve a subset of all components, based on this architecture. For instance, the SHOP4CF pilots [2] or the FSTP experiments (see Appendix D) are specific trimmed views of the entire framework architecture.

Coherence of software components means that they can be positioned with respect to each other in this framework, so such positioning depicts whether components have similar, complementary or unrelated functionality. Interoperability means that integration of components that have complementary functionality is supported by their design that primarily addresses coherence of adopted technology standards and the interpretation of data to be exchanged.

1.2 Target audience

The main recipients of this document are users (factories), system integrators, and component developers.

Users (factories) look for the functional overview of SHOP4CF components. To this end, this document contains logical (functional) views that coherently present the high-level overview of components and their functions (Section 6).

System integrators are to bring together a specific subset of SHOP4CF components so they work together as a system based on this framework architecture. To help in this process, this document elaborates the platform architecture (Section 7) as well as defines several medium-level views to present how components are connected to each other (Section 9).

Component developers look mainly for a conceptual and technical design for interoperability. These needs are addressed especially with platform (including middleware) architectures and data architectures (Sections 7, 8, 9, and Appendix B).

1.3 Document lifetime

The architecture has been developed iteratively taking into account progressing project developments as well as the feedback from integrators and developers. It has been updated in yearly revision cycles during the project lifetime.

This document is the final, fourth version of the architecture released at the end of the the project. The following parts of this document are introduced since the previous version:

- Section 1.1 is updated taking into account the recent developments.
- Sections 6.3 and 6.4 are updated taking into account the actual pilot deployments.
- Sections 8.1, 8.2 and 8.3 are updated to stress the usage of the location and alert information both in the design and execution phase, and also to rename Resource Specification to Resource Definition (for coherency).
- Section 10.3 is extended to report on the implemented approach towards IDS.
- Section 10.5 is added to report on the investigated approach to interoperability with other industrial standards.
- Appendix D is extended to present the adoption of SHOP4CF architecture in new manufacturing scenarios.
- Minor updates and editorial improvements are implemented.

This revision is prepared by the two tasks of WP3: Task 3.5 “Continuous monitoring and updates” and Task 3.4 “Interfaces to other platforms”.

1.4 Structure of this document

This document is divided into four main parts. Part 1: “Approach for the Architecture” explains the background and the methodology for the architecture design. Then, Part 2 and Part 3 presents the actual architecture design, respectively from high-level and medium-level perspective. Part 4 complements the document with conclusions and additional information.

Part 1: Approach for the Architecture

- Section 1 is this introduction.
- Section 2 presents architectural models and frameworks used in the SHOP4CF architecture.
- Section 3 presents the project-defined requirements and relevant reference architectures.
- Section 4 explains how the SHOP4CF architecture is positioned within the complete design of the SHOP4CF project, i.e. what concrete subjects are in the scope of this document.

Part 2: High-level design

- Section 5 presents the overview of existing SHOP4CF components.
- Section 6 presents the high-level logical software architecture, incl. SHOP4CF components.

- Section 7 presents the high-level logical platform architecture, incl. the FIWARE middleware.
- Section 8 presents the high-level logical data architecture (concept data models).

Part 3: Medium-level design

- Section 9 describes how SHOP4CF components connect to the middleware and other systems.
- Section 10 presents the interoperability of the SHOP4CF architecture with existing reference architectures and standards.

Part 4: Conclusions and appendices

- Section 11 explains how the SHOP4CF architecture can be easily extended with additional functionalities and technologies.
- Section 12 provides the conclusions from this document.
- Appendix A presents the information exchanged via high-level logical interfaces.
- Appendix B discusses the technical FIWARE representation of the earlier-defined data models.
- Appendix C reports about the pilot questionnaire on MES.
- Appendix D presents the adoption of SHOP4CF architecture in new manufacturing scenarios.

2 Architectural standards

This section presents models and frameworks that are used to design and present the SHOP4CF architecture in the further chapters of this document. The frameworks ensure separation of concerns, so different contexts of the architecture can be transparently presented.

2.1 The Updated Truijens 5 Aspect Framework (UT5)

This section is a quotation from a project internal report [3].

Architecture encompasses more than software design alone – there are other important aspects that need to be structured as well. The UT5 framework [4] is used to identify and organize these aspects – see Figure 1.

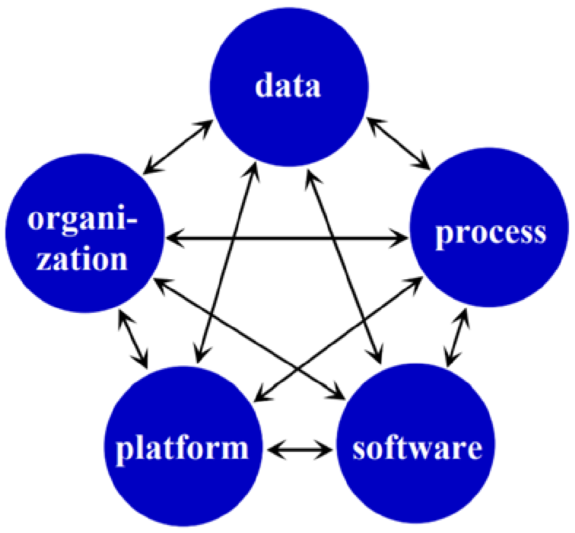

Figure 1 UT5 aspect framework

- The *software aspect* describes the organization of the software under design of an information system in terms of its modules and the connections between these modules. The specification of the software aspect is referred to as the **software architecture**.
- The *data aspect* describes the organization of the data in an information system, typically in terms of data structure diagrams or specifications. The specification of the data aspect of an architecture is referred to as the **data architecture**.
- The *platform aspect* describes the organization of the software and hardware, i.e. the technology assumed to be present to use the information system, both in terms of computing and networking facilities. The specification of the platform aspect is referred to as the **platform architecture**.
- The *process aspect* describes the organization of the (business) processes managed by or executed in an information system, typically in terms of business process models. A structure for the specification of a set of processes is referred to as a process architecture.

- The *organization aspect* describes how the information system under consideration is embedded into an organization for its design, implementation and maintenance.

2.2 The Kruchten 4+1 architecture framework (K4+1)

This section is a quotation from a project internal report [3].

In developing an architecture and the software conforming to this architecture, it is important to distinguish the specification of the user-oriented functionality (*what* the system defined by the architecture should do) from the specification of the technical realization of this functionality (*how* the system is realized by software developers and *how* it works after deployment). The well-known Kruchten 4+1 software engineering framework (K4+1) is used for this [5].

The K4+1 framework is shown in Figure 2 and explained below.

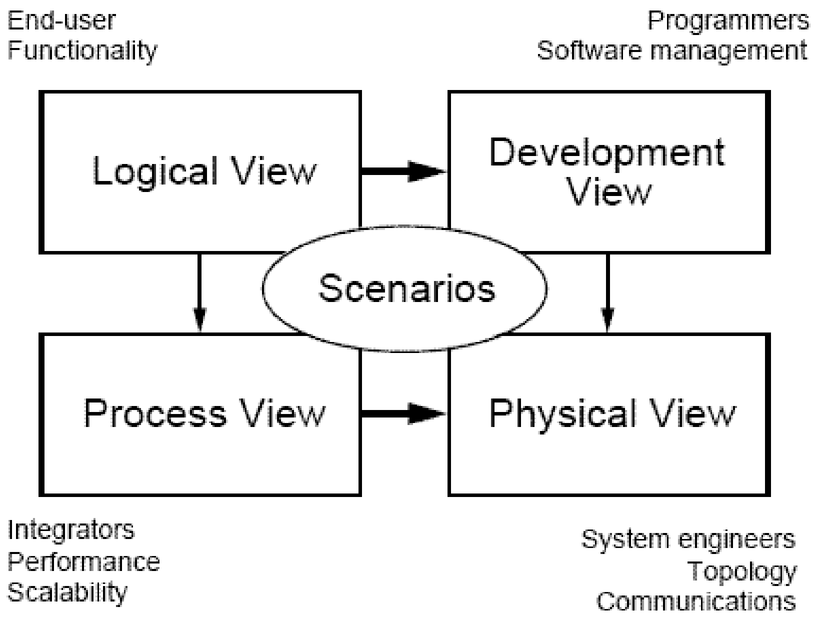

Figure 2 K4+1 view model of software architecture

The framework organizes the description of an architecture around four main views:

1. The *logical view* specifies the object/module models of the design, i.e. the structure of the application logic in abstract terms. This view mainly specifies the functionality of a system under design, so what the system should do.
2. The *development view* specifies the organization of the software in a development environment, i.e. the way the software development is supported to arrive at good software management. This view is concerned with getting good software, so how the system should be realized.
3. The *process view* specifies the concurrency and synchronization aspects of the software design, i.e. the way objects or modules in the logical view dynamically collaborate in parallel.

4. The *physical view* describes the mapping(s) of software onto hardware, thereby reflecting the distribution aspect. This view mainly specifies the operational deployment of a system, so what runs where?

Each of the four views has its prime stakeholders and its major concerns. This may lead to a content-wise divergence of ideas. To avoid this, the four basic views are illustrated by a fifth element:

5. The *scenarios* describe a few selected use cases that illustrate the four basic views. The scenarios make things concrete and provide a clear and practical basis for discussions between the various groups of stakeholders (associated with the basic four views) in the architecture design or analysis. As such, the scenarios are the 'content glue' that provides convergence of ideas.

2.3 Data modeling

Data modeling is the process of defining a data model, i.e. the structure and meaning of elements of data, and how they relate to each other and to the real world. This architecture uses the well-established formal approach by M. West (2010) [6].

The approach can be summarized as provided in Figure 3. The process starts with detailed data requirements. Based on the requirements, the concept data model is defined. The concept data model is the meaning of data and it consists of definitions of data entities, their attributes, and relationships between entities. This document presents the concept data model(s) as the logical data architecture.

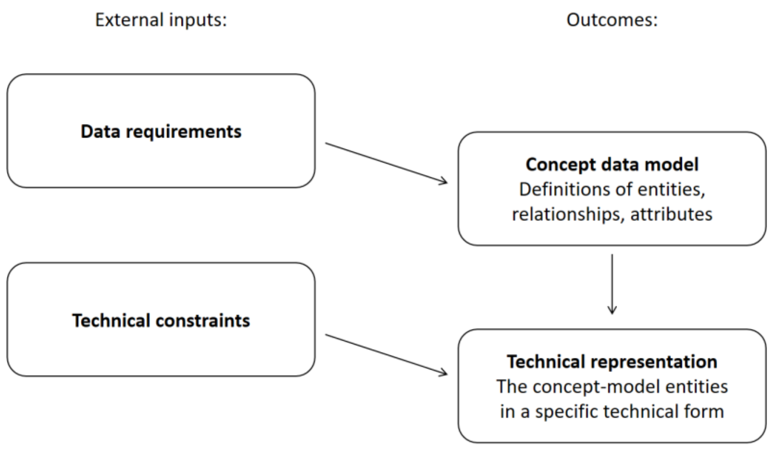
External inputs:
Outcomes:
Data requirements
Concept data model
Definitions of entities, relationships, attributes
Technical constraints
Technical representation
The concept-model entities in a specific technical form

Figure 3 Data modeling approach

Defining technical representation requires choosing a concrete technical data format (for instance SQL, an XML schema, FIWARE NGSI, etc.), i.e. technical constraints. Applying technical constraints to the concept data model gives the technical representation of the data. Technical representation corresponds to the K4+1 development view on data architecture.

In that reference data modeling approach, the technical representation is called a physical data model. In this document, the latter term is not used to avoid confusion with the K4+1 physical view.

2.4 Architecture specification techniques

For explicitness and clarity, this architecture uses the following specification techniques:

- For logical software and platform architectures: informal diagrams and UML Component Diagrams, depending on the required level of detail.
- For logical data architectures: UML Class Diagrams.

3 Requirements for the Architecture

Requirements for design decisions presented in this document are defined by project resources and by the reference architectures that the project follows. These sources are defined in this section.

3.1 Project resources

In general, SHOP4CF combines top-down and bottom-up approaches to the architecture design. Bottom-up means that the software components are already defined and developed to some extent, and they are taken under consideration for the architecture. Top-down means that there is a coherent presentation and interoperability of separately developed functionalities (i.e. the components) and that the scenarios (pilot scenarios, use cases) are defined based on requirements of the pilot partners.

Thus, key requirements for the architecture are the scenarios and the analysis of existing SHOP4CF components. These requirements are based on the prior project results, mainly:

- Deliverable D2.1: Industrial requirements report [7]
- Deliverable D3.1: Functional requirements specification [8]
- Deliverable D5.1: Definition of the deployment scenarios [1]
- (internal) Task 3.2 Component Analysis Report [9].
- Documentation of existing component interfaces [10] – online resource prepared by WP3 and WP4 to detail the high-level data flow derived from the above deliverables.

3.1.1 Scenarios

Deliverables D2.1 and D3.1 defined requirements and designed first pilot scenarios (use cases). These have been taken forward by WP5, and then detailed and extended scenarios were reported in Deliverable D5.1.

Thus, Deliverable D5.1 contains the most up-to-date scenarios together with the mapping to involved SHOP4CF components that are the input for this architecture. They are five scenarios defined at the pilot factories: Arcelik, Bosch (two scenarios), Siemens, and Volkswagen.

3.1.2 Component analysis

Originally, the SHOP4CF description of work contained the development of 24 technical components. Due to various reasons, including shift in the scope of some pilot use cases and new technological developments, adjustments to the originally planned developments were needed, resulting in 25 components presented in Section 5.

The initial information on the components was collected from component developers (partners involved in WP4) via a detailed questionnaire in month 6. That questionnaire was developed by project partners TUM, DTI, TUE, PSNC and TECNALIA as a part of Task 3.2 and Task 2.2.

The detailed description of the components capabilities, together with the analysis of their technical features and mapping to pilots scenarios is available in the Component analysis report [9]. That report concludes activities of Task 3.2, which ended in month 6. Therefore, the first version of the report is based on the components status at that time. The second version of the Component analysis report has been updated by the advancements available in month 11. The revision has been done by incorporating the direct feedback from component developers, their inputs to the documentation of component interfaces [10], and conclusions of recent meetings of Task 4.1. The second version serves as an input to the further analysis in Task 3.3 and for this document.

The initial mapping of the components to the functional landscape in Section 6.3 is also based on the questionnaire's responses. Due to early stage of project development at that time, neither the pilot scenarios nor the component functionalities were fully described. Therefore, the component developers provided information broad enough to allow potential alterations in components capabilities, as those changes would have been required to address particular needs of use cases. As the result, some components covered more than one phase and/or level of the high-level logical software architecture described in Section 6.1. With a better comprehension of components functionalities and their fit to the logical architecture, the updated mapping was delivered in the second version of the Component analysis report.

3.2 Reference architectures

With the advent of the a-priori forecasted fourth Industrial revolution (Industry 4.0), the need of exchanging data among processes and systems has become more and more relevant. Its importance has been moved by the necessity of highly connected Cyber-Physical Systems able to share knowledge among different steps in the manufacturing value chain [11]. However, considering the plurality of vendors and integrators, a need for standardization has followed the development of Industry 4.0 concepts worldwide. The outcome of these standardization processes has been encapsulated in Reference Architectures (RA). RAs propose and give solutions for enabling service manufacturing through automation, facilitated data exchanges and digitization [12] [13]. Therefore, guaranteeing facilities to cope with highly volatile market demands [14].

3.2.1 ISA-95

The International Society of Automation 95 (ISA-95) standard is a set of reference documents meant to bring homogeneity in the communication processes among business and manufacturing activities [15]. The standard is being used since 2000s and is largely used when establishing communication channels (i.e. interfaces) among different activities as highlighted above. The standard proposes nomenclature and modelling approaches that need to be used when exchanging information, therefore allowing alignment among different stakeholders [16]. Due to its wide adoption, the multi-part standard has been published as international standard in the IEC 62264 [17], and it constitutes the basis of nowadays reference architectures. Therefore, alignment in the SHOP4CF is highly considered for guaranteeing interoperability of the solutions. See Section 6.1 and 8.6.

3.2.2 RAMI 4.0

The Reference Architecture Model Industrie4.0 (RAMI4.0) is an outcome of the German initiative Platform 4.0 [18] that was published through the German standardization body DIN (Deutschen Instituts für Normung) with the DIN SPEC 91345 [19]. The model is structured in three dimensions to represent the several facades of an enterprise in the digitized word.

The three axes of the RAMI4.0 RA are layer, hierarchy levels and life cycle. The former, represents the different layers in an enterprise and allows the inclusion of digital twins through the concept of administrative shell. The second represents the hierarchy levels of an enterprise and has been created taking as reference international standards well known in the industry. Finally, the latter represents the life cycle of a product from development to production again taking as reference an international standard.

Through the integration of services and processes in the architecture, using the nomenclature and structure of the standard, it is possible to build solutions that are highly compatible with digitized processes, therefore enabling I4.0 participants to easily exchange data where needed. Due to its importance as one of the first RA for Industry 4.0, SHOP4CF considers the RAMI4.0. See Section 6.1.

3.2.3 FIWARE Smart Industry

FIWARE Smart Industry is a reference architecture and a specialization of the FIWARE framework for "smart factories" [20]. It is defined as the architecture diagram presented in Figure 4.

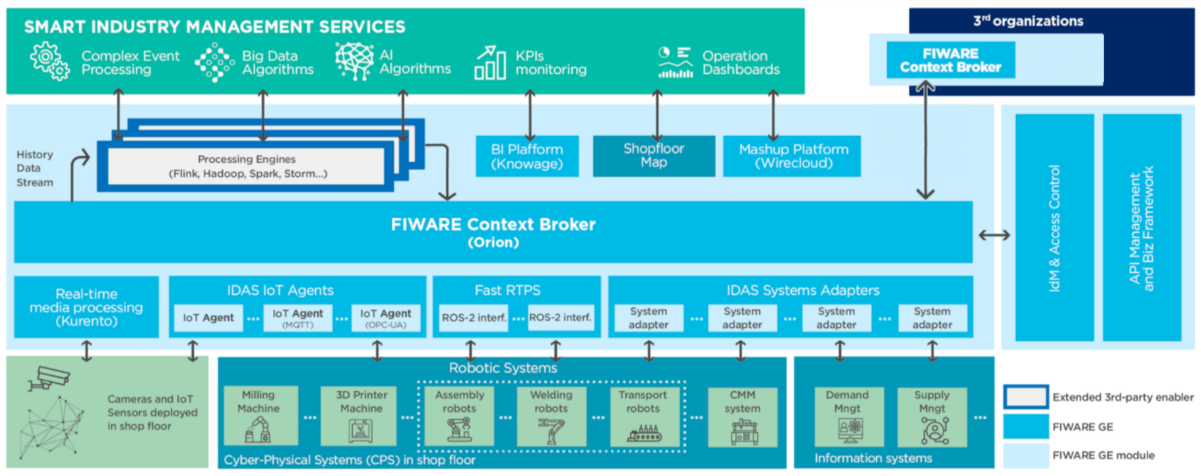

Figure 4 FIWARE Smart Industry (source: www.fiware.org)

FIWARE Smart Industry was selected as a base for the SHOP4CF architecture in the project's description of work.

FIWARE Smart Industry addresses two UT5 aspects: platform and software. They are further discussed, together with their mapping to SHOP4CF, in Section 10.

3.2.4 International Data Spaces

The International Data Spaces (IDS) reference architecture has been developed for guaranteeing data exchange across different entities and enterprises [21]. In its purpose, IDS relates to the UT5 data aspect. However for operation, it is implemented within the platform aspect, as further discussed in this section.

Due to the complexity of the matter, the architecture is structured with three perspectives: security, certification and governance. Moreover, due to the need to find commonalities among several stakeholders the architecture compromises five layers to integrate the different views (i.e. Business, Functional, Process, Information and System) and they all need to satisfy those perspectives. The IDS Reference Architecture Model (RAM) defines the interfaces, the information model and the roles to ensure data sovereignty in the open, federated marketplace [22]. It also structures the way to enforce the usage policies, which can be defined uniquely for every dataset. Finally, the RAM also specifies the rules and mechanisms for data traceability and identification of data sources. The IDS standard enables open, transparent and self-determined data exchange and is a central element of the GAIA-X architecture, which is providing the infrastructure for secure, trustworthy data sharing [23].

Figure 5 shows a simplified version of the IDS architecture. In IDS, different actors (data providers and consumers) can become part of the Data Space by implementing an IDS connector. The IDS connectors can exchange data (including its meta data) while making sure that the information model is standardized, and the usage policies are enforced. All meta data and connection information is stored in the broker (top of Figure 5). The identity providers know which connectors are installed in which company.

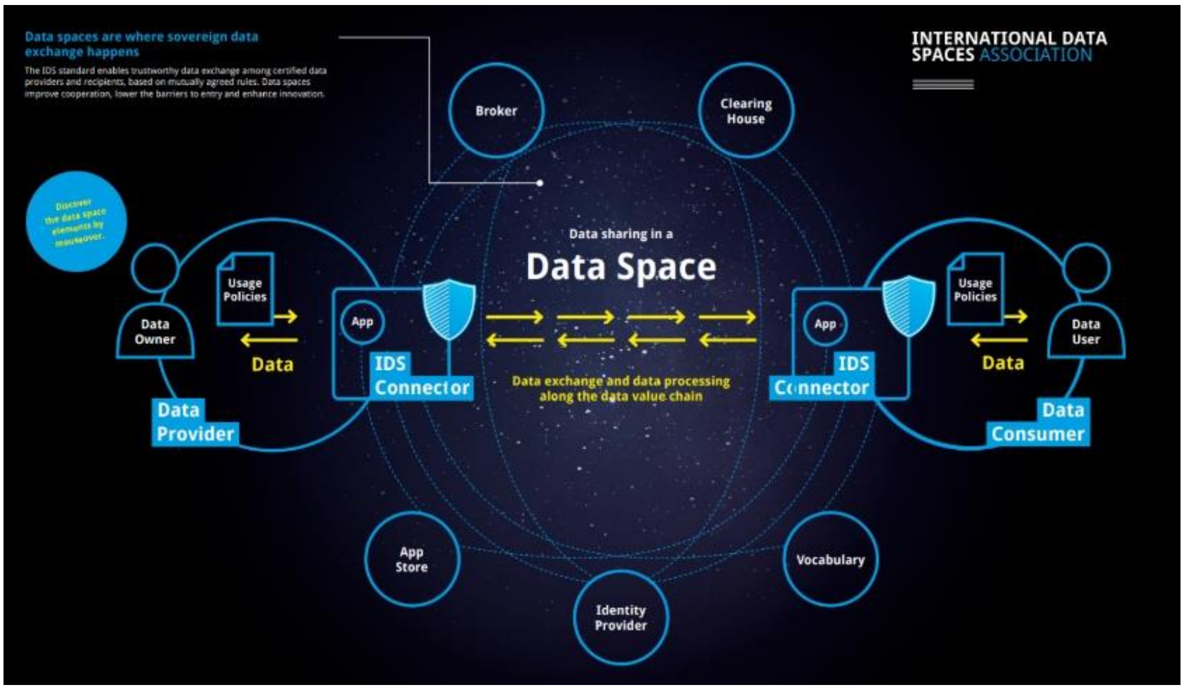
Data spaces are where sovereign data exchange happens
The IDS standard enables trustworthy data exchange among certified data providers and recipients, based on mutually agreed rules. Data spaces improve cooperation, lower the barriers to entry and enhance innovation.
INTERNATIONAL DATA SPACES ASSOCIATION
Broker
Clearing House
Discover the data space elements by mouseover.
Data sharing in a
Data Space
Data Owner
Usage Policies
Data
Data Provider
App
IDS Connector
Data exchange and data processing along the data value chain
App
IDS Connector
Usage Policies
Data
Data User
Data Consumer
App Store
Identity Provider
Vocabulary

Figure 5 IDS architecture (simplified) (source: internationaldataspaces.org)

To Better understand how the IDS works it is important to reference to some existing projects in the space. In particular, the Catena-X project [24], initially sponsored by the German federal government, is of well relevance for the sharing of data. Catena-X is a project which tries to answer the needs of building circular economies by enabling the traceability of materials in the automotive sector. This traceability is pursued by ensuring that companies participating in automotive supply chain (i.e. manufacturers, suppliers, and service providers) share digital traces (i.e. data) to the materials. However, most of these companies are unsure about the real added value of this practice. Therefore, they are reluctant to these practices. To avoid these barriers, thus Catena-X puts in place proper digital tools that allow the sovereign control of data which enable the creation of a trustworthy, open, standardized and certified data space (i.e. digital place where data can be shared).

In the SHOP4CF project, the IDS could be used for integrating lifecycles of data and, due to an ongoing alignment of IDS with the FIWARE Smart Industry, it could be easily achieved. Therefore, SHOP4CF, by using the FIWARE Smart Industry with proper connectors, can support integrations also with IDS.

In the case of SHOP4CF an interesting approach, following the example of Catena-X, can be the traceability of manufactured goods produced using one or more SHOP4CF components. To better understand this possibility, it is good to refer to the existing approach in Catena-X. In the project they aim to create the traceability across the value creation of a vehicle. Therefore, each step incurring during the production is recorded thus allowing the creation of continuous data chains. Through this data chains, it is then possible to calculate sustainability impact key performance indicators (KPIs). In Catena-X, for example, they aim to track the amount of CO_2 emitted during the value creation of a vehicle as shown in Figure 6 [25].

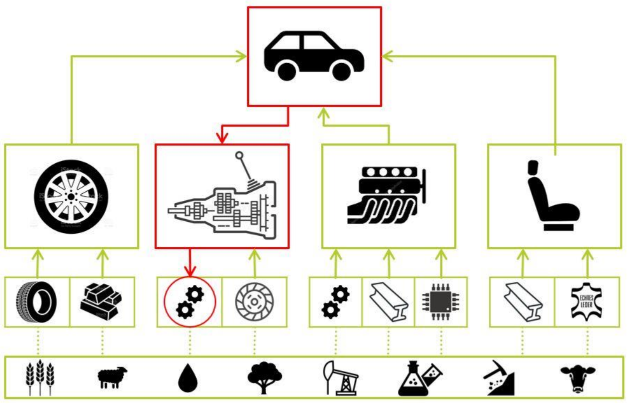

Figure 6 Simple explanation of the traceability in the vehicle value creation (source: catena-x.net)

Therefore, in SHOP4CF, considering that the FIWARE data model is already standardized, the data created can be easily shared in a data space. In this way, SHOP4CF-produced goods can be easily used in use cases where traceability is necessary. Thus, allowing a better reach of the components in different domains.

3.2.5 Architectures of prior research projects

SHOP4CF builds also on the architectures of other research and innovation projects in the European Union (EU) Horizon 2020 program: HORSE and L4MS.

In 2015-2020, **the HORSE project** designed, developed, deployed and tested the HORSE framework, which is a reference architecture for cyber-physical systems that support hybrid-manufacturing processes in the IoT context. The framework is a modular architecture that serves as a blueprint for building solutions for enterprises in the discrete manufacturing domain, towards their goal to integrate robotics safely in their end-to-end operations.

The HORSE architecture [26], from the functional high-level perspective, distinguishes between manufacturing activities taking place in a work cell and activities in a production area or even site (across work cells). This distinction is depicted with two levels, the Global and Local. There is also a clear distinction of phases, one regarding design of manufacturing activities (e.g. modeling, parameterization) and one regarding executions of manufacturing activities (actual product manufacturing), i.e. the Design and Execution phases.

The L4MS project is a four-year project (2017-2021) to become a one-stop shop for manufacturing SMEs to help them digitalize intra-factory logistics. This is achieved by Open Platform for Innovations in Logistics (OPIL) – an open IoT platform with different enablers with the common aim of simplifying the development of customized logistics solutions.

The OPIL Reference Architecture [27] consists of three layers. The IoT Nodes Layer (L1) is components that interact with the physical world using well-established technologies such as ROS or AGV. The Cyber Physical Middleware Layer (L2) adopts FIWARE and allows interoperability among components of the platform and with external ones. The Software Systems Layer (L3) are software components developed specifically for the logistics sector.

4 Methodology applied to SHOP4CF

This section explains how the methodology and frameworks described in Section 2 apply to the SHOP4CF architecture. In particular, it shows how the architecture is positioned within the complete design of the SHOP4CF project, i.e. what concerns are in the scope of this document, and what others are left for other work packages and reports.

4.1 The UT5 framework applied to SHOP4CF

The software in the scope of the SHOP4CF project can be divided into two categories:

1. Software directly supporting manufacturing execution (including also its design or configuration) in working cells, lines, and factory sites. This is the SHOP4CF components under development in WP4.
2. Software supporting the SHOP4CF Marketplace and the interaction between factories and the marketplace (e.g. the deployment of components based on the marketplace).

This document focuses only on software category no. 1. Whenever this document refers to software, category no. 1 is meant. Software category no. 2 is in the scope of WP6.

By applying the UT5 framework to the SHOP4CF project, it becomes structured what kind of aspects need to be considered and which of them are addressed in this document:

- The **software, platform, and data aspects** are addressed in this SHOP4CF architecture. They are elaborated in this document to certain extent, as explained in the next section.
- The process and organization aspects are not covered in this document as they usually depend on a specific organization, for which a concrete SHOP4CF system is designed. For the SHOP4CF pilots, these aspects are addressed in prior Deliverables D5.1 [1] and D3.1 [8].

4.2 The K4+1 framework applied to SHOP4CF

This document focuses on **the logical view** of the K4+1 framework, i.e. the structure of the application logic in abstract terms focusing on functionality. The logical view on the aforementioned aspects – software, platform, and data – is designed (Sections 6-9). These designs are referred to as logical software/platform/data architectures.

Moreover, software developers that are (among others) recipients of this document need also a unified development specification of interfaces for their work on SHOP4CF components. Thus, this need is also addressed by designing **the development view** (technical representation) of data (i.e. FIWARE representation, Appendix B).

The scenarios are not designed in this document, but it refers to the use cases defined in the prior project reports as explained in Section 3.1.

The other K4+1 views are not designed in the SHOP4CF architecture. As presented in Figure 2, the other views are further specifications of the logical view, and those perspectives are considered by other project teams:

- The development view on particular software components is considered in Task 4.1: “Development of the components” by specific component developers.
- The process view is considered in Task 4.3: “Continuous integration”.
- The physical view is considered in Task 5.2: “Pilot deployments”.

This document uses informal terms to refer to the level of detail of designs or views: high level and medium level. The lower level the designed models are more detailed. The document mainly refers to high-level views (Part 2, Sections 5-8) and medium-level views (Part 3, Sections 9-10). As various presented concepts could be further decomposed, low-level views could be also defined. However, they are out of scope in this document. In addition, the top level is referred to in some cases, and it should be understood as a part of the high level and the most general design for a specific UT5 aspect.

4.3 Data modeling approach applied to SHOP4CF

One of the focuses of this document is interoperability among SHOP4CF components. Internal data structures within particular components are out of scope, as they are addressed for particular cases by the component developers in Task 4.1. Thus, the data modeling process, defined in Section 2.3, focuses on information exchanged between the components.

The process has been applied as follows. The subsequent steps of the data modeling approach from Figure 3 are in bold.

The **data requirements** were derived from the scenarios described in Section 3.1. Expected connections between SHOP4CF components together with characteristics of data to be exchanged were analyzed, documented as a project internal report about data flows in pilot scenarios [28], and validated by particular component developers.

In addition, the requirements were supplemented by the concept data models of FIWARE [29] and of the ISA-95 standard (Section 3.2.1) to conform to the latter and to reuse the former, when it makes sense.

Based on the requirements, similar kinds of data to be exchanged were grouped into more general data entities, resulting in designing the SHOP4CF **concept data models** (the high-level logical data architecture, Section 8). The experience from the prior project HORSE helped in designing the data models [26].

As the middleware platform supporting the interoperability is FIWARE (Section 7), the platform documentation defines the **technical constraints** for data in SHOP4CF. As explained in the previous section, the K4+1 development view in this architecture focuses on FIWARE representation only.

Finally, the **technical representation** (FIWARE) of SHOP4CF concept data models is discussed (Appendix B).

5 Overview of SHOP4CF components

SHOP4CF components directly provide main functionalities considered within the SHOP4CF architecture. The analysis leading to this overview is reported in Section 3.1.

Table 1 contains the overview of the SHOP4CF components. The component acronyms has been proposed by partner TU/e in Task 3.2 in cases they were not defined by component developers.

Note that this overview will be extended in the course of the project. The component developers may propose new components as solutions for use cases that are not yet fully defined. Moreover, it is expected to acquire a new set of components as a result of the open call programme.

Table 1 List of SHOP4CF components with short descriptions

Acronym	Name	Application area / Main function	Developer
ROS-Mon	ROS Monitoring	Robot system status monitoring to support workers	DTI
WPO-RL	Workcell Process Optimization based on Reinforcement Learning	Process control and optimization based on reinforcement learning	DTI
DTS	Dynamic Task Scheduling for Efficient Human-Robot-Collaboration	Task manager for safe and efficient human-robot interaction, by distributing robot-tasks to sub tasks, tracking their status, preventing human-robot collision	FZI
FBAS-ML	Force-Based Assembly Strategies for difficult snap-fit parts using Machine Learning	Supporting human workers with a force-sensor (force-control) on classical industrial and/or collaborative robots to fit two or more parts together that require a snap connection	FZI
F-TPT	Flexible Task Programming Tool	Programming of robots with GUI to quickly develop or change new control sequences, monitoring also status feedback.	FZI
ASA	Automated Safety Approval	Determining whether the chosen robot speed is safe and the required separation distance has been chosen and can be covered by the sensor configuration	IFF
RA	Review of Risk Analysis	Risk analysis for hazards identification and risk estimation of robotic applications	IFF

Acronym	Name	Application area / Main function	Developer
OpenWIFI	Open-source implementation of 802.11 WIFI on FPGA	Low latency network connectivity between WiFi-enabled devices for real-time control to support process management, interactions with robots, collecting sensor data	IMEC
FLINT (formerly M3RCP)	Multi-Modal Multi-Range Communication Platform	Facilitation of the incorporation of IoT devices (sensors/actuators) in a factory shop floor, as well as the required local wireless IoT communication infrastructure to connect such devices	IMEC
Wi-POS	Wireless Positioning system based on UWB technology	Safe and controllable usage of AGVs by providing accurate localization.	IMEC
HA-MRN	Human Aware Mobile Robot Navigation in Large-Scale Dynamic Environments	Mobile robot navigation with human detection and trajectory adaptation according to safety and social rules	JVERNE-FZI
H-ZoneS	Human Zone Scheduler	Component integrating a scheduler and a Flexsim simulation template to generate and validate schedules considering zone occupation and accessibility constraints	JVERNE
IL-DT	Digital Twin for Intralogistics	Automatic building of digital twin based on simulation model to solve intralogistics challenges depends on analysis level	PSNC
PMADAI	Predictive Maintenance and Anomaly Detection in Automotive Industry	Prediction or prevention of potential failures and incidents. Planning of services and repairs	PSNC
VQC	Visual quality check for automatic paint defect detection in Autom. Ind.	Quality monitoring. Detection (and potentially classification) of car paint defects	PSNC
VR-RM-MT	Virtual Reality Set for Robot and Machine Monitoring and Training	Training and supporting human workers in collaborative tasks through remote visualization and monitoring	TAU
M2O2P	Multi-Modal Offline and Online Programming solutions	Online/Offline robot programming using input methods based on human natural actions	TAU
DCF	C2NET Data Collection Framework	Data collection from the factory shop floor and ERP systems.	TAU

Acronym	Name	Application area / Main function	Developer
		Analysis of process and data streams using a Complex Event Processing (CEP) engine	
DT-PC	Digital Twin (Planning and Control)	Remote visualization, performance monitoring, and control of discrete processes at runtime	TAU
ADIN	Adaptive Interfaces	Adaptation of interfaces depending on the information collected from production line devices and the user's profile, skills, and roles within the system	TAU
AR_Manual_Editor (AR_Man_Edit)	Augmented reality-based manual editor	Mixed Reality (MR) Component Simulator for operator training in customized product assembly process, including recognition of objects, sequence of operations and AR guidance to operators.	TECNALIA
AR_Teleassistance (AR_Teleassist or AR Content)	Augmented reality-based teleassistance	Communication between workers and experts through video streaming and augmented reality indications supporting operators with the maintenance and collaboration of working processes.	TECNALIA
VR_Creator	VR_Creator	Virtual Reality (VR) to assist workers on training on machines	TECNALIA
MPMS	Manufacturing Process Management System	End-to-end manufacturing process management, i.e. design, enactment and orchestration of manufacturing processes, with dynamic agent allocation, exception handling and process monitoring	TUE
AR-CVI	AR for Collaborative Visual Inspection	Visual support to humans in inspection tasks	TUM
WoT-IL	Interoperability Layer through Web of Things	Translation of OpenAPI specification into Web of Things (WoT) "Thing Description" to improve interoperability	UPM

6 High-level logical software architecture

This section presents the logical view (as in K4+1) on the software aspect (of UT5), i.e. the organization, from the functional perspective, of the SHOP4CF components.

The following subsections present the functional overview of SHOP4CF, abstract interfaces of this high-level architecture, the positioning of particular SHOP4CF components, and interoperability defined for the pilots.

6.1 Top-level logical software architecture

Manufacturing processes can be supported in their different **phases**: at design of the processes, at their execution (i.e. actual product manufacturing), and at further analysis of (the data resulting from) the execution. Moreover, processes can be supported at different manufacturing **levels**: within specific work cells and across work cells. These levels are referred to as the local level and the global level, respectively. Work cells are defined as in the ISA-95 equipment hierarchy model or as station in the hierarchy levels dimension of RAMI4.0 (see Section 3.2).

SHOP4CF addresses all these phases and levels by providing relevant functionalities. Thus, the top-level overview of SHOP4CF functionality (i.e. the logical software architecture) consists of the six subsystems, as presented in Figure 7. At this level of detail, it is not yet considered how the subsystems interact with each other and with the external world.

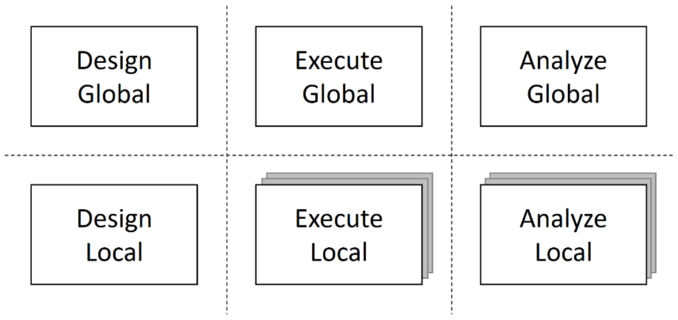

Figure 7 Top-level logical software architecture

The SHOP4CF Architecture Workshop [3] defined the levels and phases in detail as follows:

The architecture has two levels. Conforming to the HORSE architecture, the lower level is the local level and the upper level is the global level:

- *The local level provides functionality to support for individual work cells, for example augmented reality support with a robot.*
- *The global level provides functionality across individual work cells, for example end-to-end manufacturing process support.*

The architecture has three columns:

- *The left-hand column provides support for designing intra-work-cell (local) and inter-work-cell (global) manufacturing environments. Note that this does not mean designing software; it means parameterizing software to suit the needs of a specific shop floor.*
- *The middle column provides support for the execution of the actual manufacturing process, i.e. the material transformation process. From a logical point of view, there is one copy of the global subsystem and a copy of the local subsystem per work cell (although physically, these copies may be the same system). (…)*
- *The right-hand column provides support for analysis of data resulting from system execution in the middle column for both real-time monitoring of production processes (e.g., via a dashboard) and optimization of either execution (i.e. without explicit redesign) or design (i.e. with explicit redesign). Support includes 'traditional' business intelligence, but also manufacturing-specific functionality like digital twins.*

The Execute Local and Analyze Local subsystems are replicated (shown in the figure by a 'stack of boxes') because from a logical architecture point of view, every manufacturing cell has its own instance (copy) of these subsystems. In a physical architecture, these can be mapped to a single component per site that keeps track of the state of each individual cell.

Categorizing whether a software component belongs to the global or the local level is not always trivial; for instance for a component supporting a few work cells. If a component provides functionality to specific work cells, without being aware what is happening outside of these work cells, then such a component is considered local. Otherwise, it is a global-level component.

6.2 High-level interfaces

The high-level logical software architecture from Figure 7 is further detailed by elaborating the interfaces between the six subsystems and designing databases that support the communication, as presented in Figure 8. This high-level overview was first designed in SHOP4CF Architecture Workshop [3], and is inspired by the design of the HORSE logical software architecture [26].

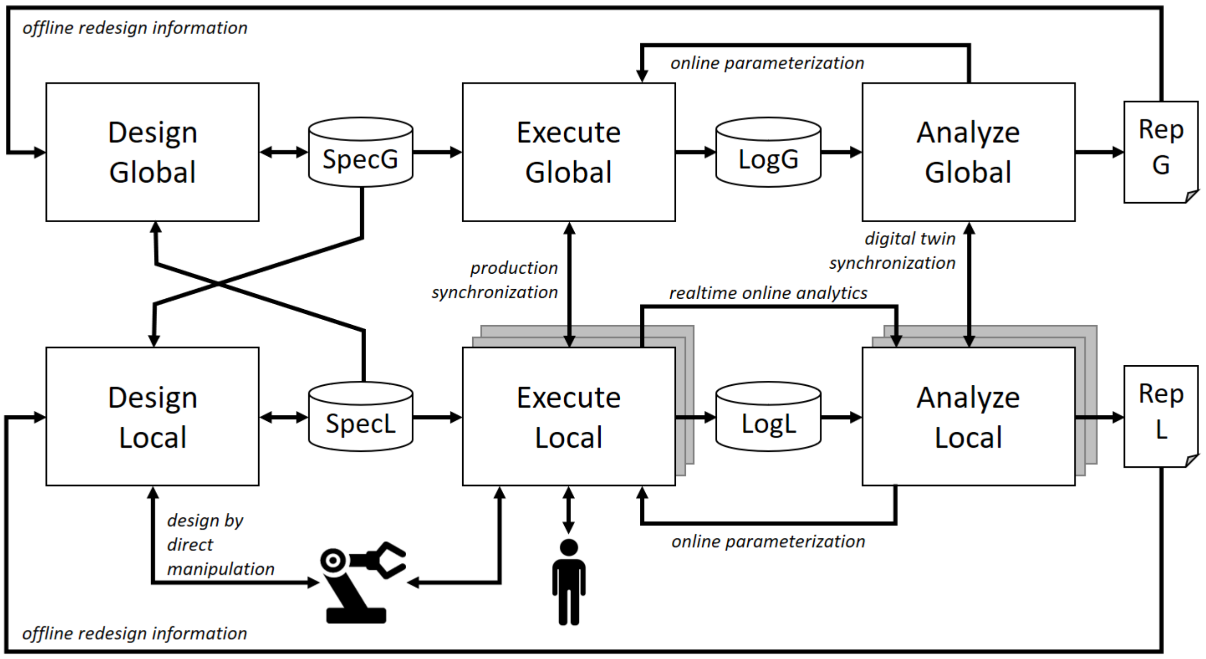
offline redesign information
online parameterization
Design Global
SpecG
Execute Global
LogG
Analyze Global
Rep G
production synchronization
digital twin synchronization
realtime online analytics
Design Local
SpecL
Execute Local
LogL
Analyze Local
Rep L
design by direct manipulation
online parameterization
offline redesign information

Figure 8 High-level logical software architecture with interfaces

That workshop defines these interfaces as follows:

> *The architecture contains direct and indirect interfaces between the six subsystems. The indirect interfaces are modeled as databases. Interaction using these interfaces is asynchronous and does not have a real-time character. The direct interfaces are modeled as direct connections. These connections can be synchronous (i.e. requiring an instant reaction to requests) or asynchronous (i.e. not requiring an instant reaction to requests or no reaction at all). The direct connections have a real-time mode of operation, which can be soft-real-time (e.g., for inter-cell synchronization) or hard-real-time (e.g., for digital twins used for safety reasons).*

This diagram presents also that the Execute Local subsystems interacts with human workers performing actual manufacturing steps. The Design and Execute Local subsystems interact with robotic systems to, respectively, design and execute manufacturing steps. Note that this diagram does not depict all possible dashboards (user interfaces) that may exist in different subsystems and for different kinds of users.

The information exchanged between the six subsystems and the databases is detailed in Appendix A.

Section 7.4 further discusses the interfaces from the perspective of the platform architecture, and Section 8.1 addresses the interfaces from the perspective of the data architecture.

6.3 Positioning of SHOP4CF components

The six subsystems from Figure 7 are decomposed into particular SHOP4CF components. Thus, all the SHOP4CF components are mapped to the logical software architecture, such that an initial “functional landscape” of the software under development is obtained. The process leading to this result is reported in Section 3.1.

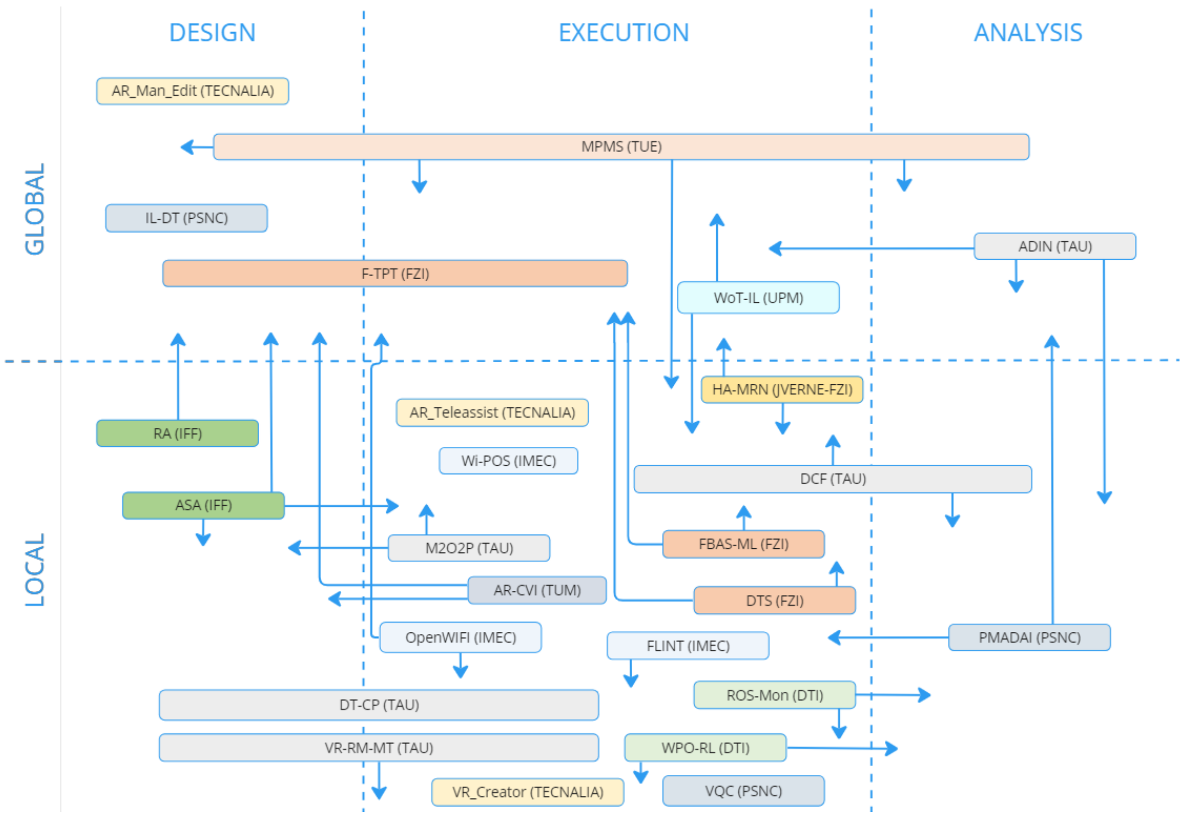

Figure 9 Mapping of SHOP4CF components to the high-level logical software architecture

The mapping was then updated in the fourth year of the project, based on the functions of the components actually used in the pilots [2]. Figure 9 depicts this mapping.

Components are indicated as rectangles with their acronyms, and with their responsible component developer in parentheses. For visualization purposes, the rectangles are colored the same as in the Table 1. Arrows indicate the (horizontal or vertical) integration of a component with other components positioned in the pointed phase and level of the diagram. This creates the overview of all integration possibilities between components that is foreseen in the current stage of the project.

6.4 Mapping to scenarios

Figure 10 presents interoperability among components that is specific to the actual pilot scenarios [2]. This serves as a robust example on how a concrete SHOP4CF system could be designed from the functional perspective.

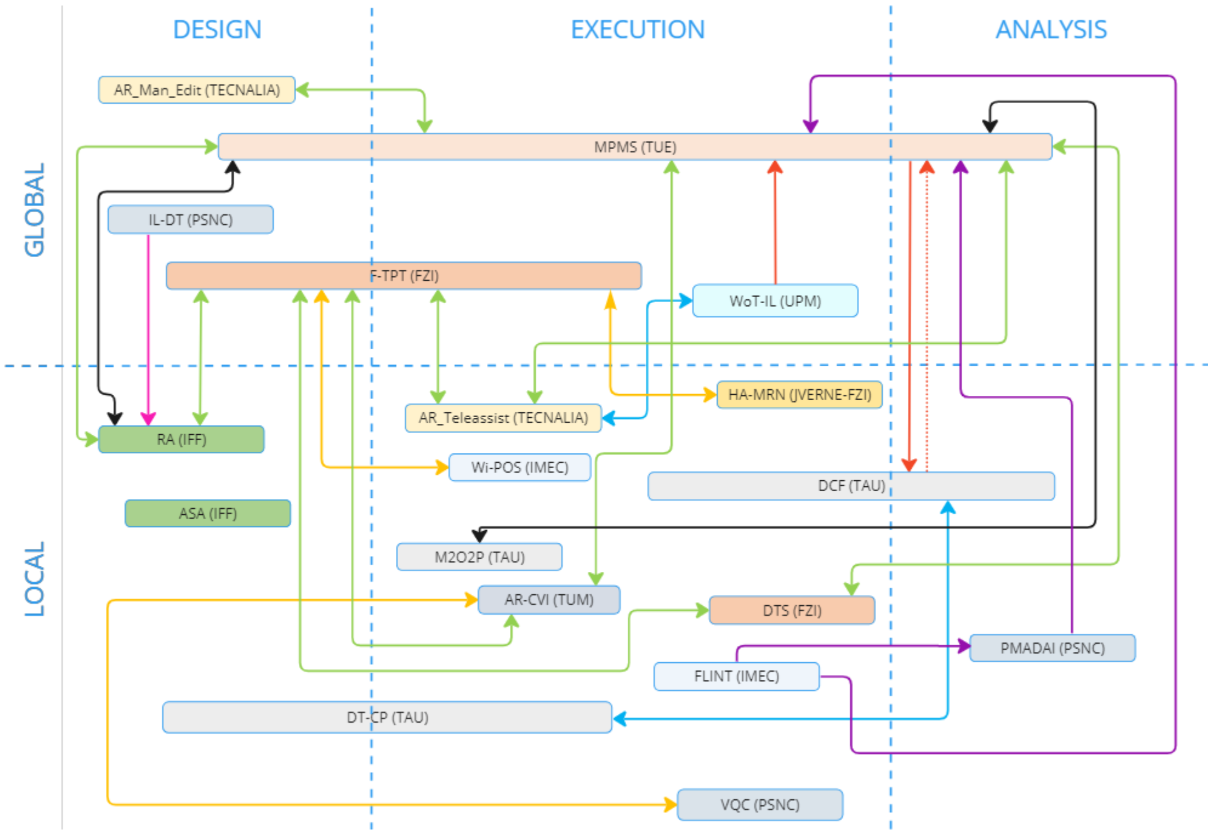

Figure 10 Interoperability among components in the pilot scenarios

SHOP4CF components that were not assigned to any of the use cases are omitted in this diagram. In addition, components, for which it was not defined whether they integrate with other SHOP4CF components within the use cases, are also omitted.

Communication within each of the use cases is depicted with the following colors:

- Arcelik – blue,
- Bosch Use case 1 – orange,
- Bosch Use case 2 – red,
- Siemens Use case 1– green,
- Siemens Use case 2 – black,
- Volkswagen Use case 1 – purple,
- Volkswagen Use case 2 - pink.

Exact routes of the arrows (going through other subsystems) do not matter in this diagram.

Moreover, the dashed arrows indicate the integration between components that, at the first revision of this deliverable, was still under consideration and under discussion in Task 5.2: “Pilot deployment”.

7 High-level logical platform architecture

This section presents the logical view (as in K4+1) on the platform aspect (of UT5), i.e. the organization, from the functional perspective, of software and hardware that is necessary for the components from the software architecture to work. The middleware is especially considered as well as its relation to the software architecture.

7.1 Top-level logical platform architecture

The top-level logical platform architecture of SHOP4CF is presented in Figure 11. It consists of the software and hardware layers.

The software layer consists of SHOP4CF components, the middleware, containers (i.e. OS-level virtualization [30]), and 3rd-party information systems (i.e. external to SHOP4CF, such as MES) that may exist in a shop floor. The hardware layer consists of servers. In addition, cyber-physical systems and IoT devices of a shop floor may belong to the both layers.

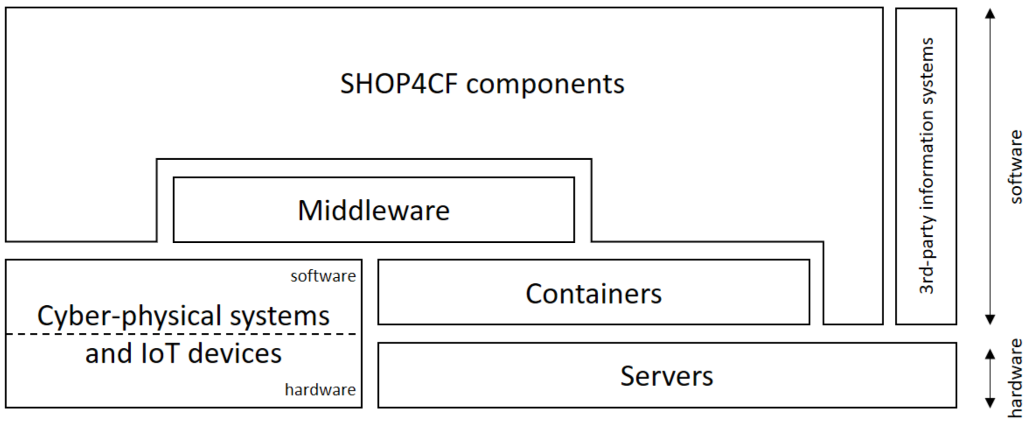

Figure 11 Top-level logical platform architecture

Vertical adjacency depicts connections between platform components. SHOP4CF components connect to the middleware. The middleware connects to cyber-physical systems and IoT devices but also some SHOP4CF components can directly connect to those. Both SHOP4CF components and the middleware run in containers. In addition, some SHOP4CF components can run directly on bare servers, without containers, in justified cases.

Moreover, some SHOP4CF components connect to 3rd-party information systems. Other connections of the 3rd-party systems are out of scope and are not necessarily depicted in the diagram.

Containers are used to make software components easy to deploy and control. The chosen implementation for containers is Docker [31].

The three platform components – Cyber-physical systems, Middleware, and SHOP4CF components – correspond to the three layers of the OPIL Reference Architecture (Section 3.2.5).

7.2 Overview of FIWARE middleware

For interoperability of SHOP4CF components, the SHOP4CF architecture focuses especially on the middleware. Middleware can be defined as “software glue” [32], i.e. software providing services (for instance for exchange of information) to functional software components (for instance SHOP4CF components). Thanks to the middleware, functional components do not care about architectures and connections to other components.

The middleware component from Figure 11 must be decomposed to show how it supports the software architecture. To decompose the middleware from the perspective of its functionality (i.e. the logical view of K4+1), the adopted implementation of the middleware is first discussed.

The chosen implementation for the middleware is FIWARE [33]. FIWARE focuses on management of context information, i.e. the current state of the surrounding real world, understood as the state of relevant physical and virtual objects (for instance, a virtual object may be a manufacturing task to be executed). The use of context information helps to develop what is referred to as a “smart factory”.

SHOP4CF components exchange information via the FIWARE middleware whenever possible. Only connections that have hard real-time constraints are organized directly between two involved components (or between a component and IoT) as the FIWARE middleware does not guarantee response times for hard real-time systems [34].

7.3 High-level middleware architecture

The top-level logical platform architecture from Figure 11 is further elaborated by decomposing the middleware, as presented in Figure 12 and discussed below. Note that the focus is on the middleware, thus SHOP4CF components are treated as a black box and their other connections are not considered.

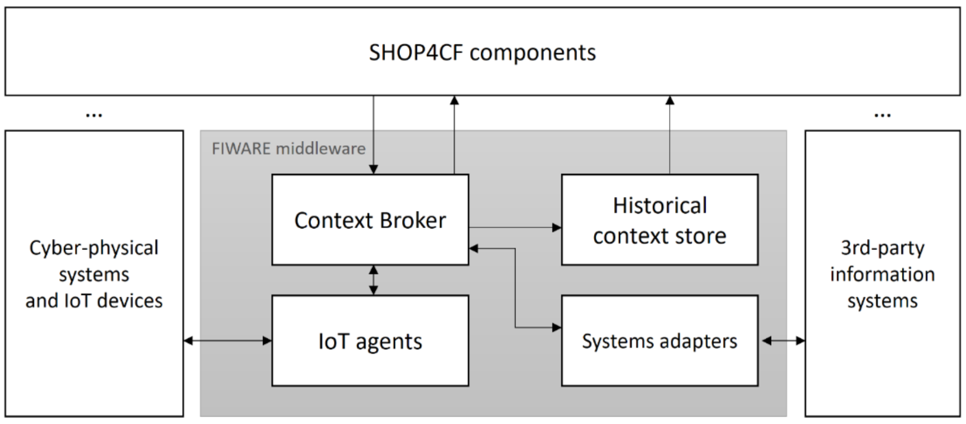

Figure 12 High-level logical middleware architecture

The core component of FIWARE is the **Context Broker** that facilitates exchange of context information. The chosen implementation of Context Broker is Orion-LD [35] and API is NGSI-LD [36].

As the Context Broker only keeps the current state of the real world, FIWARE offers also components for storing historical data [33]. Different implementations exist[1], thus in this document, they are jointly referred to, from the functional perspective, as **historical context store**.

Moreover, FIWARE offers components to interface the Context Broker with cyber-physical systems and IoT devices – these are **IoT agents** – and with 3rd-party systems – these are **Systems adapters**.

Figure 12 could be extended with further middleware components in the future, for instance in case of new use cases that require components to exchange streaming data, for which the Context Broker is not suitable.

The connections from SHOP4CF components to the middleware are further elaborated in the medium-level architectures in Section 9. The mapping to FIWARE Smart Industry architecture, a more detailed FIWARE decomposition, is discussed in Section 10.

7.4 Mapping to software architecture

Figure 13 presents how the middleware components support communication between SHOP4CF components, i.e. how the high-level logical software architecture (Figure 8) maps to the logical middleware architecture.

[1] For instance, FIWARE Cygnus and Comet, but also a few others.

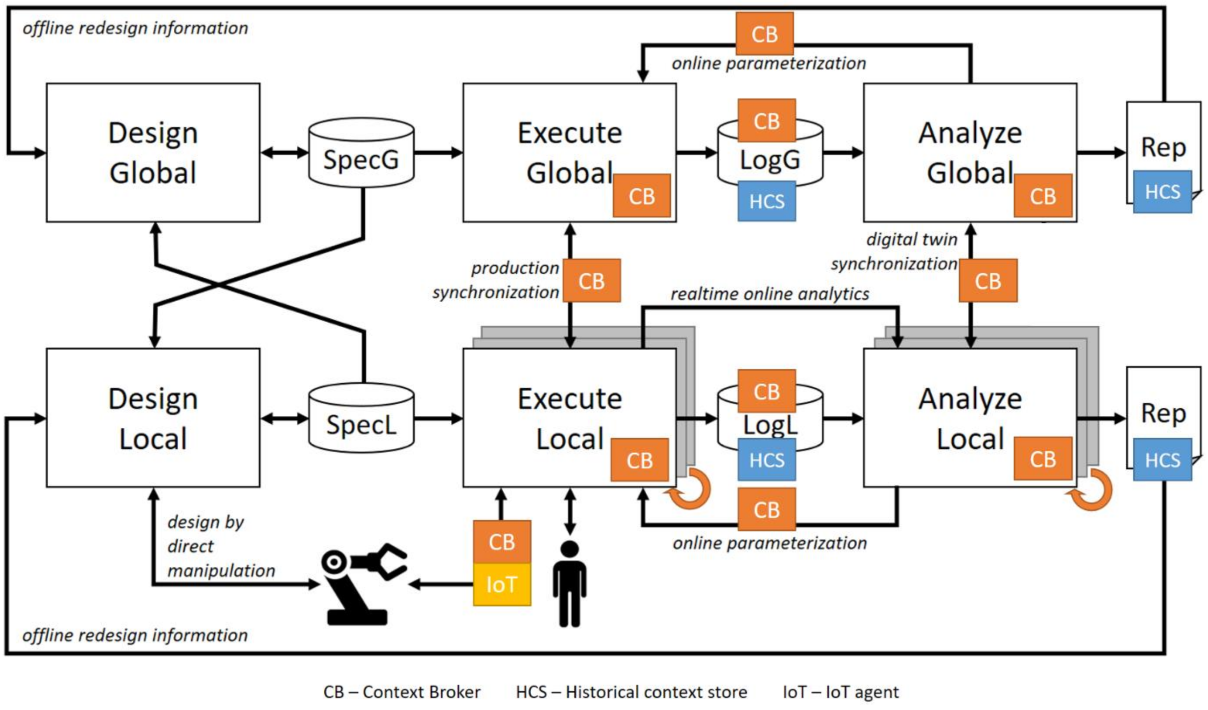

Figure 13 Mapping of high-level logical software and middleware architectures

The Context Broker supports communication:

- between the Execute and Analyze phases (implementing the LogL/LogG databases) and between their Global and Local levels,
- between Execute Local and cyber-physical systems,
- between components within any of the four Execute/Analyze Global/Local subsystems, including communication across instances of the Local subsystems (that are represented by the stacks of boxes with the rounded arrows).

As for some use cases, the current context information is not enough, some communications are handled by the historical context store. This is some part of communication between Execute and Analyze but also from Analyze to Design (implementing the Rep database).

In addition, communication between the two Design subsystems that focuses on static information (e.g. processing models, task definitions) may be handled by other databases, such SQL databases or Business Process Model and Notation (BPMN) files. Feasibility of adopting FIWARE for the Design phase is under consideration. Note however that the Design subsystems already use the information from FIWARE by accessing the historical data stores (via the “offline redesign information” path).

Note that this mapping only expresses that this is a feasible implementation of the connections, but this is not the only way that is admissible by the SHOP4CF architecture.

Note also that this is the logical view (of the K4+1 framework) on the middleware. From the physical perspective, all the multiple Context Brokers and historical data stores in Figure 13 could be implemented by single instances of relevant software applications.

8 High-level logical data architecture

This section presents the logical view (as in K4+1) on the data aspect (of UT5), i.e. the organization, from the functional perspective, of SHOP4CF concept data models. The data modeling process was explained in Section 4.3.

The following subsections present first the top-level data architecture and relevant groups of data models. Next, the lifecycle of selected data models imposed by the FIWARE middleware and their mapping to scenarios are discussed. Finally, the mapping to the ISA-95 standard is reported.

8.1 Top-level logical data architecture

The data models focus on information exchanged among SHOP4CF components. The data models are divided into the two main groups:

- **Design data models** – These are entities that do not change their status during manufacturing execution. They are shop-floor locations, definitions of manufacturing processes, of tasks, etc. and are usually communicated via interfaces no. 1-9 of Figure 42 (the high-level logical software architecture).
- **Execution data models** – These are entities that may change their status during manufacturing execution. They are tangible resources in a shop floor, tasks instances under execution, alerts, etc. They are communicated via interfaces no. 8-22 of Figure 42.

Moreover, execution data models may reference design data models (e.g. a task under execution referencing its definition).

The high-level overview of concept data models in these two groups is presented in Figure 14. These and more detailed data models together with specific relationships are further elaborated in the two following subsections.

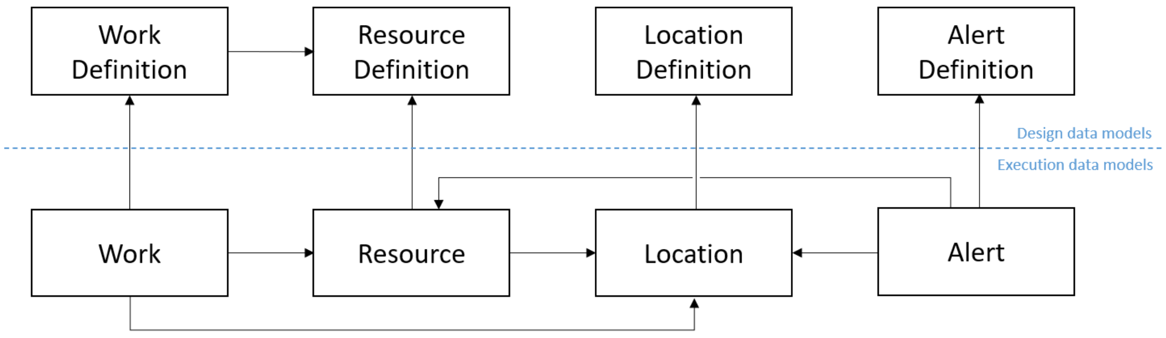

Figure 14 Top-level logical data architecture

8.2 Design data models

The design data models are the concepts of location definitions, work definitions, resource definitions, and alert definitions, as elaborated in the following subsections.

8.2.1 Location Definition

Location Definition represents a specific place in a factory such as work cell, production line, area, site, etc. as in the ISA-95 equipment hierarchy model (see Section 3.2.1).

Locations instantiates location definitions in the execution phase. Location (in execution) may represent its current state (e.g. "temporarily closed").

Location Definition may consist of other Location Definitions. Similarly, Location may consists of other Locations, as presented in Figure 15. This relationship enables modelling hierarchies such as a production line consisting of "work cells".

Locations are referenced by execution data models to represent where something is, happens, is target at, etc.

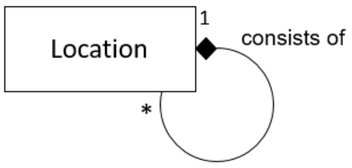

Figure 15 Location data model

8.2.2 Work Definition

Work definitions is **Process Definition** that consists of **Task Definitions** that consists of **Step Definitions**. Step Definition may consist of further Step Definitions (substeps).

These relationships are presented in Figure 16.

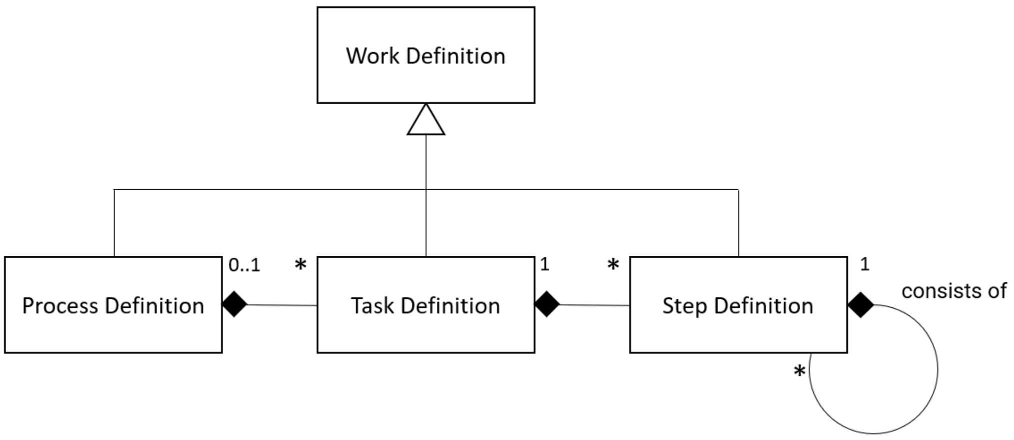

Figure 16 Work definitions data model

Step Definition defines how to perform the work in detail. Step Definition may have attributes such as required skills that denote what skills are required by a resource to perform the step. This may be further extended by concrete needs of specific scenarios.

8.2.3 Resource Definition

Resource Definition (earlier called **Resource Specification**) models a type (a kind or a class) of Resources (see Section 8.3.1), if no concrete Resource instance needs to be known. Resource Definition is used in the design phase to supplement Task Definition, i.e. to represent what kind of Resources are to be used for a certain Task Definition. They can be also used for scheduling, i.e. based on Resource Definition, the scheduler knowns to which Resources it can assign a Task (i.e. which Resources are of the qualified type).

The relevant relations are presented in Figure 17.

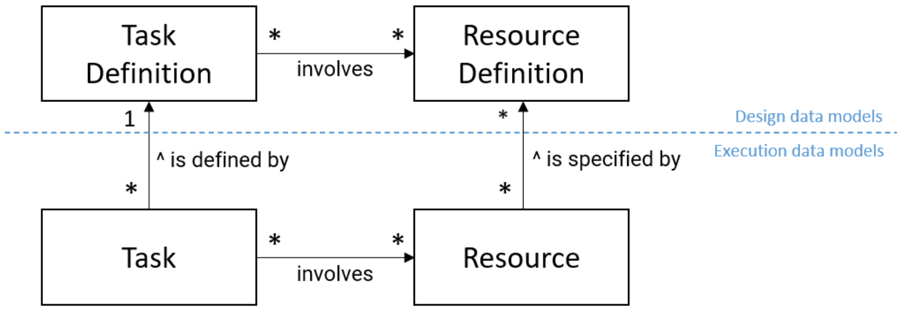

Figure 17 Resource Definition data model

For instance, assume we have Task Definition “assemble a PCB”. This Task Definition involves Resource Definition “worker qualified for electronics”. Three Resources are currently available: AGV #5, worker John Winter, and worker Anna Smith, but only the latter is specified by that Resource Definition. A scheduler is about to assign Task “assemble PCB no. 2342852”. Thanks to the Resource Definition, the scheduler knows that at the moment, the task can be only given to Resource “worker Anna Smith” that is of the qualified type.

8.2.4 Alert Definition

Alert Definition models a type (a kind or a class) of Alerts which may occur and require reactions during manufacturing execution. These are not predictable and not recurring events. A broken device may be (raise) an alert, while a completed work may not.

8.3 Execution data models

The execution data models are the concepts of resources, tasks, processes, and alerts, as elaborated in the following subsections.

Taking into consideration the platform architecture, the execution data models are meant to be exchanged mainly via the FIWARE middleware (see Section 7.3). Thus, to ensure portability and interoperability of SHOP4CF components, the execution data models were defined using Smart Data Models (FIWARE Data Models), provided that such corresponding models already existed [37].

One of the execution data models is Location, which is already presented in Section 8.2.1, and thus is not repeated in this section.

8.3.1 Resource

Resource is an abstract entity representing tangible objects that are present in a shop floor and that are of importance to manufacturing processes. Abstract means that no Resource instance exists directly. This entity is defined for readability purposes, and can be instantiated only by its concrete subtypes: Device, Material, Asset, and Person, as presented in Figure 18.

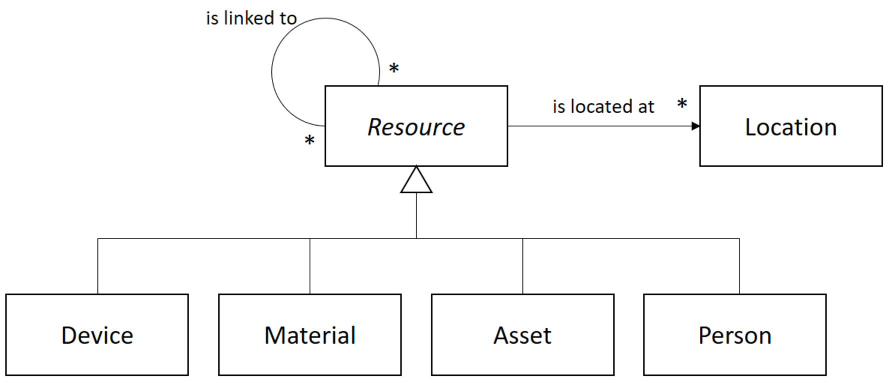

Figure 18 Resource data model

A resource references locations, at which it is currently located (present). For instance, an AGV can be at a specific production line and at a specific work cell (belonging to that production line) in the same moment.

A resource references other resources that it is physically linked to. For instance, an AGV with a robotic arm may be modelled as two resources but linked to each other. The information about the link may be used for scheduling purposes, e.g. when one of linked resources is busy, the other one cannot be used for different purpose at the same time.

A resource references Resource Definitions that specifies the type of the resource (see Figure 17).

Device is defined exactly the same as the Device entity of Smart Data Models as quoted below [29].

> *An apparatus (hardware + software + firmware) intended to accomplish a particular task (sensing the environment, actuating, etc.). A Device is a tangible object which contains*

some logic and is producer and/or consumer of data. A Device is always assumed to be capable of communicating electronically via a network.

Device entity has the same attributes as the origin FIWARE model. Device may be able to perform a step in manufacturing. For instance, robots, AGVs, or sensors are devices.

Material is a product (final or intermediate) or an ingredient of a manufacturing process. Its attributes represent its current state. The state may be physically observed, for instance location, but also process-specific, for instance a result of quality control. For instance, manufactured PCBs or capacitors required for assembling PCBs are materials.

Asset is a tangible item that is needed for a manufacturing process but is neither a material nor a device. It may be a tool or an element of a device. For instance, a gripper for a robot (i.e. for a device) or a hammer for a person are assets. The Asset model is not derived from the currently known data requirements but it is defined to complement the subtypes of Resource, so they cover together all kinds of resources experienced in the real world. Asset may be necessary in future scenarios.

Person is a human (human worker) that can perform a step in manufacturing.

8.3.2 Task

Task is a manufacturing operation that is to be executed. This data model is considered indivisible, although in the real world, it may be a complex multi-step operation. Task references Task Definition that specifies how to perform the work in detail (see Section 8.2.2).

Task may specify a set of involved resources: persons or devices required to work, assets to be used, materials to be used (as ingredients) or produced within the task.

Task may also specify locations, at which the operation should happen, together with the specific function of each location within the task. The functions of locations may be for instance “source location” or “target location” but their number and interpretation depends on a specific Task Definition.

Task’s relationships are presented in Figure 19. Locations’ functions are attributes of the Task-Location relationship.

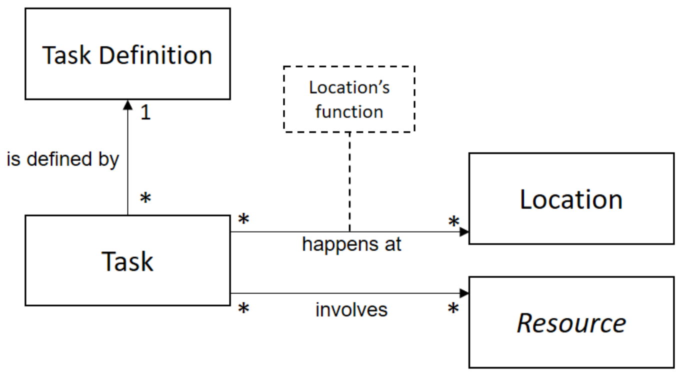

Figure 19 Task data model

Task's attributes are:

- specific work parameters (depending on Task Definition),
- the current status of the work (progress),
- output parameters (e.g. a binary result of a quality-check task).

The work parameters are constant for a specific task, and the other attributes change during execution.

For instance, a task may be the following: "Task for AGV #5 to move 5 pallets from the storage to production line #6; and for person X to move then the pallets onto the line". It is assumed that there is a Task Definition already defined that describes such a hybrid transportation work, but yet without parameter values such as how, how many, what, where from, where to. Then, this task could be represented as given in Figure 20.

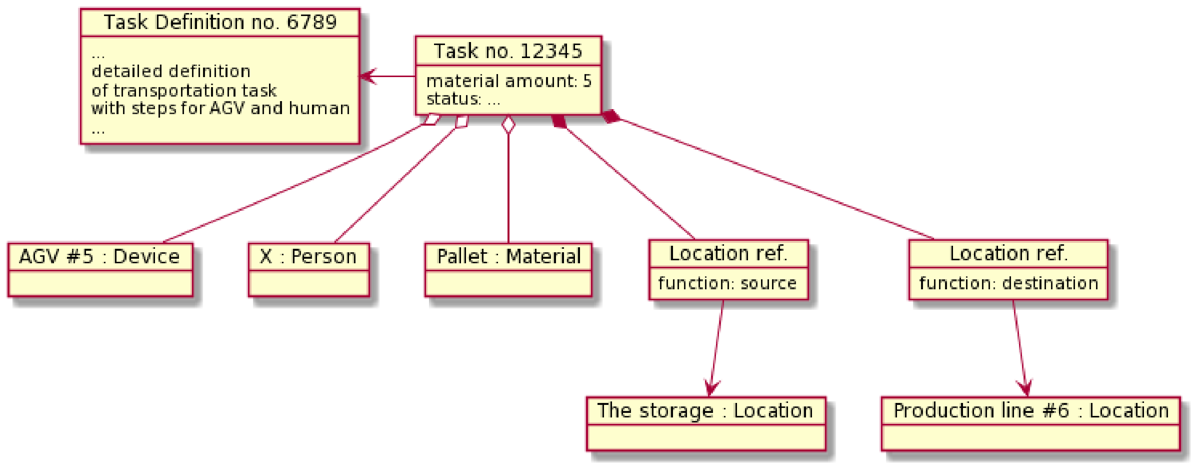

Figure 20 Example modeled task (UML object diagram)

8.3.3 Process and Step

Process models a set of manufacturing operations to be executed, i.e. Tasks. Process references Process Definition that specifies how to perform the work in detail. Process is not executed directly, but its constituent Tasks are executed by concrete agents.

Step defines how to perform the work in detail. Step may consists of other Steps (substeps).

These relationships are presented in Figure 21.

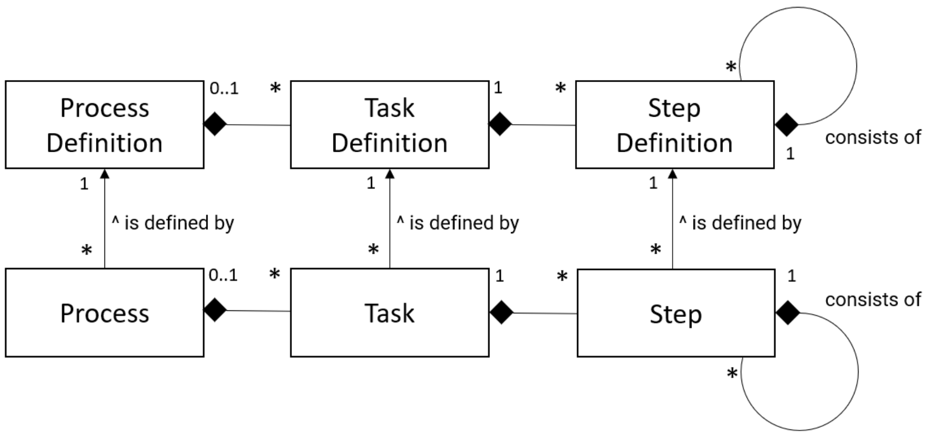

Figure 21 Process, Task and Step data model

Similarly as for Task, Process' attributes are:

- specific work parameters (depending on Process Definition),
- the current status of the work (progress),
- output parameters.

The work parameters are constant for a specific process, and the other attributes change during execution.

8.3.4 Alerts

Alert is defined exactly the same as FIWARE Alert data model (i.e. the existing FIWARE concept data model is adopted for SHOP4CF) as quoted below [29]. Examples not related to manufacturing are omitted in this quote.

> *This entity models an alert and could be used to send alerts related to (…) [specific categories of alerts]. The purpose of the model is to support the generation of notifications for a user or trigger other actions, based on such alerts.*
>
> *An alert is generated by a specific situation. The main features of an alert is that it is not predictable and it is not a recurrent data. That means that an alert could be an accident (…).*

Alert may be related to a number of resources or locations, as presented in Figure 22. This extends the origin entity from Smart Data Models.

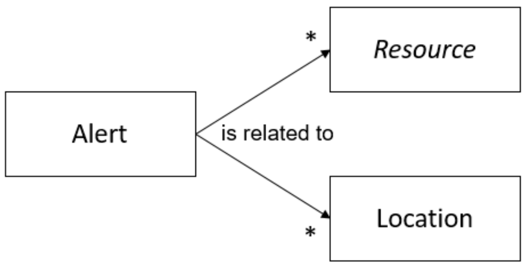

Figure 22 Alert data model

Alert entity has the same attributes as the origin FIWARE model. However, the allowed values for attributes “category” and “subcategory” are to be extended to model necessary types of alerts in SHOP4CF.

Example alerts are an alert generated by an observed safety breach at a production line, or a predicted maintenance request to prevent unexpected failures of a device.

8.4 Mapping to middleware architecture

The context information in FIWARE (Section 7.2) is based on the execution models as they represent the current state of manufacturing processes.

The concept of context information imposes additional requirements on how the lifecycle of data instances (i.e. concrete instances of entities) is organized (i.e. how long instances are kept in the current context). These lifecycle rules for the execution data models apply only when they are communicated via FIWARE, and they are defined below.

In general, an entity instance (an object) should be kept in the current context (i.e. in the Context Broker) as long as it stays relevant for manufacturing processes. Therefore, a concrete SHOP4CF system should be designed in such a way that not-anymore-relevant data instances are removed from the Context Broker. Note that removing data from the Context Broker does not mean it is also removed from the historical data store.

Resources (as defined in the previous section) are usually long living and define the state of the corresponding real-world objects at a given moment in time. There is always exactly one instance for each relevant real-world object. If the state of a real-world object changes (e.g. a change of location for a robot, a change of measured value by a sensor), software components do not send a new instance to the Context Broker, but only update relevant attributes of existing instance.

Processes and **Tasks** are usually short living (e.g. minutes, hours) and immutable, except for the status and output attributes. These entities should be removed from the current context when they become not relevant anymore. What relevancy means depends on a use case. Usually, a process or a task is no longer relevant when its status is marked as completed and this information is read by the component that created the task, so that component can safely delete the task instance from the Context Broker. Note that after deleting a task from the Context Broker, it remains in the historical context store (if in use) for statistics or analysis.

Alerts are usually short living and immutable. For this model, a few deletion policies can be adopted for different alert categories (subcategories):

- Alert’s creator deletes the alert instance when it observes that the alert is no more relevant.
- A component reacting to the alert deletes it once the situation is handled.

- On creation, the alert may be marked as transient, i.e. may contain an explicit expiration time.

The usual lifecycle rules are summarized in Table 2. A specific use case may require modifying some rules.

Table 2 Lifecycle rules for execution data models in FIWARE

Data model	Updates	Deletion
Resource	Every time the state of the corresponding real-world object changes	Rarely. Deleted by its creator when the resource is no more relevant (e.g. material leaving the factory).
Process, Task	Immutable, except for the status and output attributes	Deleted by its creator, usually when the task is completed.
Alert	Immutable	Either deleted by its creator, or deleted by a component that reacts, or marked as transient (auto-expiring)

8.5 Mapping to scenarios and middleware architecture

It was derived from the scenarios what specific information is to be exchanged between SHOP4CF components during actual manufacturing execution (the process is explain in Section 4.3).

Table 3 presents how the example specific information should be organized using the defined execution data models and the FIWARE middleware.

Table 3 Mapping of data models to specific information from scenarios

Specific information from scenarios	Data model	Details
Sensor values	Device	Updates to attribute “value”
AGV’s positions: x/y coordinate in cm	Device	Updates to attribute “location”
AGV’s/Robot’s status, for instance: “idle”, “busy”, “charging”, etc.	Device	Updates to attribute “deviceState”
Results of quality check of a product	Material (and/or) Task	Updates to the status attribute (and/or) Updates to the output attributes
Maintenance predictions: alert referring to a specific equipment	Alert	New instance created
Task orders to perform specific manufacturing operations, for instance: what kind of operations, how many parts to be loaded, etc.	Task	New instance created

Task's status updates (from actors)	Task	Update to the status attribute

8.6 Mapping to existing standards

For increased interoperability, the SHOP4CF data models do not only follow Smart Data Models (see Section 8.3) but also conform to the models of the ISA-95 standard (see Section 3.2).

Regarding Smart Data Models, it must be noted that no data models specifically dedicated for Smart Industry (Smart Manufacturing) existed in 2020, when this data architecture was initially designed. Thus, at that time, SHOP4CF adopted only the already-existing Device and Alert models from general-purposed Smart Data Models.

In 2022, three new data models were added to Smart Data Models, i.e. Manufacturing Machine Model, Manufacturing Model, and Manufacturing Machine Operation [38]. At that stage of the project, it was too late to include them in this data architecture. However, for interoperability between a SHOP4CF system and future systems adopting those new models, the relation among relevant data models has been analyzed.

The resulting mapping between the concept models of SHOP4CF and the aforementioned standards is summarized in Table 4.

Table 4 Mapping between SHOP4CF data models and existing standards

SHOP4CF data model	**ISA-95 data model**	**Smart Data Models**
Location	Hierarchy scope	-
Process Definition Task Definition Step Definition	Process Segment	-
Resource Definition	Equipment/Personnel/Material Specification	Manufacturing Machine Model
Device	Equipment	Device Manufacturing Machine
Material	Material Lot	-
Asset	Physical Asset	-
Person	Person	-
Process	Operations Definition	-
Task	Operations Segment	Manufacturing Machine Operation
Alert	Work Alert	Alert

9 Medium-level logical platform architecture

This medium-level design aims at presenting how software components connect to platform components, i.e. how particular SHOP4CF components connect to middleware components, 3rd-party information systems, and IoT devices. This further elaborates the high-level platform architectures presented in Section 7.

To this end, SHOP4CF components are grouped into five **interoperability classes** that have different characteristics of such connections. A single component can have a few such logical connections, thus it can be assigned to more than one class.

Elaborating the top-level platform architecture (Figure 11) and the high-level middleware architecture (Figure 12), by decomposing SHOP4CF components into the interoperability classes, leads to Figure 23. The interoperability classes are the five modules within SHOP4CF components.

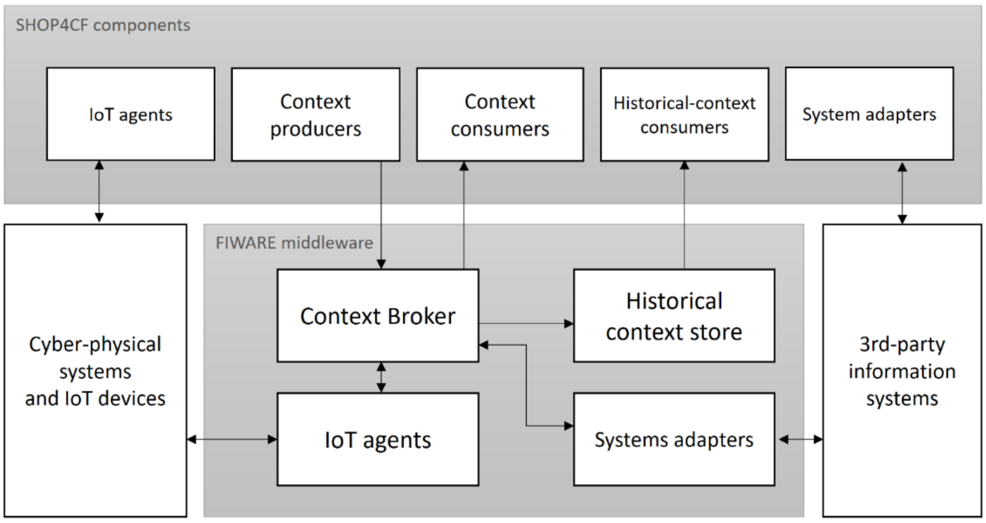

Figure 23 Interoperability classes of SHOP4CF components

The set of interoperability classes can be extended in the future revisions if new use cases are identified. Such case may be for instance producers or consumers of stream data (see also Section 7.3).

The rest of this section is organized as follows. Firstly, the mapping of concrete SHOP4CF components to the classes is discussed. Secondly, the classes are further characterized. Finally, the example medium-level architecture of a widely interoperable component is presented.

9.1 Mapping to software architecture

This framework architecture does not specify a fixed mapping of components to the interoperability classes as this may vary depending on future use cases. However, the mapping based on the already-defined pilot scenarios is presented in Table 5. It is based on the analysis of the data requirements, as discussed in Section 4.3.

This mapping should not be considered as an initial one, based on the first pilot scenarios.

Table 5 Mapping of SHOP4CF components to interoperability classes

Interoperability class	**High-level subsystems**	**SHOP4CF components**
Both context producer and consumer	Execute G./L. Analyze G./L.	ADIN, DTS, HA-MRN, FLINT, MPMS, VR-RM-MT, VQC
Context producer	Execute G./L. Analyze G./L.	PMADAI, Wi-Pos
Context consumer	Execute G./L. Analyze G./L.	AR_Man_Edit, AR_Teleassist, AR-CVI
Historical-context consumer	Design G./L. Analyze G./L.	IL-DT, PMADAI
System adapter	all	DCF, M2O2P, FLINT, MPMS
IoT agent	Design Local Execute Local	DTS, FBAS-ML, HA-MRN, M2O2P, FLINT, ROS-Mon, Wi-Pos

9.2 Characteristics of interoperability classes

Most of FIWARE components, including the Context Broker, communicate via FIWARE NGSI API [39]. Depending on the interoperability classes, the communication between software and platform components is organized differently, as presented below.

Context producer is a component that sends updates of the context information to the Context Broker, as presented in Figure 24.

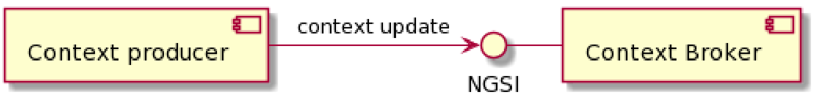

Figure 24 Platform architecture for context producers

Context consumer can work in two modes, as follows.

1. Context consumer in subscription mode first subscribes to certain context changes, and then Context Broker initiates the communication when relevant updates occur, as presented in Figure 25. It is usually the preferred mode for consuming context.

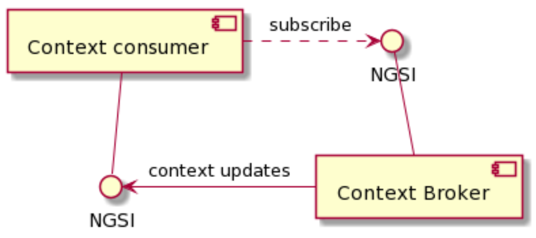

Figure 25 Platform architecture for context consumers in subscription mode

2. Context consumer in query mode is a component that queries Context Broker for a specific context information, as presented in Figure 26. It is the simpler but rarer case as it requires active polling from the component. It makes sense when the component needs to request the context information rarely, only in certain situations.

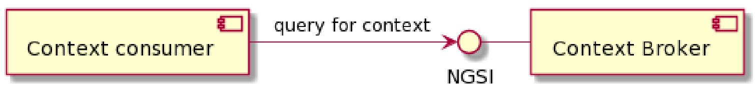

Figure 26 Platform architecture for context consumers in query mode

Historical-context consumer is a component that queries the historical context store, as presented in Figure 27. Several implementations and interfaces of this store exist [33].

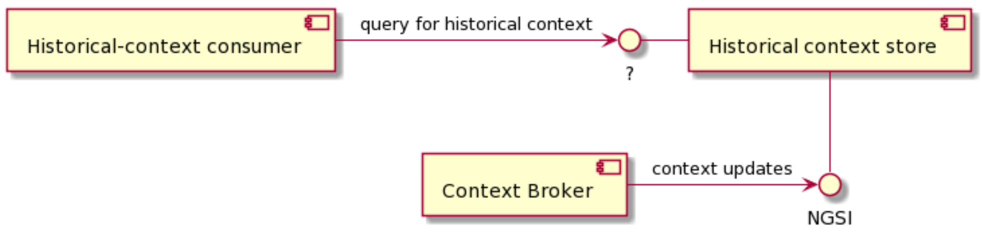

Figure 27 Platform architecture for historical-context consumers

System adapter and **IoT agent** are components that communicate with 3rd-party information systems (such as MES) or IoT devices (cyber-physical systems), respectively, via specific interfaces that such platforms provide. Such communication may be initiated from both ends, depending on concrete platforms and use cases.

9.3 Extensive example of interoperability

From the perspective of interoperability, the Manufacturing Process Management System (MPMS) is the most complex case among SHOP4CF components. It addresses most of the pilot scenarios and belongs to most of the interoperability classes: context producer, context consumer, and system adapter.

In this section, its logical software architecture is first elaborated. Then, its interfaces to other platform components are specified. This section focuses on the Design Global and Execute Global subsystems of the high-level logical software architecture of Figure 7, as these are the two subsystems that MPMS covers in all known use cases.

MPMS provides end-to-end (i.e. from order reception until product delivery) manufacturing process management and orchestration of activities by:

- modeling processes and agents,
- executing in automated way the processes by assigning activities to agents,
- providing process monitoring for a complete status overview of the manufacturing processes.

Regarding the design side, MPMS is decomposed in a number of sub-components, as shown in Figure 28. The "Process/Agent/Shop Floor Data" DB is part of the specG DB of Figure 8. The "Task/Step/Cell Data" is part of the SpecL DB of Figure 8. "Product Defin." DB is also added as this provides useful information in the process flow modelling. As it is part of external systems (e.g. an ERP), it is shown in blue.

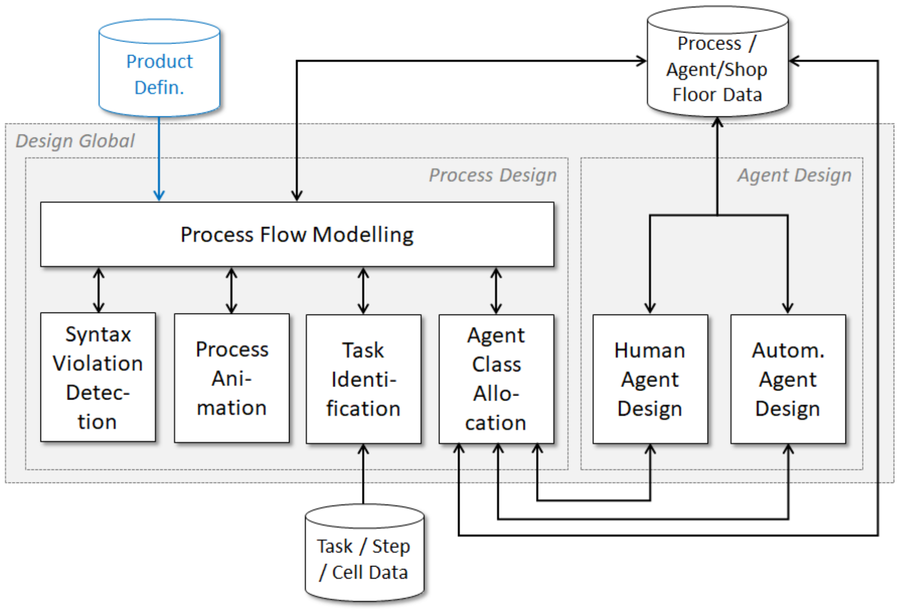

Figure 28 Logical architecture of MPMS in Design Global subsystem

Regarding the execution side, MPMS mainly consists of a Process Engine that enacts the process models and assigns tasks to agents. A Production Execution Monitoring visualizes the production status. All sub-components are shown in Figure 29.

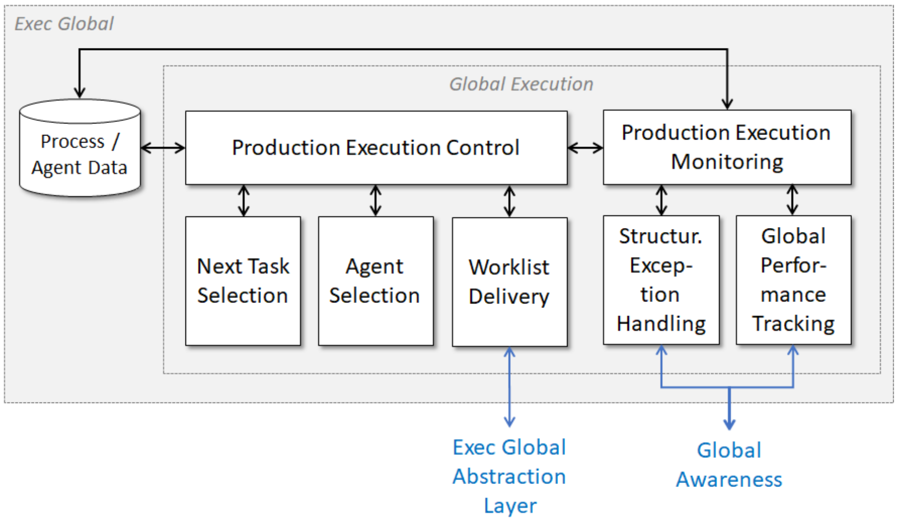
Exec Global
Global Execution
Process / Agent Data
Production Execution Control
Production Execution Monitoring
Next Task Selection
Agent Selection
Worklist Delivery
Structur. Excep-tion Handling
Global Perfor-mance Tracking
Exec Global Abstraction Layer
Global Awareness

Figure 29 Logical architecture of MPMS in Execute Global subsystem

Regarding the interfaces from/to other SHOP4CF components, for sake of brevity in this medium level, the focus is only on the main sub-components for each phase, i.e. the Process Flow Modelling (Modeler) and the Process Execution Control (Process Engine). The platform architecture with the relevant interfaces is shown in Figure 30.

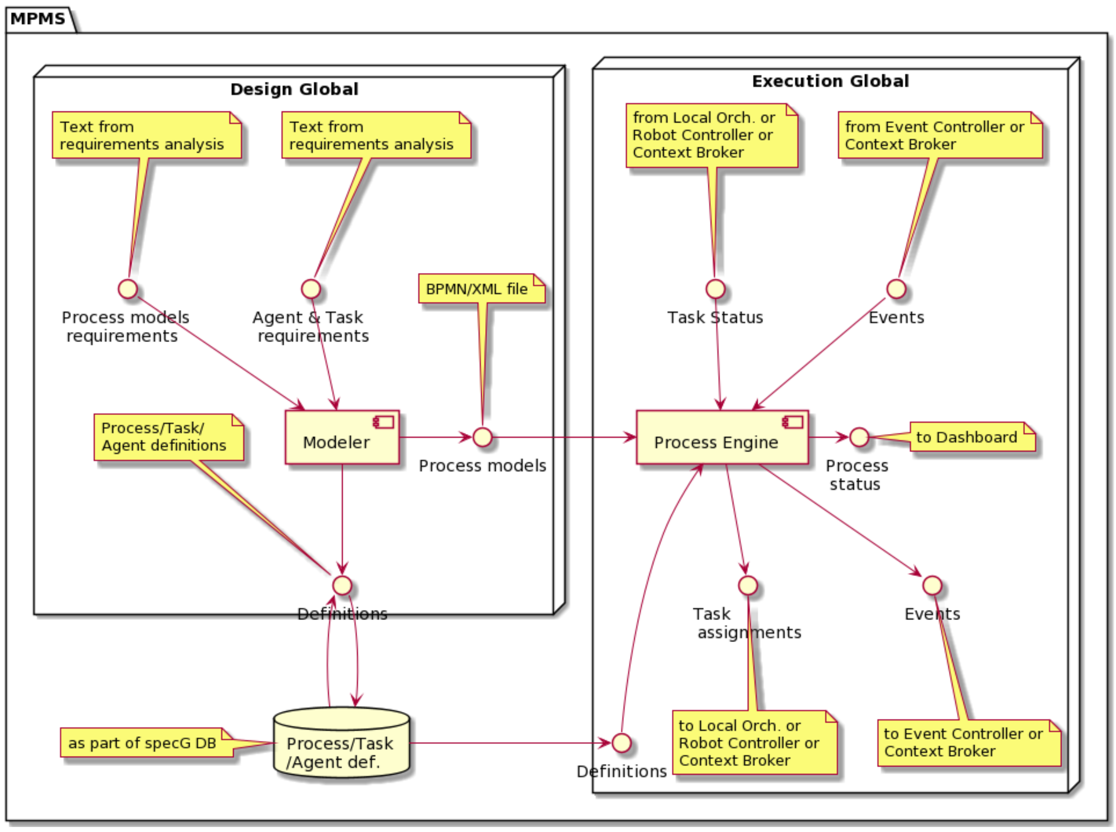
MPMS
Design Global
Text from requirements analysis
Text from requirements analysis
Process models requirements
Agent & Task requirements
BPMN/XML file
Process/Task/ Agent definitions
Modeler
Process models
Definitions
as part of specG DB
Process/Task /Agent def.
Execution Global
from Local Orch. or Robot Controller or Context Broker
from Event Controller or Context Broker
Task Status
Events
Process Engine
to Dashboard
Process status
Task assignments
Events
Definitions
to Local Orch. or Robot Controller or Context Broker
to Event Controller or Context Broker

Figure 30 Platform architecture of MPMS

10 Interoperability of the architecture

10.1 Relation to FIWARE Smart Industry

The FIWARE Smart Industry (FIWARE-SI) architecture is introduced in Section 3.2.3. This section presents how the SHOP4CF architecture implements FIWARE-SI, and specifically how the latter is extended by SHOP4CF.

From the perspective of SHOP4CF, FIWARE-SI addresses two UT5 aspects: platform and software. The context of each of the two aspects designed in SHOP4CF is presented separately in the following subsections.

10.1.1 Platform aspect

FIWARE Smart Industry (FIWARE-SI) corresponds to the SHOP4CF platform architectures presented in Section 7. FIWARE-SI is based on the technology-oriented diagram presented earlier in Figure 4, while the SHOP4CF platform architectures are the K4+1 logical view (i.e. functionality-oriented, not technology-oriented).

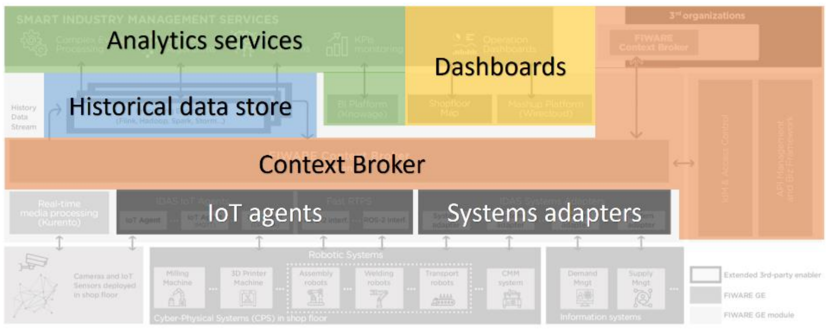

Figure 31 Transformation of FIWARE Smart Industry architecture to the logical view

Thus, to be able to present the mapping between SHOP4CF and FIWARE-SI, the latter is first transformed to its logical view, as presented in Figure 31. The transformation is done by aggregating the technology-oriented components into logical components: Context Broker, historical data store, IoT agents, systems adapters, analytics services, and dashboards. Real-time media processing is omitted, as (currently) there is no such use case in SHOP4CF.

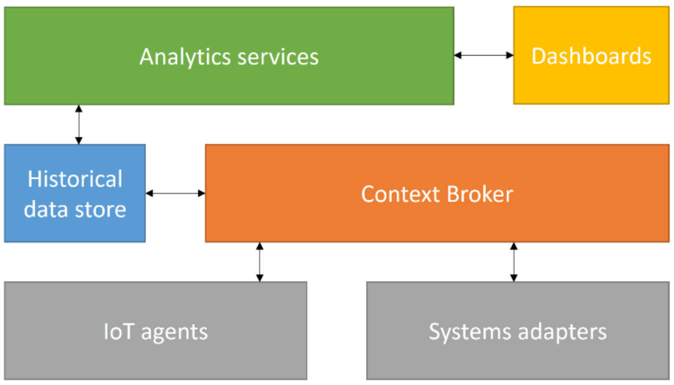

Figure 32 Logical view of FIWARE Smart Industry

The resulting diagram of the (relevant) logical components of FIWARE-SI is presented in Figure 32. The arrangement and colors of the components corresponds to Figure 31.

Having that, the mapping between the logical view of FIWARE-SI and the SHOP4CF high-level middleware (platform) architecture (Figure 12) can be designed, and this is provided in Figure 33. The mapping depicts how the SHOP4CF architecture implements the FIWARE-SI architecture. Note however that only a subset of SHOP4CF components cover Analytics services.

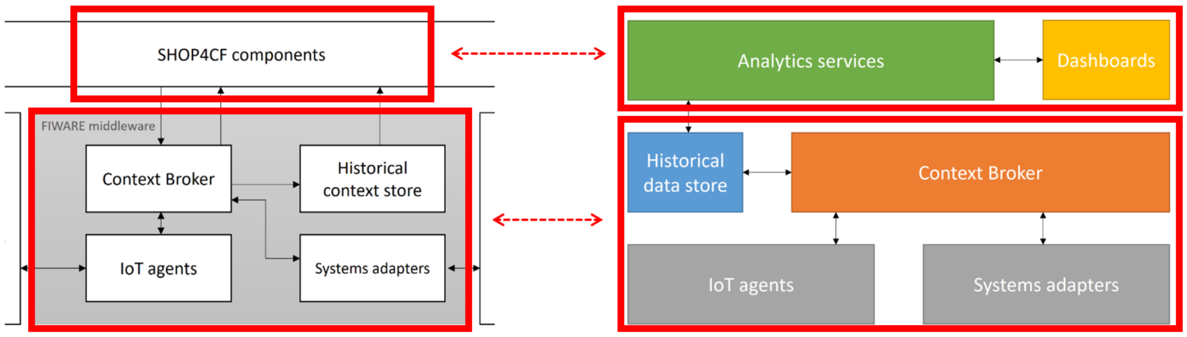

Figure 33 Mapping between the logical middleware architecture and FIWARE Smart Industry

In addition, differences regarding the arrows exist between the two diagrams. In FIWARE-SI, Analytics services connect only to Historical context store, and Dashboards connect only to Analytics services. In SHOP4CF, both Analytics services and Dashboards can directly connect also to Context Broker to access the (current) context information. This is considered an extension of FIWARE Smart Industry.

10.1.2 Software aspect

The FIWARE-SI diagram (Figure 4) contains also components that, from the perspective of SHOP4CF, provide high-level functionality and belong to the UT5 software aspect. Thus, it can be presented how the six subsystems of the high-level software architecture (Figure 7) can be positioned in the FIWARE-SI design.

This mapping is presented in Figure 34 and explained below.

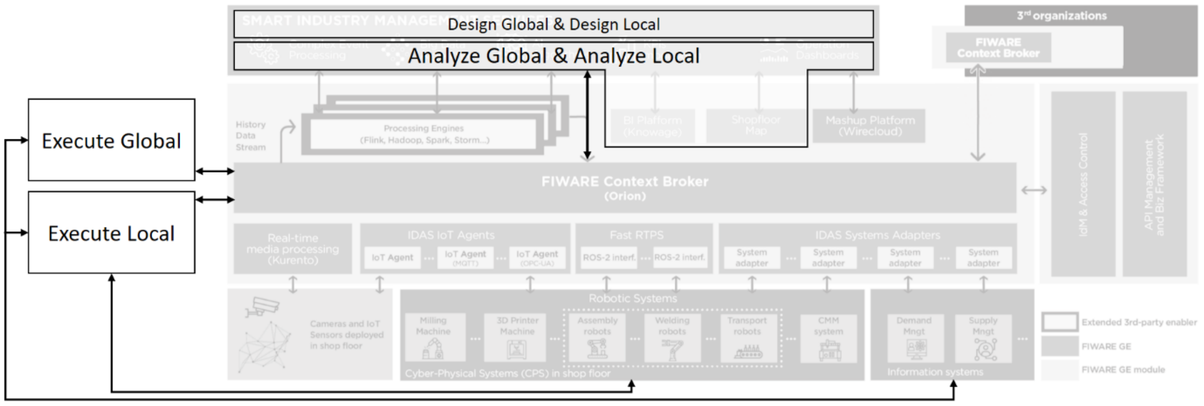

Figure 34 Mapping between the logical software architecture and FIWARE Smart Industry

Firstly, the most intuitive part of this mapping is subsystems **Analyze Global & Local** mapping to the analytics components of FIWARE-SI (such as Complex Event Processing, Big Data Algorithms, AI Algorithms, KPIs monitoring; see Figure 4). However, components of the Analyze phase do not need to rely only on data from the Processing Engines (in SHOP4CF referred to as the historical data store), as the original arrows depict, but they can also directly access the (current) context information in the Context Broker (see also Figure 13 for how the Log databases can be implemented). Thus, the FIWARE-SI architecture is extended on this matter by adding the connection from Analyze to the Context Broker (i.e. the vertical black arrow at the top).

Secondly, subsystems **Design Global & Local** can also be seen (at least partially) in the same place of FIWARE-SI as based on the historical data, the analytics components may provide useful insights for (re-)design of manufacturing processes. This interpretation corresponds to the two “offline redesign information” arrows in Figure 8.

Finally, FIWARE-SI does not directly address functionalities that SHOP4CF models in the execution phase (see Section 6.1). Therefore, subsystems **Execute Global & Local** extend FIWARE-SI. Moreover, as some SHOP4CF components can directly connect to robotic systems and 3rd-party information systems (see Figure 23), such connections are also depicted (the arrows at the bottom).

10.2 Interoperability with the Robot Operating System (ROS)

To highlight the adaptability with robotic agents and applications, the SHOP4CF architecture considers integration with the Robot Operating System in two different versions, i.e. ROS and ROS2. The ROS and ROS2 libraries available in the market include robot control, sensors, and different state-of-the-art applications, such as safety applications for human-robot collaboration or shared robot control. As a result, it becomes necessary to deploy a communication bridge between FIWARE and ROS/ROS2, which guarantees that all the robot components can access and share data available in the Context Broker.

Existing ROS-FIWARE bridges are available for FIWARE NGSI v2 [40] [41]. However, the non-existent communication node for FIWARE NGSI-LD created the necessity for a new ROS-

FIWARE package, and a system connector for ROS/ROS2 node is under development and testing. It is expected to be released as open source in the near future. The node will communicate with the Orion-LD Context Broker, publishing robot data and subscribing to context information provided by the other SHOP4CF components (e.g., trajectory points, achieved tasks, sensor information, among others). It can be integrated to either a ROS or ROS2 workspace, and it provides different configuration files for the Context Broker configuration, the expected data model, and the ROS topics.

The architecture regarding this system adapter to integrate robotic agents with the SHOP4CF architecture can be seen in Figure 35. The System adapter is a ROS node that receives a configuration file representing the data models and then publishes/listens to topics accordingly. For now, the system adapter considers data models in the Task and Device, but it could be extended for other entities through that configuration file.

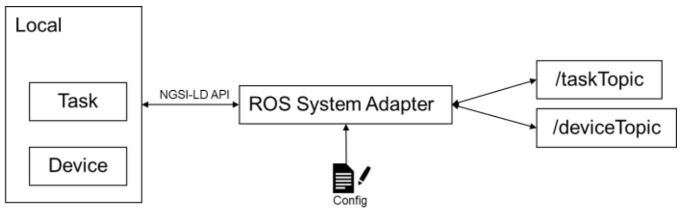

Figure 35 Architecture of the system adapter for ROS/ROS2

10.3 Interoperability using International Data Spaces

As outlined in Section 3.2.4, in IDS, different actors (data providers and consumers) can become part of the Data Space by implementing an IDS connector. The IDS connectors can be enhanced with data apps from the IDS App Store. These data apps can implement interfaces to third party systems, such as ERP systems or FIWARE ecosystems.

Figure 36 shows the concept of the FIWARE TRUE Connector [42], which enables trusted data exchange between FIWARE instances by making use of the IDS ecosystem. In addition, data exchange between FIWARE systems and non-FIWARE compliant data providers becomes possible with this architecture. The NGSI-LD data app enables communication with the FIWARE context brokers.

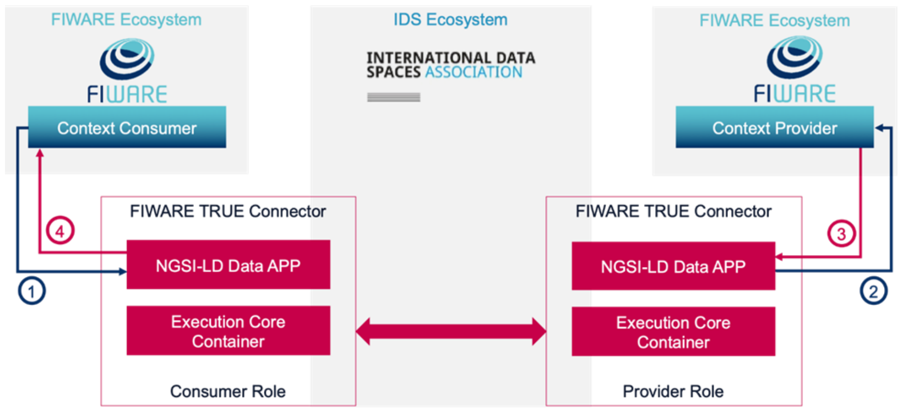

Figure 36 FIWARE TRUE Connector (source: fiware-true-connector.readthedocs.io)

In SHOP4CF, the Context Broker of choice is Orion-LD [35], which implements the NGSI-LD information model and API [36].

Data exchange using FIWARE TRUE Connector has been tested during the period of the project but the implementation did not result to be simple and straightforward. Moreover, adaptation to different implementation requirements could not be easily provided. Therefore, an analysis of the current SHOP4CF pilots' interoperability and final deployment was performed, as also reported in the Connected Factories D1.4 [43]. During this second analysis it was discovered that the almost all the pilots were deploying the components in local instances not connected to the internet. Therefore, for the integration of an IDS compliant system this had to be considered.

In IDS compliant data spaces, the preferred connector is the open source backed Eclipse Dataspace Connector (EDC) [44]. Such connector is supported by the Eclipse foundation, and it implements the main component of a data space, the control plane. Such connector, thus, ensures that data can be discovered, connected, negotiated, enforced and audited. In other words, the connector is a "gatekeeper" to safe data spaces as required often by companies. Therefore, SHOP4CF decided to use the same well-known connector. Thus, ensuring high interoperability also with other data spaces initiatives (e.g. Catena-X, Gaia-X). The connector works with the same principles of IDS, for sake of clarity this is shown also in Figure 37. Therefore, to guarantee data transfer two connectors should communicate in a data space. On one side, the provider of data, also known as source. On the other side, the receiver of data, also known as sink. Therefore, after having established the connection the two connectors enter in a contract management phase where the legal entities behind the connector negotiate the terms of the data sharing while the data is described just with meta data. Afterwards, having the two entities agree on the terms the connector executes the real data transfer. The benefits of using this connector are multifold. However, the most important for SHOP4CF is the possibility to share the data using either a local deployment or relying on trusted cloud services (e.g. AWS. Azure).

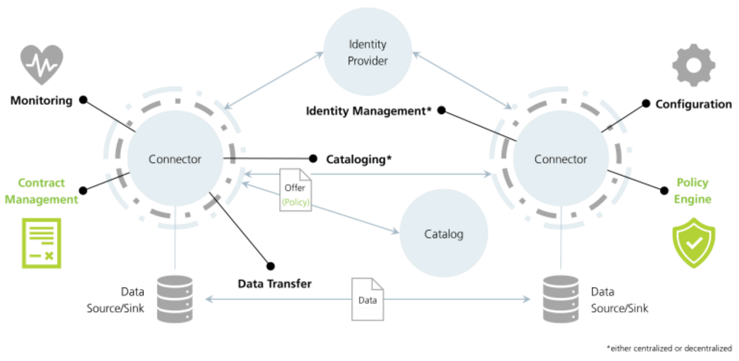

Figure 37 Exemplary data transfer among using EDC (source: eclipse-dataspaceconnector.github.io)

Considering the requirements of local deployment and the capabilities of the EDC different concepts were drafted. Those are shown in Figure 38. The main idea behind these concepts is to enable the sharing of data living in the Orion-LD Context Broker.

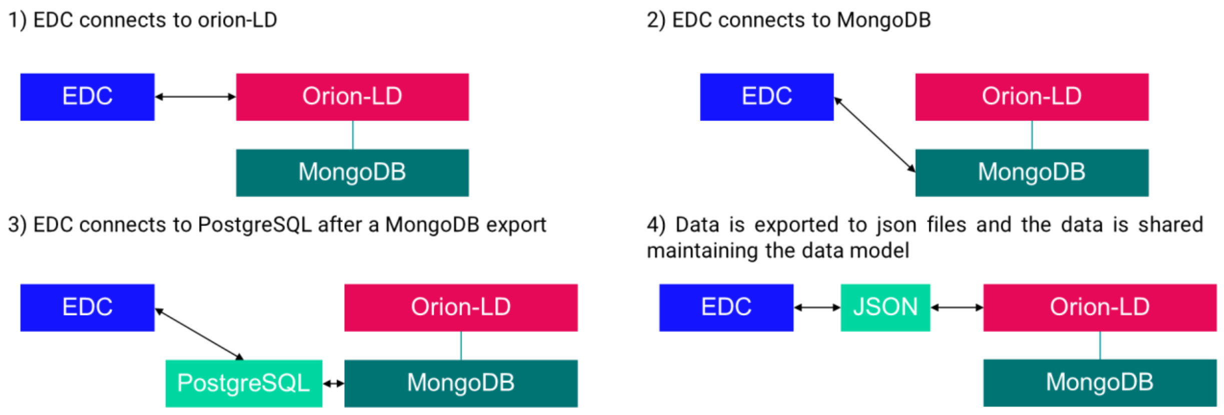
1) EDC connects to orion-LD
EDC
Orion-LD
MongoDB
2) EDC connects to MongoDB
EDC
Orion-LD
MongoDB
3) EDC connects to PostgreSQL after a MongoDB export
EDC
Orion-LD
PostgreSQL
MongoDB
4) Data is exported to json files and the data is shared maintaining the data model
EDC
JSON
Orion-LD
MongoDB

Figure 38 Draft concepts for the integration of EDC with SHOP4CF (FIWARE) middleware

By enabling this, all the data shared between the components can be made available to external parties, thus allow the integration of use cases related to traceability or other. The main difference between the concepts is the technical implementation and they can be described as follows:

- Concept 1. In this concept the EDC connects directly to the Orion-LD context broker and through an interpreter it can export all the data and make it available to the data space. Due to the direct connection to the broker, the standardization with the SHOP4CF data models is ensured.
- Concept 2. In this concept EDC connects to the underlying database which Orion-LD uses. As long MongoDB is a well-known database, this idea leverages the already-built standard interfaces for the database for obtaining the data, thus simplifying implementation. However, ensuring consistency with the SHOP4CF data model cannot be guaranteed as long the data stored in the MongoDB might not respect the standard. Additionally, EDC might access the database at the same time as Orion-LD, therefore, data integrity might be jeopardized.
- Concept 3. This is an extension of concept 2. The underlying idea is the same. However, through the adoption of an additional database based on PostgreSQL to

reduce the possibilities to jeopardize data integrity. Additionally, having this database in the middle can improve data security as long steps must be taken to select which data should be transferred from MongoDB to the PostgreSQL. In this concept two wrappers should be added to ensure that data is correctly transferred between MongoDB and PostgreSQL.

- Concept 4. This is an additional extension of concept 3. Therefore, it guarantees the data integrity by having the data transferred on a temporary storage. However, differently from the previous concept, instead of having two databases communicating together the data is exported directly from the Orion-LD in its JSON-LD format. This approach guarantees that on one side the data exposed to EDC is just the one that the source wants to share. On the other side, it guarantees that the data follows the SHOP4CF data model, which was not in concept 3. Finally, as the data is in JSON-LD, a well-known practical standardized data format [45], it remains interoperable with many use cases.

With this list it was decided for concept 4 as long it allows the maximum security and flexibility in the implementation. Therefore, with this concept the technical concept was developed further and, considering the limits that pilots might face, the selected implementation of concept 4 is as shown in Figure 39.

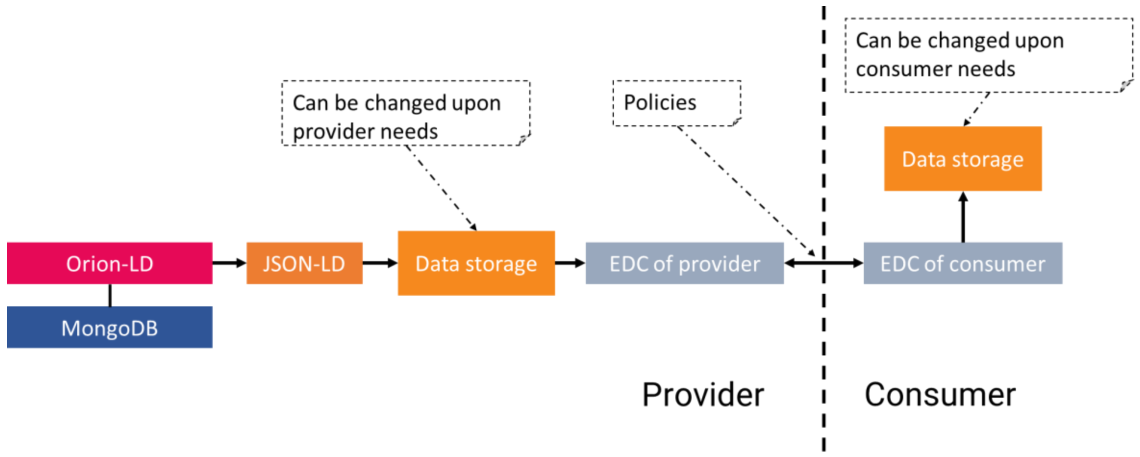

Figure 39 Technical implementation of data sharing using the EDC connector

Through this implementation, the manufacturing company can deploy all the components on a non-internet facing network. Thus, satisfying the needs of the factory floor. Then, if the company want to engage in data sharing it exports the desired data into JSON-LD format and uploads them in a safe cloud storage compatible with the company policies. Afterwards the EDC connector can be simply configured to use that storage as source and then policies can be simply enforced by the data provider. Therefore, the manufacturing company has full control of the transmission chain of its data and there is a high level of customization of the target storages as long EDC supports several cloud systems.

However, it is important to underline that due to this infrastructure, the data that will be exchanged it will be non-real time. Therefore, this concept addresses use cases where data should be exchanged without hard time constraints. For hard time constraints other methodologies should be investigated.

For the sake of clarity this infrastructure was tested with a simple use case concerning the data sharing of images collected at a robotic workcell to test if the data sharing could be applied using as backend the Amazon Web Services (AWS). The implemented architecture is shown in Figure 40.

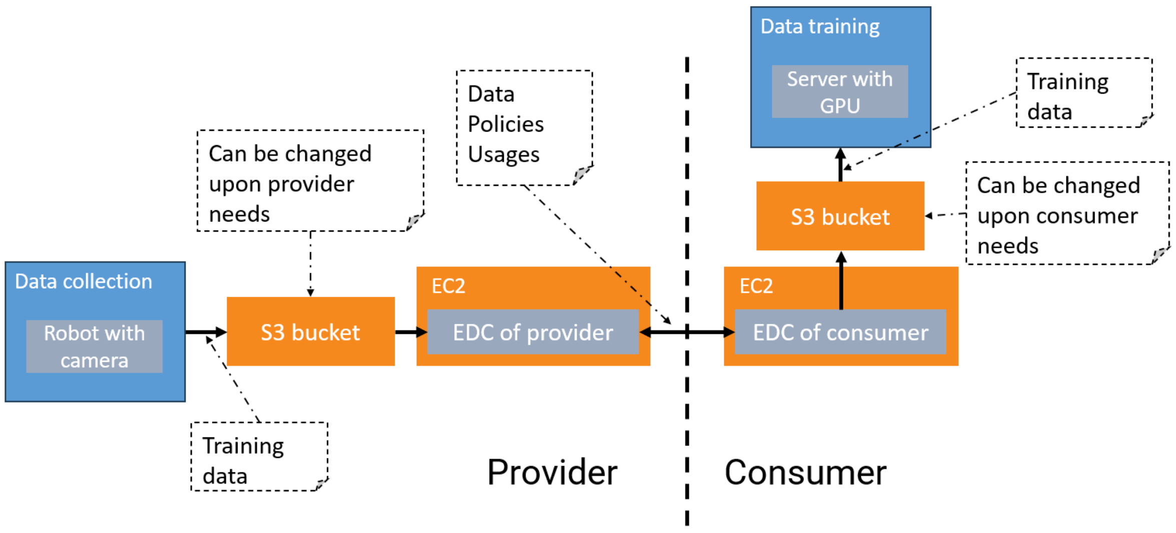

Figure 40 Example implementation of EDC connector in Amazon Web Services (AWS) public cloud

In this example the AWS has been used as follows. First of all, the S3 buckets act as storage solution and data can be uploaded/downloaded by the respective provider and consumer considering the company policies. Afterwards, to make the data available to the data space an EC2 instance has been integrated with the EDC connector. The same was done for the consumer. Next, having the two EDC, which require a public IP address, the data sharing can be triggered.

10.4 Interoperability to other system adapters

As shown before, an adapter for robotic agents is being developed. However, due to many different systems deployed at different sites, as shown from the pilot questionnaires (see Appendix C), a need for a more general system adapter arose. More precisely, different custom protocols were integrated (e.g. OPC UA, XML) and doubts on how to correctly interface with the SHOP4CF architecture were identified.

After this preliminary analysis, in SHOP4CF, the Web of Things (WoT) was identified as suitable for the task. The Web of Things (WoT) is a web application-layer for IoT. The main idea of WoT is take advantage of the potential of IoT, making it easier to create applications without the need to master the disparate variety of IoT technologies and standards.

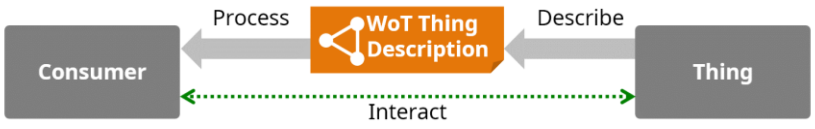

Figure 41 FIWARE Consumer-Thing interaction (source: www.w3.org)

The WoT-IL is a tool that will port any REST interface based on OpenAPI on the W3C Web of Things (WoT) standard in order to extend the interoperability through the standard approach. Basically, the component makes WoT compatible with OpenAPI. In this way, it is possible to take advantage of the greater power of WoT by allowing describing semantic contexts.

This tool was originally conceived as a vertical component that can be used globally and locally with the restriction that the functionality you want to be mapped had to be provided through a web-based API and documented with OpenAPI. Now, the component has started to adapt to other project needs and the ability to serve as a communication bridge between FIWARE and standards such as OPC UA or OpenAPI has been added. Therefore, WoT should be used in case translations between data models are necessary. However, in order to translate into different vocabularies, either adaptive algorithms should be developed to allow translating one data model into another, or translation fields should be manually defined for the mapping.

10.5 Interoperability to existing industrial architectures

The last investigated topic is the possibility to use the SHOP4CF architecture with other existing industrial architectures, in this case the Siemens Industrial Edge [46], as long it is a known industrial architecture, which can enable industrialization of some of the SHOP4CF concepts. More specifically, Siemens Industrial Edge is a platform that enables to share apps across different Industrial Edge devices. However, to enable the communication between different apps, the MQTT broker has been selected as standard databus for the platform.

Therefore, to enable the use of SHOP4CF components in the Siemens Industrial Edge, two possible integrations have been explored. The first compromises the addition of an FIWARE Context Broker that host the FIWARE communication. The second is the conversion and the usage of the MQTT broker for the communication among software components.

For the first option, fortunately, the Siemens Industrial Edge easily allows such integration by introducing the FIWARE Context Broker as a custom Docker container [47]. Thanks to that, the integration does not require any additional effort apart from the standard network and access configuration.

Regarding the second option, unfortunately no direct solution is possible. However, it was discovered that MQTT can accept any kind of JSON formatted data, such as the technical representation of the SHOP4CF data models. Adapters such as the WoT-IL or the eProsima Integration Service could be used for this purpose [48].

11 Extending the framework

The SHOP4CF architecture defines an open framework (a template) that is intended to be easily extended in the future. Extensions could be designed by the SHOP4CF project but also by third parties, even so that the project consortium was not involved.

Considered extensions are mainly about functionality of the SHOP4CF framework, but also about middleware components supporting high-level functionality. Extending functionality may happen by either adding new software components or extending (upgrading) existing ones. Mapping of such extension approaches to the architecture design is presented below.

Adding a new software component to SHOP4CF imposes the following design decisions:

1. Define a scenario involving the new functionality.
2. Position the component in the high-level logical software architecture (Figure 7) by assigning it to the phases and levels.
3. Identify necessary integrations with other components (see the example in Figure 10).
4. Identify middleware components that can support these integrations (see Figure 13).
5. Identify data models that can represent the information to be communicated (see Figure 14).
6. Identify necessary connections to other systems (see Figure 23).
7. Switch from the above logical views to the development view (see Figure 2) and implement gaps such as necessary interfaces, for instance.

Extending an existing software component requires at least a subset of above design decisions. It may require adapting previously taken decisions for this component in each of the aforementioned steps.

Adding middleware components is aimed at extending functionality of the internal services that middleware provides to functional software components. Such functionality may be for instance supporting exchange of streaming data. It may be an intermediate step towards functional extensions discussed above.

Adding a middleware component requires positioning of such a component in the middleware architecture (Figure 12) and defining how software components can interact with this component (see Figure 23 and Section 9.2). It might also require defining some data constraints (as for instance in Section 8.4).

12 Conclusions

This document defined the SHOP4CF framework architecture that ensures coherence and interoperability of the SHOP4CF components. Coherence of the components is addressed mainly by positioning of components in the logical software architecture and arranging the relationships between them (Section 6). Interoperability is addressed mainly with the platform architectures (Section 7 and 9) and the data architecture (Section 8) that both facilitate and standardize communication among the components.

Cross-dependencies between the various views where emphasized, namely how the platform components support the functional software components but also how the data architecture maps to relevant parts of the software and platform architectures.

Further design decisions may be taken and further architecture details may be specified as the project advances. Concrete areas for such decisions were highlighted, and these are for instance SHOP4CF components with their specific functionalities, more specific data models, new middleware components, new types of software-middleware connections. The next revision of this document will report changes introduced in the meantime.

The numerous logical architecture views from this document are to be taken forward by software developers and system integrators, so they can make design decisions based on their specific perspectives (i.e. corresponding to the K4+1 development and process views).

Bibliography

[1] P. Bouklis and A. Garbi, "Deliverable D5.1: Definition of the deployment scenarios," SHOP4CF, 2020.

[2] P. Bouklis, “Deliverable D5.3: Final pilot demonstrations,” SHOP4CF, 2022.

[3] P. Grefen, "Results Architecture Workshop (internal report)," SHOP4CF, 2020.

[4] P. Grefen, "Business Information System Architecture (Version Spring 2015)," Eindhoven University of Technology, 2015.

[5] P. Kruchten, "Architectural Blueprints – The 4+1 View Model of Software Architecture," *IEEE Software,* vol. 12(6), pp. 42-50, 1995.

[6] M. West, Developing High Quality Data Models, 2010.

[7] T. Kuula, S. Aromaa and P. Heikkilä, "Deliverable D2.1: Industrial requirements report," SHOP4CF, 2020.

[8] Z. Domagala, "Deliverable D3.1: Functional requirements specification," SHOP4CF, 2020.

[9] K. Traganos, I. Vanderfeesten, Z. Domagala, G. L. Mallman and P. Grefen, "Component Analysis Report, Task 3.2 (internal report)," SHOP4CF, 2020.

[10] "Communication in SHOP4CF," 2020. [Online]. Available: https://shop4cf.github.io/communication-docs/. [Accessed 1 December 2020].

[11] J. Zhou, Y. Zhou, B. Wang and J. Zang, "Human–cyber–physical systems (HCPSs) in the context of new-Generation intelligent manufacturing," *Engineering,* vol. 5, no. 4, pp. 624-636, 2019.

[12] S. Angelov, P. Grefen and D. Greefhorst, "A framework for analysis and design of software reference architectures," *Information and Software Technology,* vol. 54, no. 4, pp. 417-431, 2012.

[13] M. Moghaddam, M. N. Cadavid, C. R. Kenley and A. V. Deshmukh, "Reference architectures for smart manufacturing: A critical review.," *Journal of Manufacturing Systems,* no. 49, p. 215–225, 2018.

[14] J. Lee, B. Bagheri and H. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters,* vol. 3, pp. 18-23, 2015.

[15] ISA, "ANSI/ISA 95 - Enterprise-Control System Integration. Multi-part Standard," International Society of Automation, North Carolina, USA, 2000.

[16] ISA, "ISA95, Enterprise-Control System Integration," [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95. [Accessed 14 December 2020].

[17] ISO, "IEC 62264:2013 Enterprise-control system integration, multi-part standard," International Organization for Standardization, Geneva, Switzerland, 2013.

[18] T. Bauernhansl, "Die Vierte Industrielle Revolution - Der Weg in ein wertschaffendes Produktionsparadigma," in *Handbuch Industrie 4.0 Bd.4: Allgemeine Grundlagen*, Berlin, Heidelberg, Springer Berlin Heidelberg, 2017, pp. 1-31.

[19] Deutsches Institut für Normung, "DIN SPEC 91345:2016 - Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)," Berlin, 2016.

[20] "Smart Industry - FIWARE Open Source Platform for Smart Industry," [Online]. Available: https://www.fiware.org/community/smart-industry/. [Accessed 1 December 2020].

[21] B. Otto, "Reference Architecture Model for the Industrial Data Space," 2017.

[22] “IDS Reference Architecture Model,” [Online]. Available: https://internationaldataspaces.org/use/reference-architecture/. [Accessed 1 December 2021].

[23] “GAIA-X,” [Online]. Available: https://www.data-infrastructure.eu/GAIAX/Navigation/EN/Home/home.html. [Accessed 1 December 2021].

[24] "Catena-X," [Online]. Available: https://catena-x.net/en/. [Accessed 26 October 2022].

[25] "Traceability as the backbone of Catena-X," [Online]. Available: https://catena-x.net/en/mehrwerte/traceability. [Accessed 26 October 2022].

[26] P. Grefen and G. Boultadakis, Designing an Integrated System for Smart Industry: The Development of the HORSE Architecture, Independently Published, 2021.

[27] R. Zanetti, "OPIL architecture," L4MS Project, 2018.

[28] M. Zimniewicz, "Data flow in pilot scenarios (internal report)," SHOP4CF, 2020.

[29] "FIWARE - Data models," [Online]. Available: https://fiware-datamodels.readthedocs.io/. [Accessed 1 December 2020].

[30] "OS-level virtualization - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/OS-level_virtualization. [Accessed 1 December 2020].

[31] "Docker," [Online]. Available: https://www.docker.com/. [Accessed 1 December 2020].

[32] S. Krakowiak, "What's middleware?," [Online]. Available: http://www.middleware.org/whatis.html. [Accessed 1 December 2020].

[33] "FIWARE - Developers Catalogue," [Online]. Available: https://www.fiware.org/developers/catalogue/. [Accessed 1 December 2020].

[34] V. Araujo, K. Mitra, S. Saguna and C. Åhlund, "Performance evaluation of FIWARE: A cloud-based IoT platform for smart cities," *Journal of Parallel and Distributed Computing,* no. 132, 2019.

[35] "Orion-LD," [Online]. Available: https://github.com/FIWARE/context.Orion-LD. [Accessed 1 December 2021].

[36] "NGSI-LD," [Online]. Available: https://en.wikipedia.org/wiki/NGSI-LD. [Accessed 1 December 2021].

[37] "Smart Data Models," [Online]. Available: https://github.com/smart-data-models/. [Accessed 1 December 2021].

[38] "Smart Data Models - ManufacturingMachine," [Online]. Available: https://github.com/smart-data-models/dataModel.ManufacturingMachine. [Accessed 13 October 2022].

[39] "FIWARE-NGSI v2 Specification," 2018. [Online]. Available: https://telefonicaid.github.io/fiware-orion/api/v2/stable/. [Accessed 1 December 2020].

[40] "eProsima," [Online]. Available: https://github.com/eProsima/FIWARE-SH. [Accessed 1 December 2021].

[41] "FIROS," [Online]. Available: https://firos.readthedocs.io/en/latest/index.html. [Accessed 1 December 2021].

[42] "FIWARE TRUE Connector," [Online]. Available: https://github.com/Engineering-Research-and-Development/fiware-true-connector/. [Accessed 1 December 2021].

[43] "Connected factories - WPs & Deliverables," [Online]. Available: https://www.connectedfactories.eu/sites/default/files/d1.4_cf2_attachment_0-15.pdf. [Accessed 21 November 2022].

[44] "Eclipse Dataspace Connector," [Online]. Available: https://projects.eclipse.org/projects/technology.edc. [Accessed 26 October 2022].

[45] "JSON-LD 1.1," [Online]. Available: https://www.w3.org/TR/json-ld11/. [Accessed 26 October 2022].

[46] „Industrial Edge," Siemens, [Online]. Available: https://www.siemens.com/global/en/products/automation/topic-areas/industrial-edge.html. [Data uzyskania dostępu: 20 November 2023].

[47] „IE App Publisher - Industrial Edge Documentation,” Siemens, [Online]. Available: https://docs.eu1.edge.siemens.cloud/develop_an_application/ieap/download_the_ie_ap.html. [Data uzyskania dostępu: 25 September 2023].

[48] „eProsima / Integration-Service,” [Online]. Available: https://github.com/eProsima/Integration-Service. [Data uzyskania dostępu: 25 September 2023].

[49] P. Grefen, I. Vanderfeesten and G. Boultadakis, "Architecture design of the HORSE hybrid manufacturing process control system," 2016.

[50] P. Bouklis, M. Bruch, G. Beruvides, A. Arenillas, N. Knezevic and G. Ricque, "Deliverable D5.4: Definition of FSTP experiments," SHOP4CF, 2021.

Appendix A Interfaces in the high-level logical software architecture

The information exchanged between the six subsystems and the databases of the high-level logical software architecture from Figure 8 is detailed here.

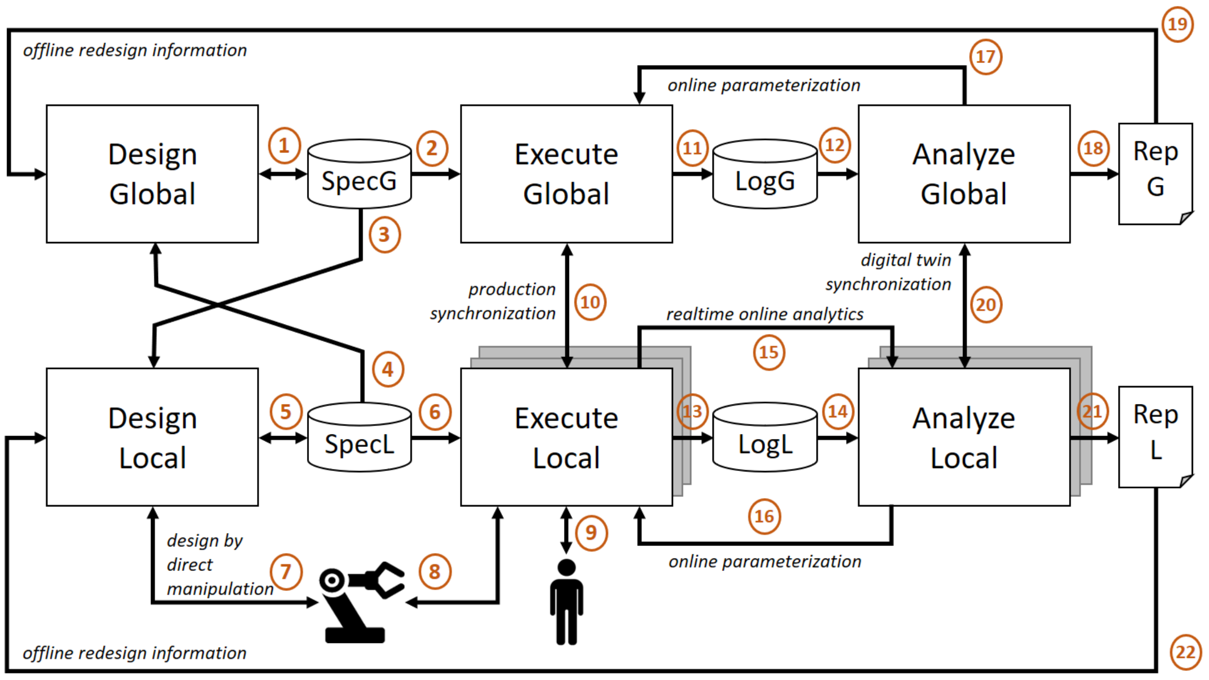

Figure 42 High-level logical software architecture with numbered interfaces

Figure 42 repeats that previous view and contains the identifiers of interfaces. All the interfaces together with high-level categories of information carried between subsystems are listed in Table 6.

Table 6 Interfaces in the high-level logical software architecture

Interface	Direction	Information	Remarks
1	DG → SpG	Process models (sequence of tasks) Agent models (incl. capabilities) Allocation models (role models) Shop Floor Models	
	SpG → DG	(as above)	
2	SpG → EG	(as above)	
3	SpG → DL	Capability models Shop Floor Models	
4	SpL → DG	High-level (black-box) characteristics of task definitions High-level (black-box) characteristics of work cell definitions	

5	DL → SpL	Task and step definitions Safety and risk analysis results	
	SpL → DL	Task and step definitions	
6	SpL → EL	Contents of task (work instructions/scripts) Safety and risk analysis results	
7	DL → Autonomous agent	Contents of task (work instructions/scripts) Trajectories/Movements	
	AA → DL	Design feedback	
8	EL → AA	Contents of task (work instructions/scripts)	
	AA → EL	Task control confirmations Task statuses	
9	EL → Human worker	Contents of task (work instructions)	
10	EG → EL	Task control commands Product definitions	
	EL → EG	Task control confirmations Task statuses Agents statuses (availability/positioning) Alerts Measurements	
	HA → EL	Task control confirmations Task statuses	
11	EG → LgG	(Global) Execution data/logs (Global) Event data/logs	
12	LgG → AG	(as above)	
13	EL → LogL	(Local) Execution data/logs (Local) Event data/logs Factory topography Agents movements Video streams/Images Measurements	
14	LogL → AL	(as above)	
15	EL → AL	(Local) Execution data/logs (Local) Event data/logs	Synchronous interface
16	AL → EL	(Local) Analyzed execution/event data	Synchronous interface
17	AG → EG	(Global) Analyzed execution/event data (Global) Simulation data	Synchronous interface
18	AG → RepG	(Global) Analyzed execution/event data	
19	RepG → DG	(Global) Analyzed execution/event data for redesign	
20	AG → AL	(Global) Analyzed execution/event data for local analysis	
	AL → AG	(Local) Analyzed execution/event data for global analysis	

21	AL → RepL	(Local) Analyzed execution/event data	
22	RepL → DL	(Local) Analyzed execution/event data for redesign	

Appendix B FIWARE data representation

This Appendix is the K4+1 development view on the UT5 data aspect. It provides the technical representation for the SHOP4CF data models defined in Section 8. Reviewing Section 8 may be necessary to understand the technical representation better.

B.1. NGSI format

Data in FIWARE is represented in the FIWARE NGSI format[2]. There are two leading versions of the format: NGSI v2 and NGSI-LD. The SHOP4CF consortium chose NGSI-LD.

In addition, some additional conventions are defined in the following section.

B.2. Conventions

Beyond the strict rules defined by FIWARE NGSI, SHOP4CF defines the following convention.

An entity identifier should be a URN[3], built as "urn:ngsi-ld:<entity-type>:<factory-id>:<entity-id>", for instance: "urn:ngsi-ld:Device:company-xyz:sensor-abc-12345".

The <factory-id> element is introduced to ensure easy adoption of hypothetical future scenarios of smart supply chains, i.e. exchanging data via FIWARE across factories.

B.3. Examples

Example FIWARE entities are provided and kept as online resource to ease its maintenance and contributions from the community. This is available under the following address:

https://shop4cf.github.io/communication-docs/data-models/

[2] https://fiware-datamodels.readthedocs.io/en/latest/howto/index.html
[3] https://en.wikipedia.org/wiki/Uniform_Resource_Name

Appendix C Pilot questionnaires on MES

C.1 Survey and results

To understand the needs of the pilots and needs for interoperability of the SHOP4CF architecture with existing infrastructures, an open-end question survey was conducted. The survey was integrating questions regarding both general Manufacturing Execution System (MES) information and capabilities of connection as attached below.

Detailed results are not here reported due to the confidential data communicated by the partners.

However general anonymized results of the four partners are here reported and summarized in Table 7. Two main points were obtained from the user survey. More than half of the participants were open for data sharing and MES had supporting functionalities for exporting data.

Table 7 High-level results for the pilot questionnaires

	Yes	**No**
Open to data sharing	75%	25%
Established supplier connection	75%	25%
MES supports APIs	75%	25%

Three KPIs were extracted from the questions. Percentage of the pilots supporting or not the three KPIs are reported. However, each MES had different interfaces. Therefore, tuning on the exchanged data needs to be properly customized.

C.2 Questionnaire questions

Company background on MES

1. What MES system is used in your company?
2. Does your MES system supports multi-language?
3. What type of functionalities does your MES support?
   - [ ] Operations/Detailed Sequencing
   - [ ] Resource Allocation and Status
   - [ ] Document Control
   - [ ] Performance Analysis Process management
   - [ ] Data collection & acquisition
   - [ ] Maintenance management (they use another tool for the Maintenance)
   - [ ] Quality management (in line quality management)
   - [ ] Product tracking and genealogy

- [ ] Labour management (it was added manually as long it was not a default)
- [ ] Dispatching production units (Routing based)

4. Can you please order the previous functions to display how your MES focus on them? Please order them from the lowest important to the highest important.
   - [ ] Operations/Detailed Sequencing
   - [ ] Resource Allocation and Status
   - [ ] Document Control
   - [ ] Performance Analysis Process management
   - [ ] Data collection & acquisition
   - [ ] Maintenance management
   - [ ] Quality management
   - [ ] Product tracking and genealogy
   - [ ] Labour management
   - [ ] Dispatching production units

MES information for ISA-95

5. Are you aware if your MES system is using ISA-95? (skip this question if you replied NO to 4)
6. How are machines defined in the MES system? (i.e. Which name is used?) If they are not defined write NO
7. How are human operators defined in the MES system? (i.e. Which name is used?) If they are not defined write NO
8. How are products and sub-products defined in the MES system? (i.e. Which name is used?) If they are not defined write NO
9. How are tools defined in the MES system? (i.e. Which name is used?) If they are not defined write NO

Usage of the MES system

10. How machines communicate to the MES system and vice versa? (e.g .file transfer via USB, file transfer via TCP, etc..)
11. How humans communicate to the MES system and vice versa? (e.g. file transfer via USB, file transfer via TCP, HMI, etc..)
12. How tools communicate to the MES system and vice versa? (e.g. file transfer via USB, file transfer via TCP, etc..)
13. How products' status and sub-products' status communicate to the MES system and vice versa? (e.g. sensor connected, RFID etc...)
14. How do you insert an order in the system (e.g. produce 100 washing machines)?
15. How the order tracking is done (e.g. status)? (i.e. Who or what marks the order as done?)
16. If you change your physical layout of your factory (new line/new machine) how this is reflected in the MES?
17. How is the communication between MES and ERP done?
18. How much paper-based documentation is used at the factory? Is there any process for including paper-based documentation in the MES/ERP system?
19. Does your MES system supports APIs or Interfaces to integrate communication with other components? If yes please specify which one (e.g. MQTT, OPC UA)? If not, which other methods can be used to retrieve data from MES?

20. How does the MES handle alerts/exceptions?

MES information for lifecycle and layer management

21. Do you track the life (“vita”) of an asset (e.g. production, usage, maintenance)? If yes how this is reflected in your MES
22. How is the connection between suppliers of some sub-products integrated in the MES system? (e.g. if you need to get an electric motor from a supplier how this is integrated and tracked?) If you do not share information skip to 27.
23. If you do no share information with suppliers, why you do not do that? Are there concerns on data integrity or other?
24. If there could be a way to easily transfer/receive the data in a secured way from suppliers would you be interested on using the technology?
25. Do you track the position of your assets (i.e. human, machine, tool, product) with a special hierarchy (i.e. station, control device, site, area, workcell)? If yes how this is reflected in your MES?
26. Do you use the concept of Asset Administration Shell in your MES or IT system? If yes, how is it implemented?

SHOP4CF architecture

27. Can you MES communicate directly with FIWARE (no translation necessary)? If yes, how is it implemented?
28. What is the high-level data model of the concepts dealt by the MES and how are these realized on the exchanged messages with other systems (through the interfaces)?

Appendix D Architecture of FSTP experiments

The SHOP4CF architecture is a common template for concrete systems. Some new systems adopting the architecture are defined via the SHOP4CF open calls and resulting FSTP experiments.

This appendix aims at presenting that the SHOP4CF architecture is being adopted also in such manufacturing scenarios designed outside of the project consortium.

The following subsections present the architecture of FSTP experiments implemented in the all three rounds of the open calls.

D.1 BrainWatch

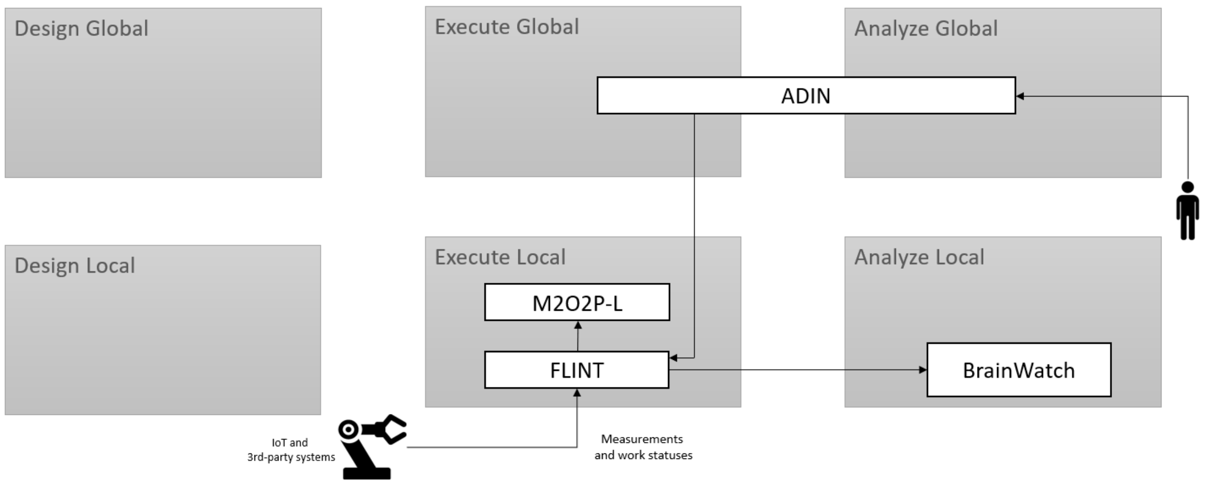

Figure 43 BrainWatch – Logical software architecture

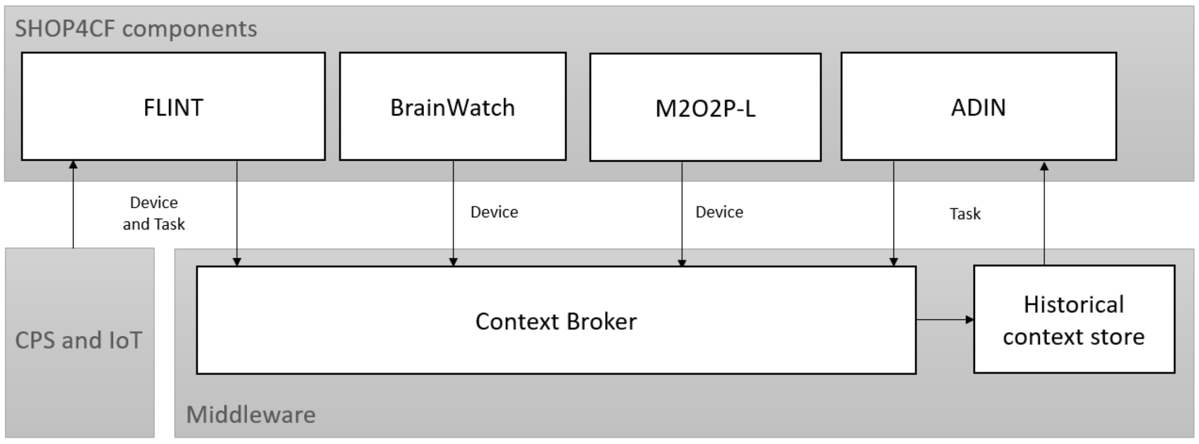

Figure 44 BrainWatch – Logical platform architecture

D.2 ISTSME

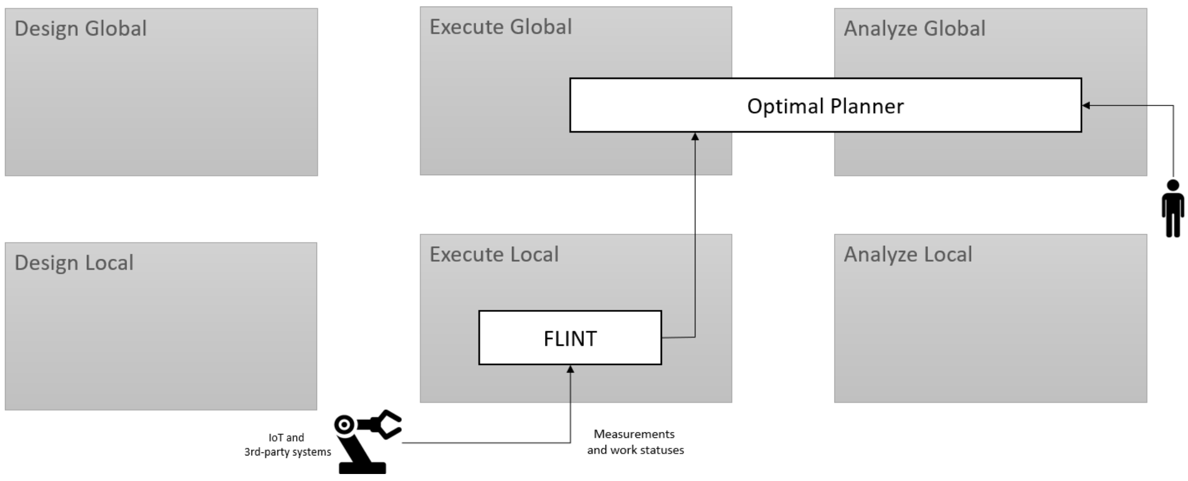

Figure 45 ISTSME – Logical software architecture

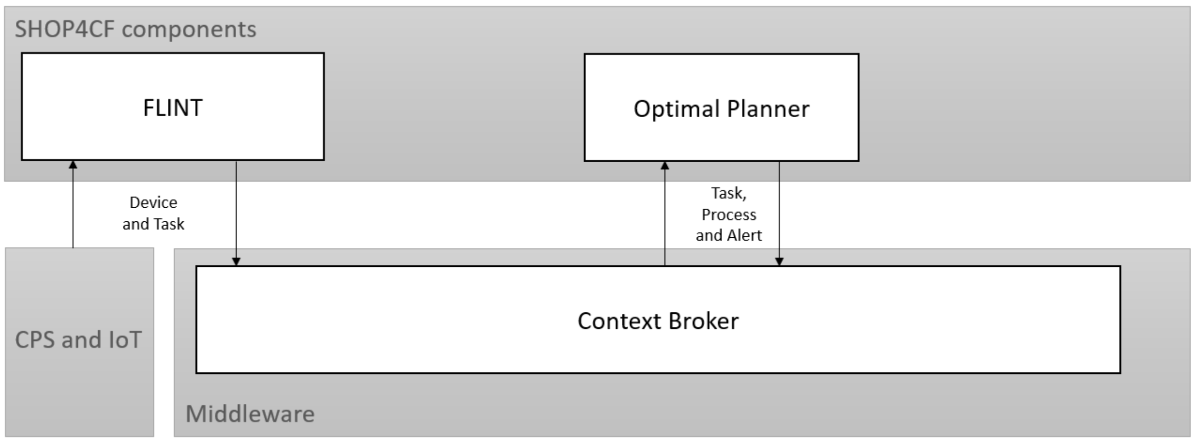

Figure 46 ISTSME – Logical platform architecture

D.3 BUCK

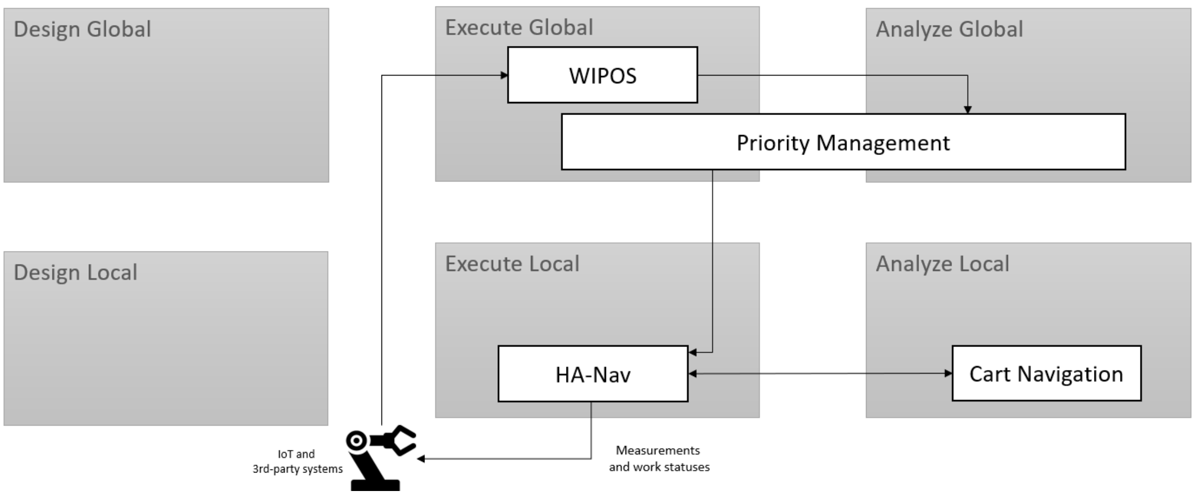

Figure 47 BUCK – Logical software architecture

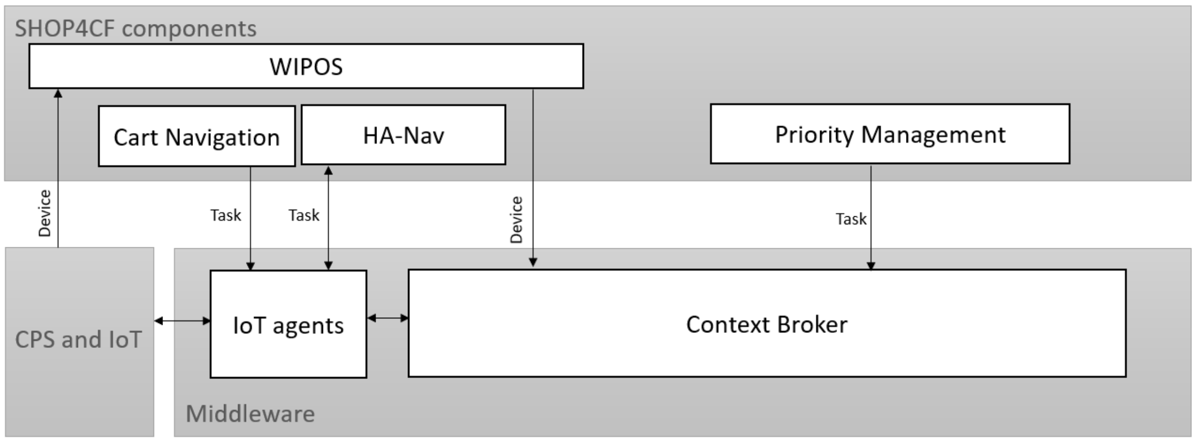

Figure 48 BUCK– Logical platform architecture

D.4 PosWeTool

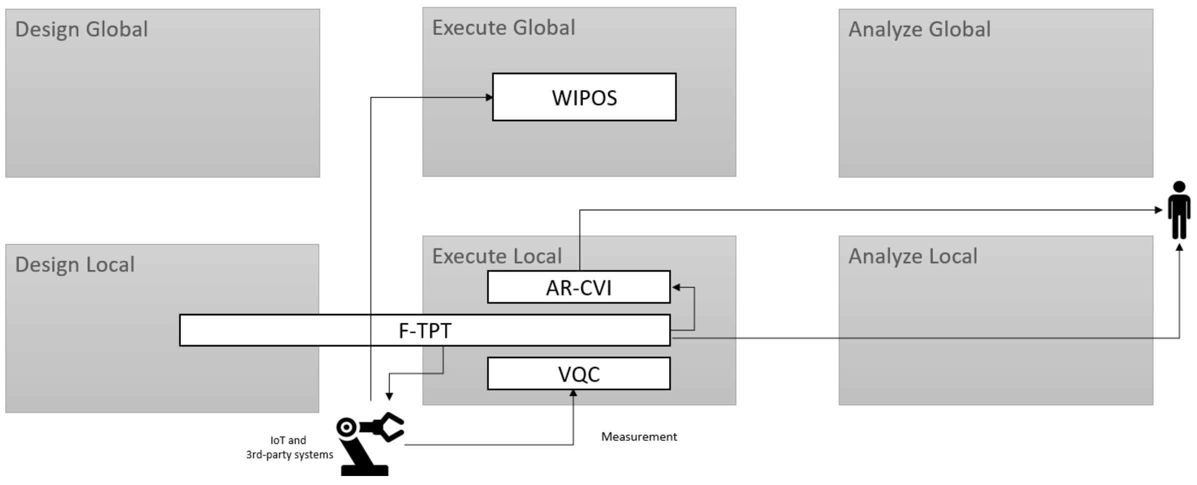

Figure 49 PosWeTool – Logical software architecture

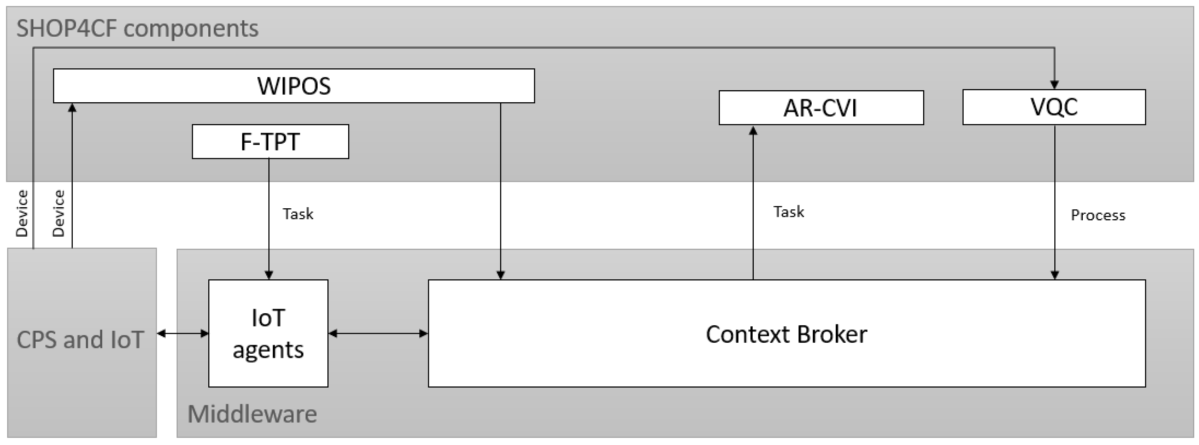

Figure 50 PosWeTool – Logical platform architecture

D.5 MATTRESS

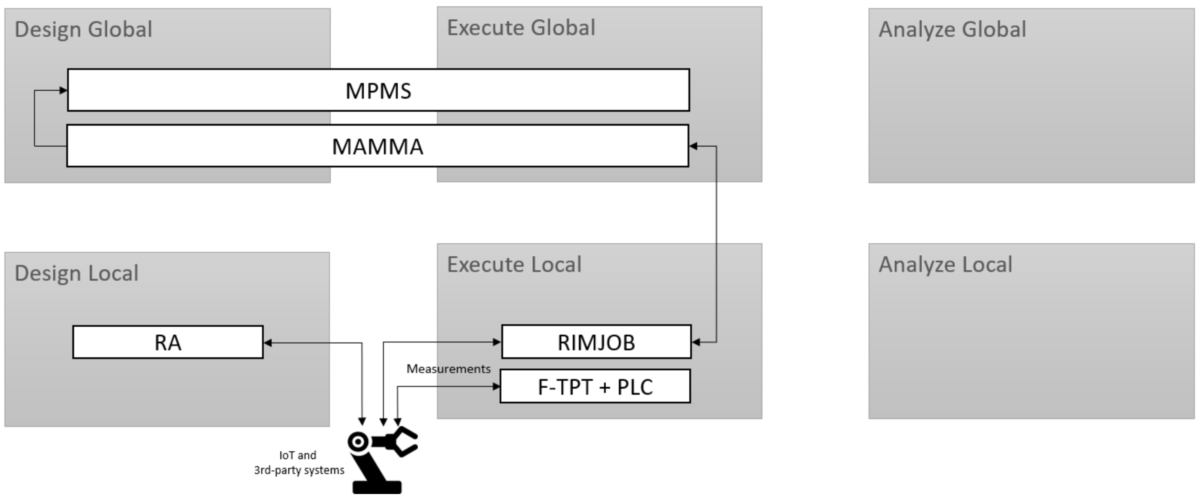

Figure 51 MATTRESS – Logical software architecture

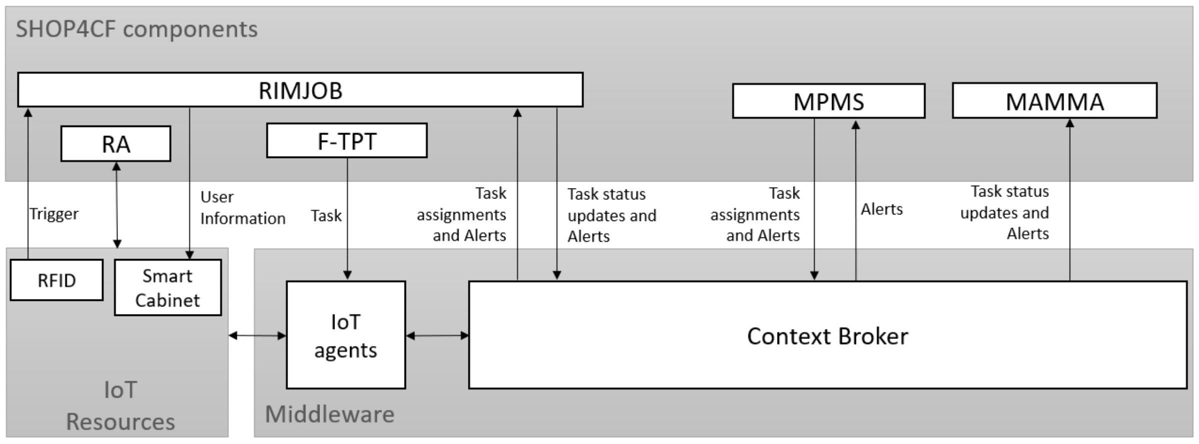

Figure 52 MATTRESS – Logical platform architecture

D.6 ANGEL

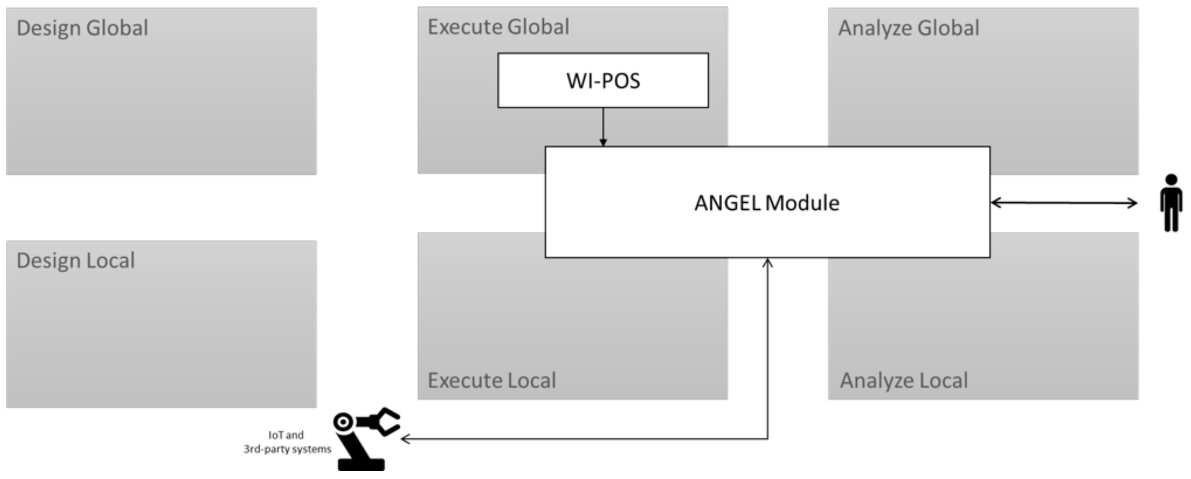

Figure 53 ANGEL – Logical software architecture

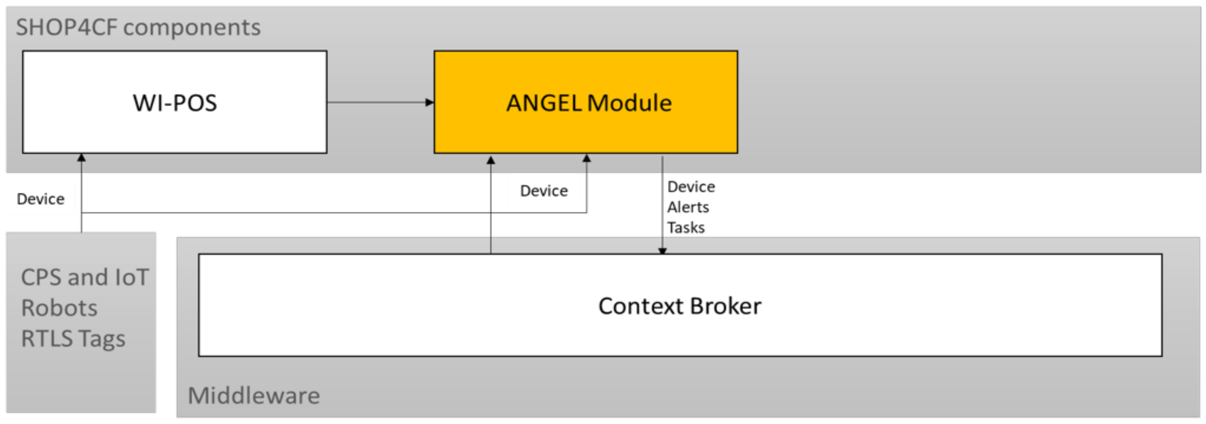

Figure 54 ANGEL – Logical platform architecture

D.7 AI4Dim&SurfQA

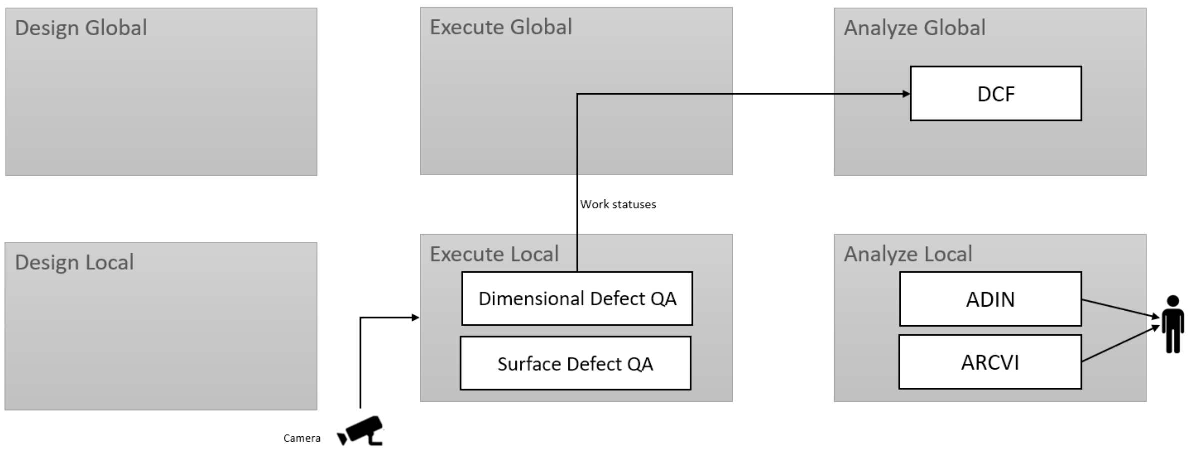

Figure 55 AI4Dim&SurfQA – Logical software architecture

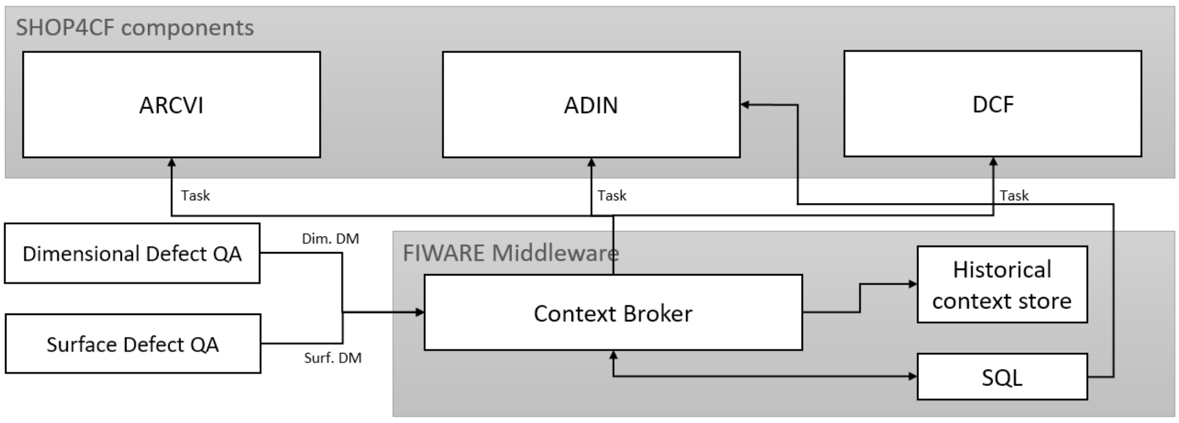

Figure 56 AI4Dim&SurfQA – Logical platform architecture

D.8 ASMOSA

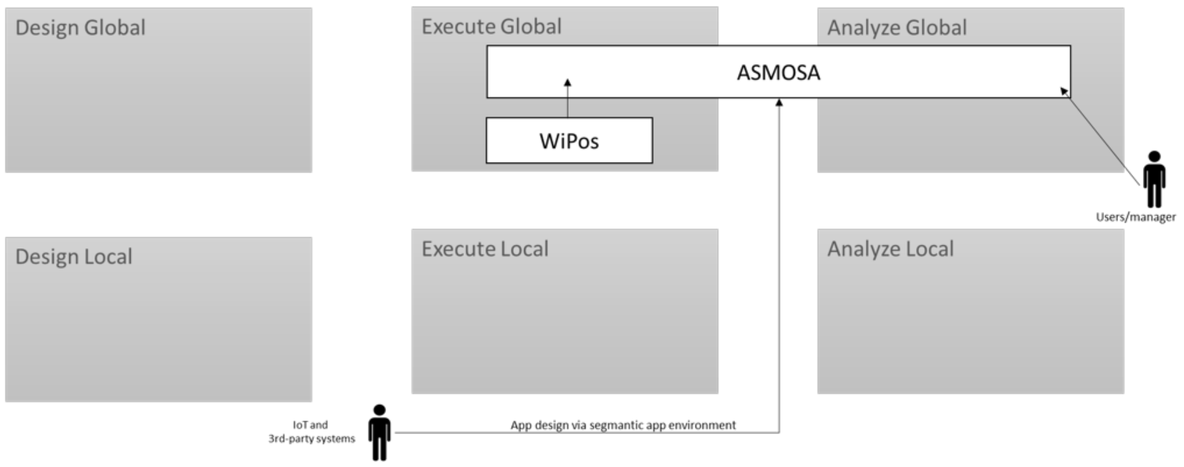

Figure 57 ASMOSA – Logical software architecture

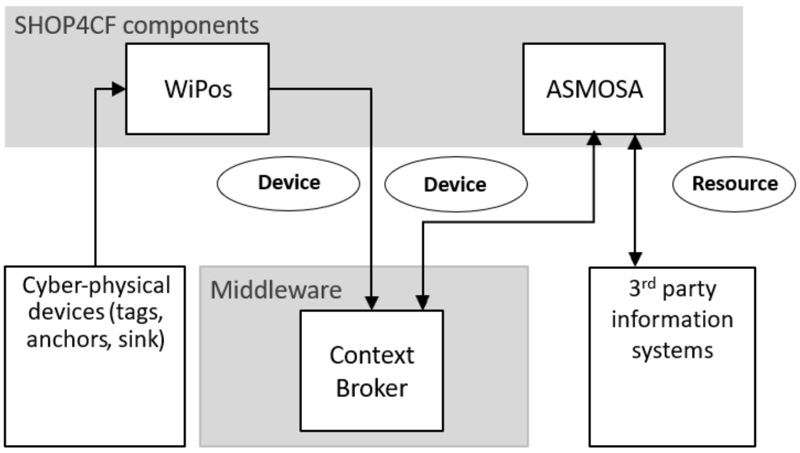
SHOP4CF components
WiPos
ASMOSA
Device
Device
Resource
Cyber-physical devices (tags, anchors, sink)
Middleware
Context Broker
3rd party information systems

Figure 58 ASMOSA – Logical platform architecture

D.9 CO-SCREW

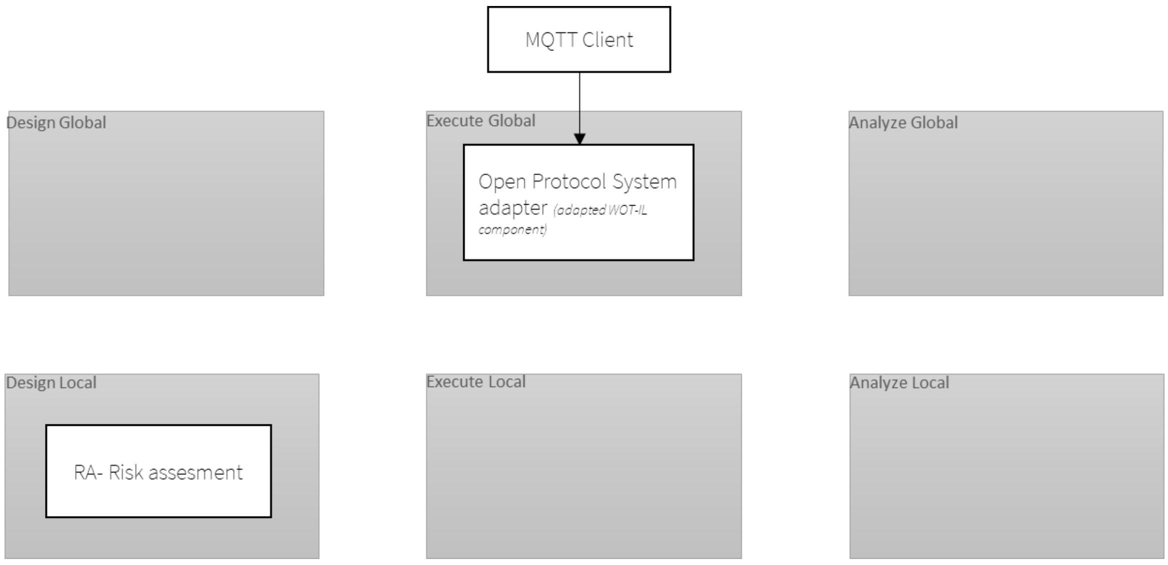

Figure 59 CO-SCREW – Logical software architecture

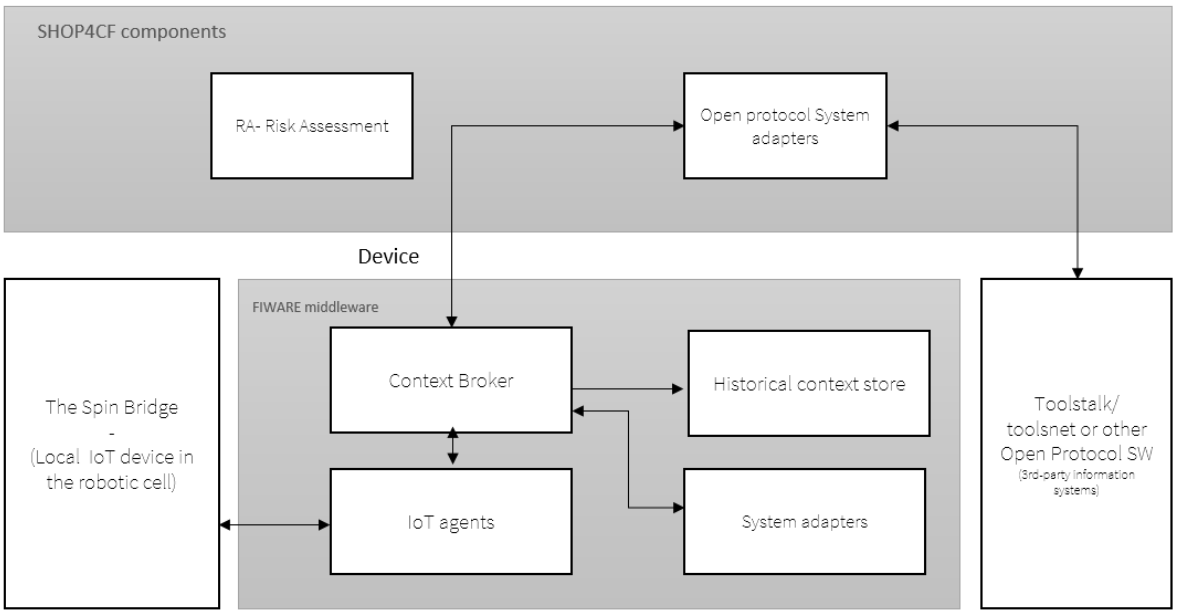
SHOP4CF components
RA- Risk Assessment
Open protocol System adapters
Device
FIWARE middleware
The Spin Bridge - (Local IoT device in the robotic cell)
Context Broker
Historical context store
IoT agents
System adapters
Toolstalk/ toolsnet or other Open Protocol SW (3rd-party information systems)

Figure 60 CO-SCREW – Logical platform architecture

D.10 HOPE_Foreman

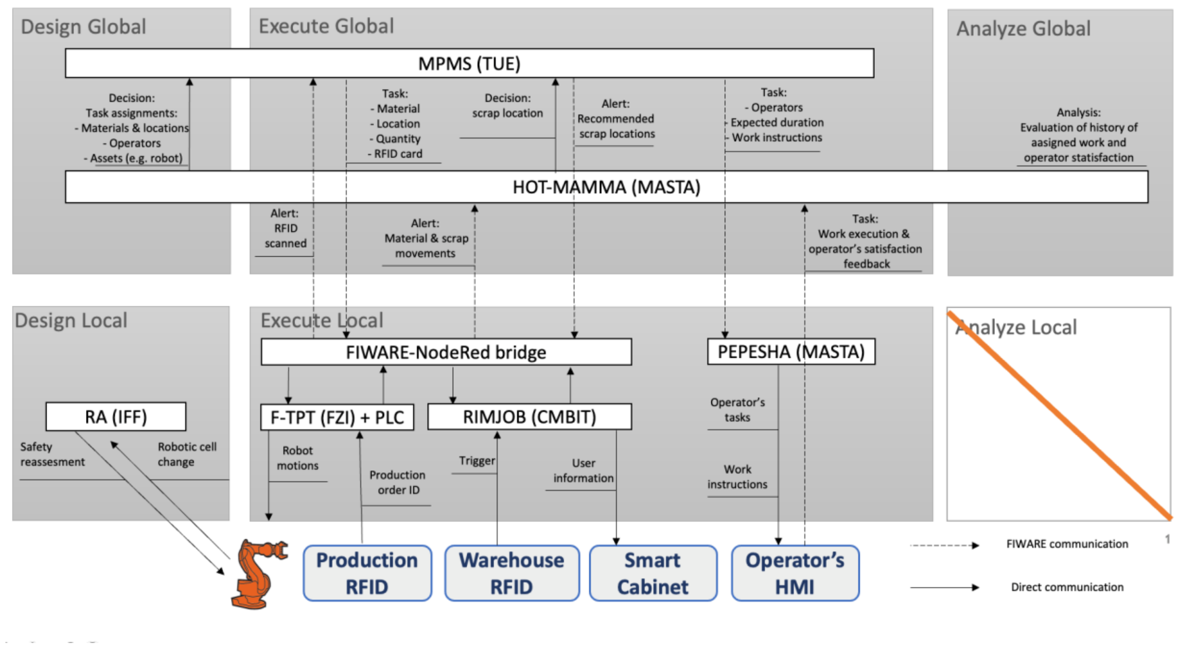

Figure 61 HOPE_Foreman – Logical software architecture

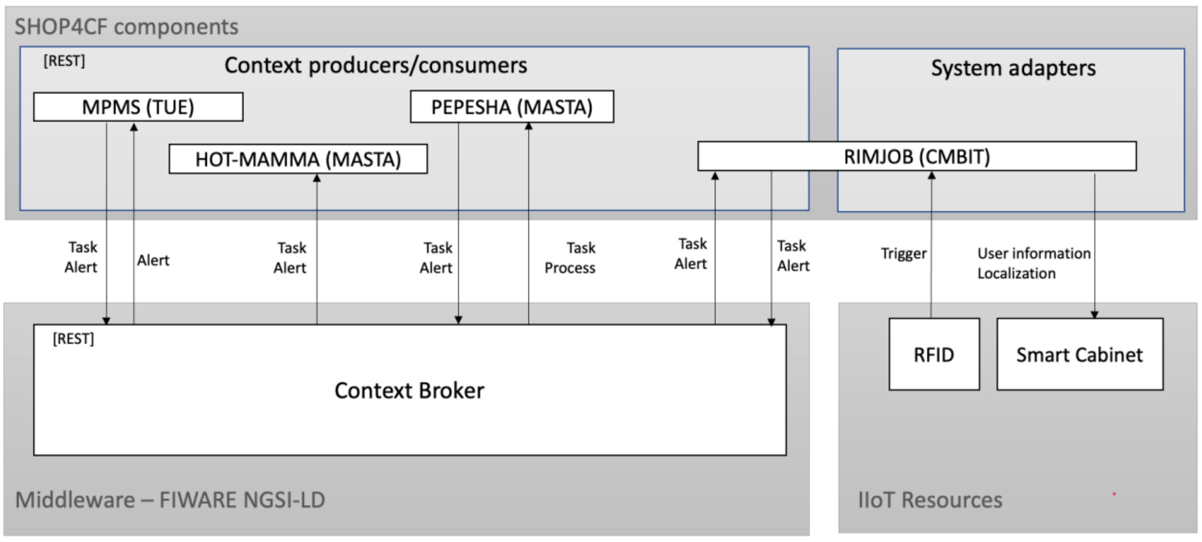

Figure 62 HOPE-Foreman – Logical platform architecture

D.11 IN4STATION

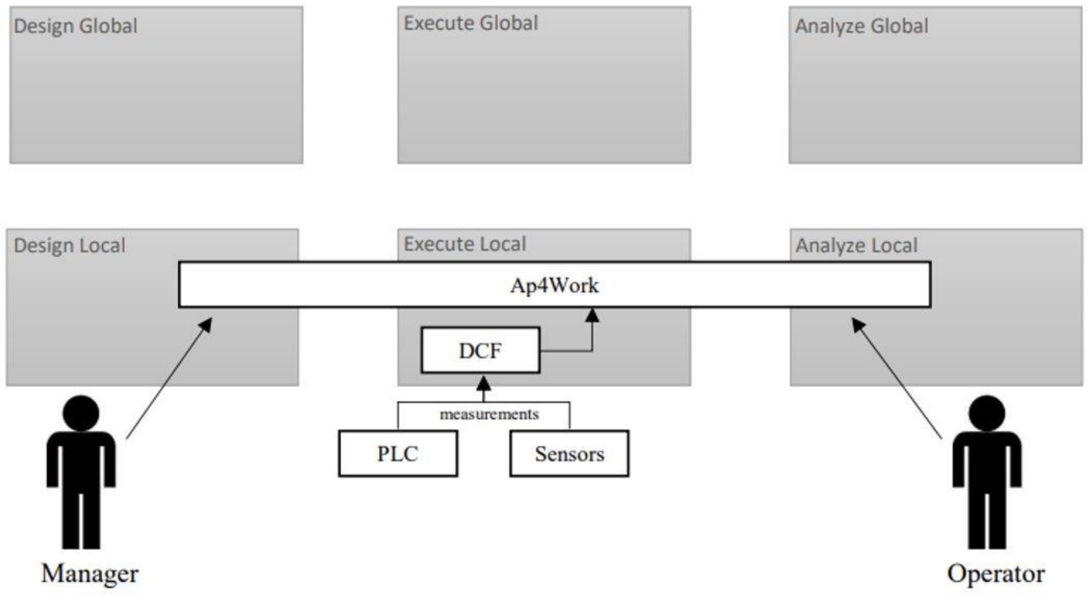

Figure 63 IN4STATION – Logical software architecture

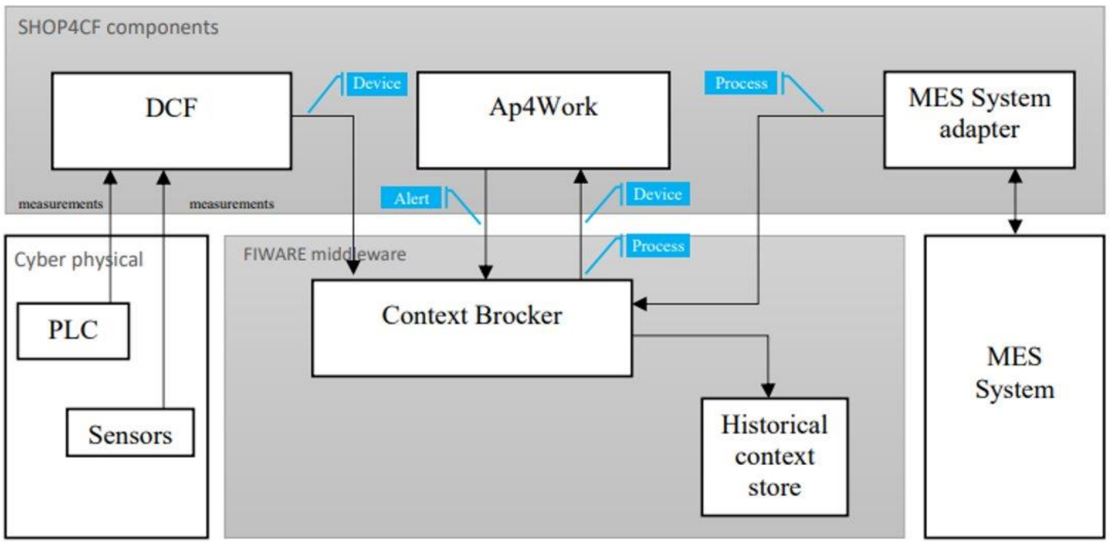

Figure 64 IN4STATION – Logical platform architecture

D.12 MARSH

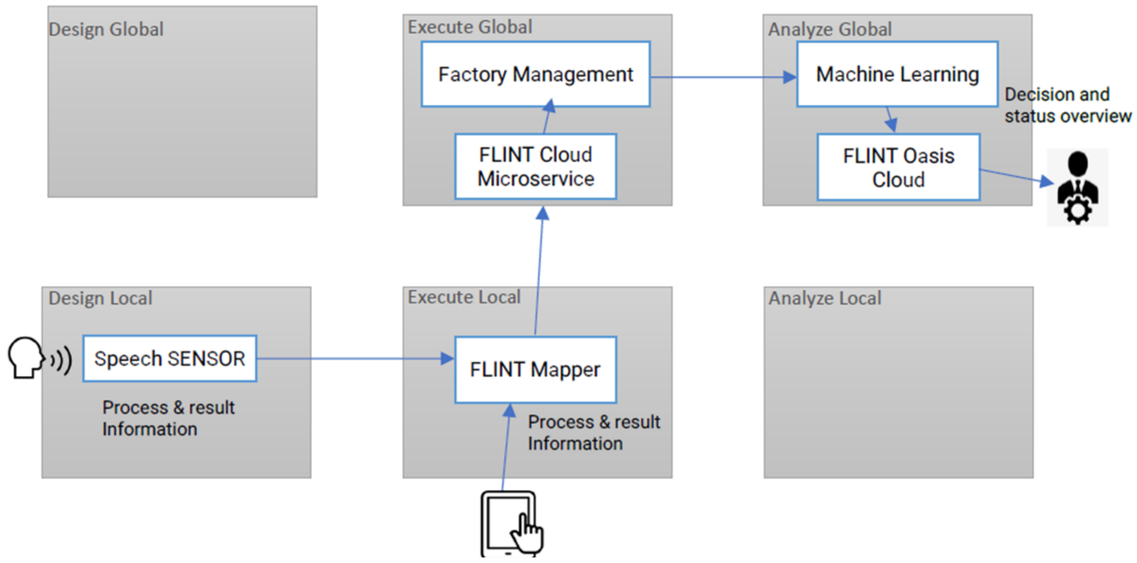

Figure 65 MARSH – Logical software architecture

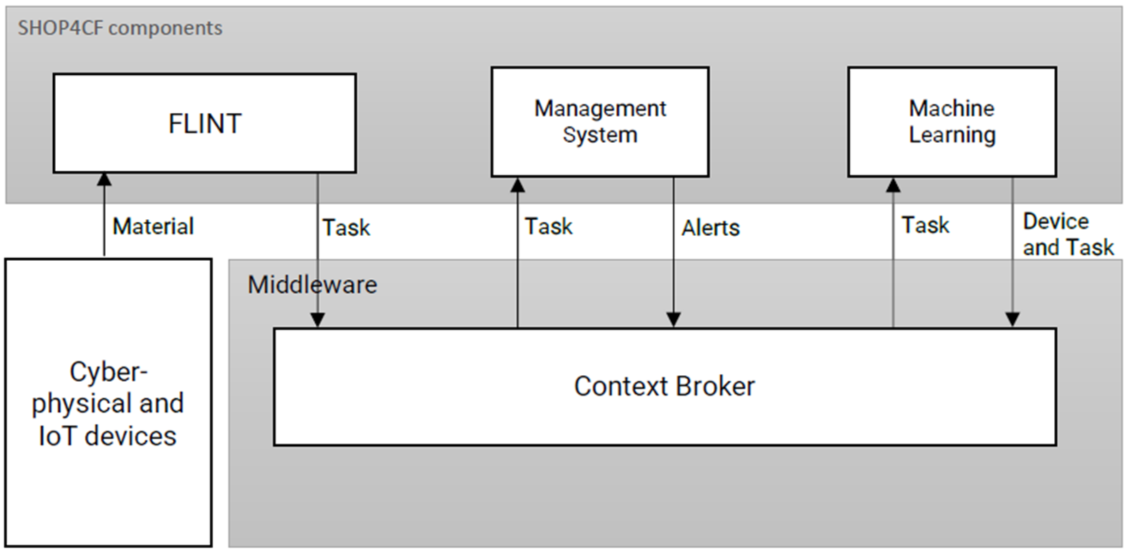

Figure 66 MARSH – Logical platform architecture

D.13 PROPHET

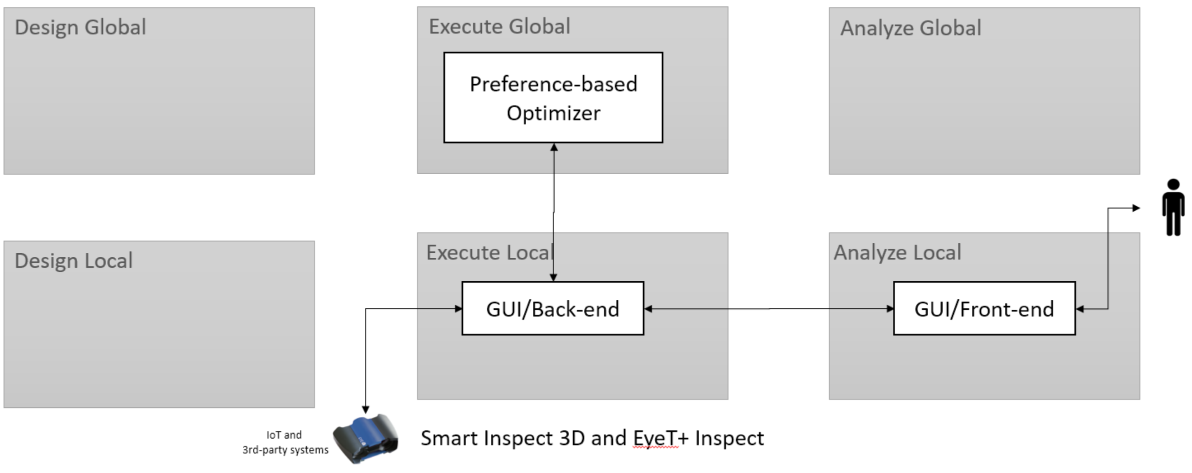

Figure 67 PROPHET – Logical software architecture

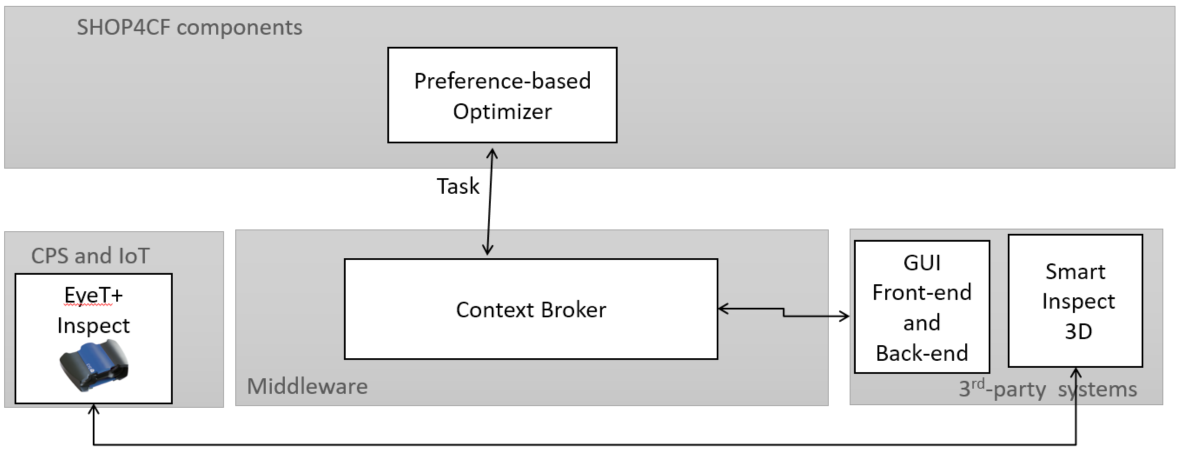

Figure 68 PROPHET– Logical platform architecture

D.14 QPC-AI

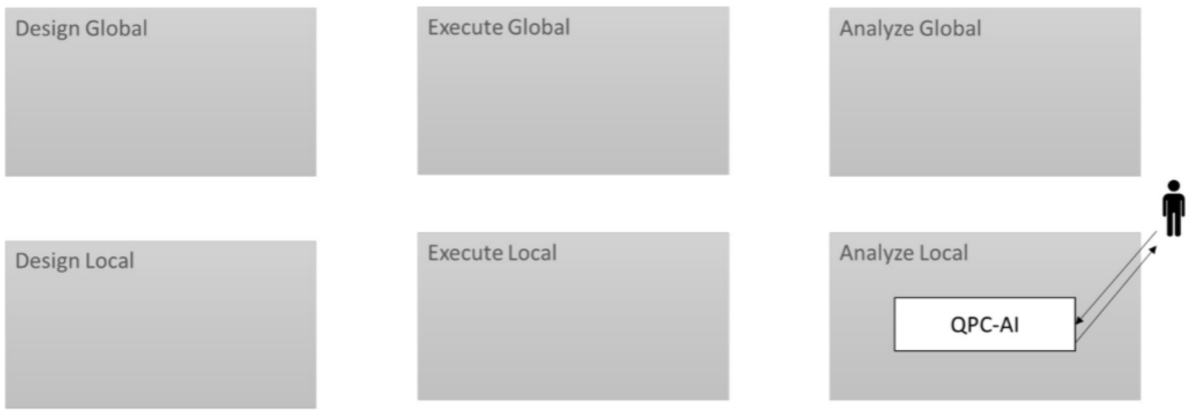

Figure 69 QPC-AI – Logical software architecture

D.15 RASP

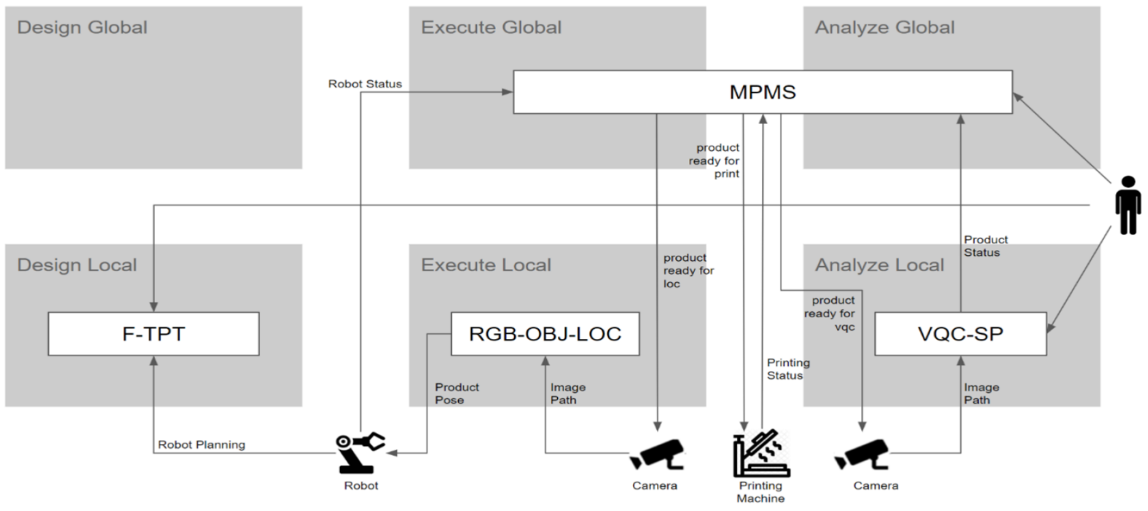

Figure 70 RASP – Logical software architecture

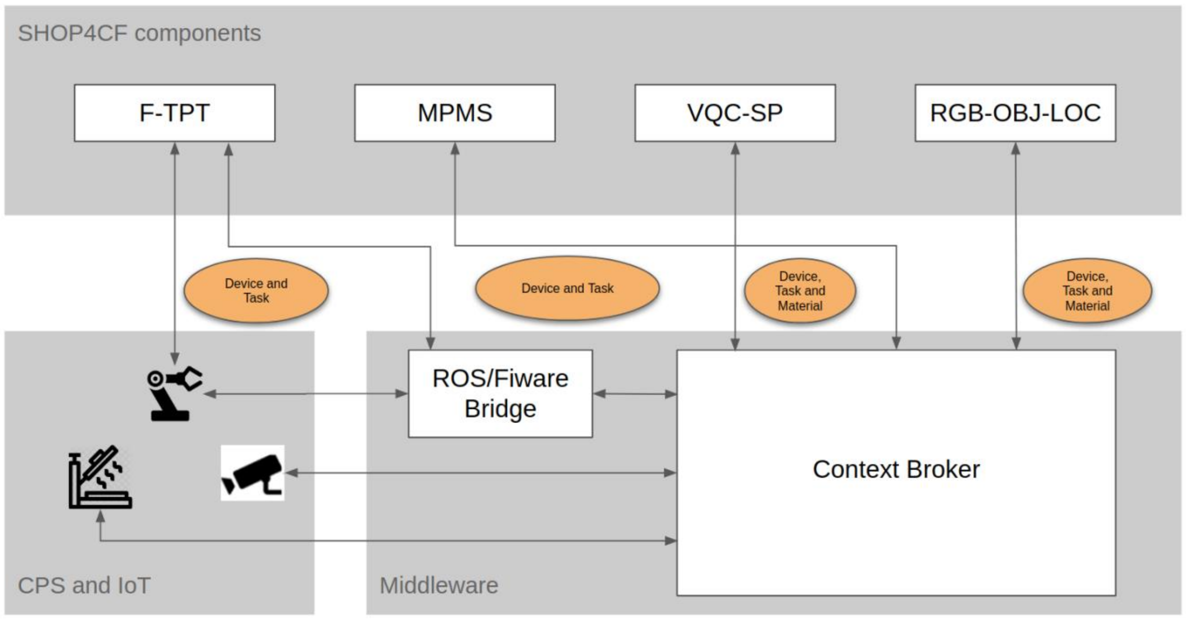

Figure 71 RASP – Logical platform architecture

D.16 RTLS4SHOP

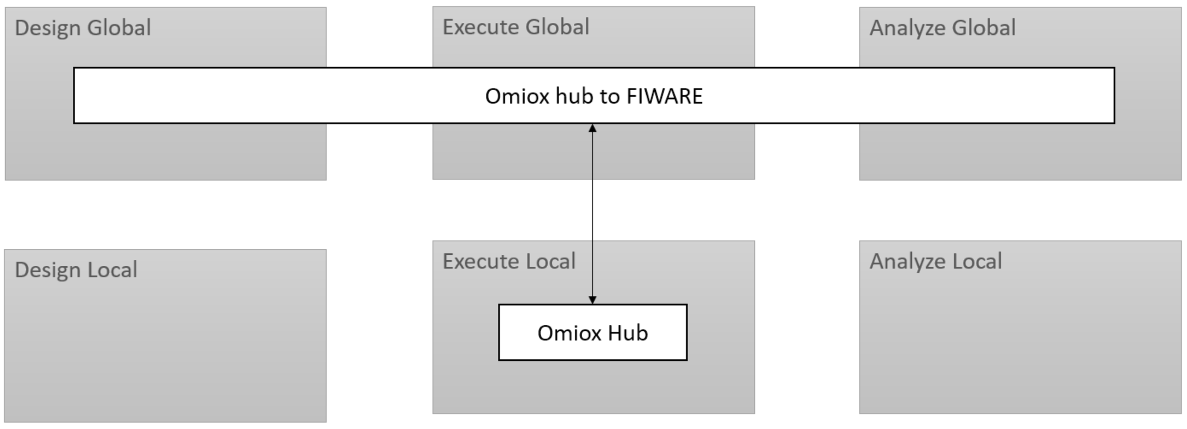

Figure 72 RTLS4SHOP – Logical software architecture

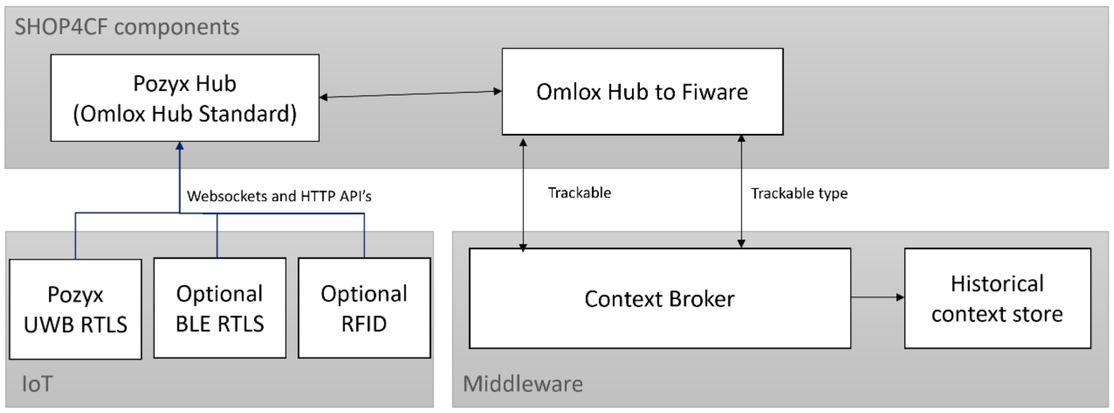

Figure 73 RTLS4SHOP – Logical platform architecture

D.17 SmartPartsDetector

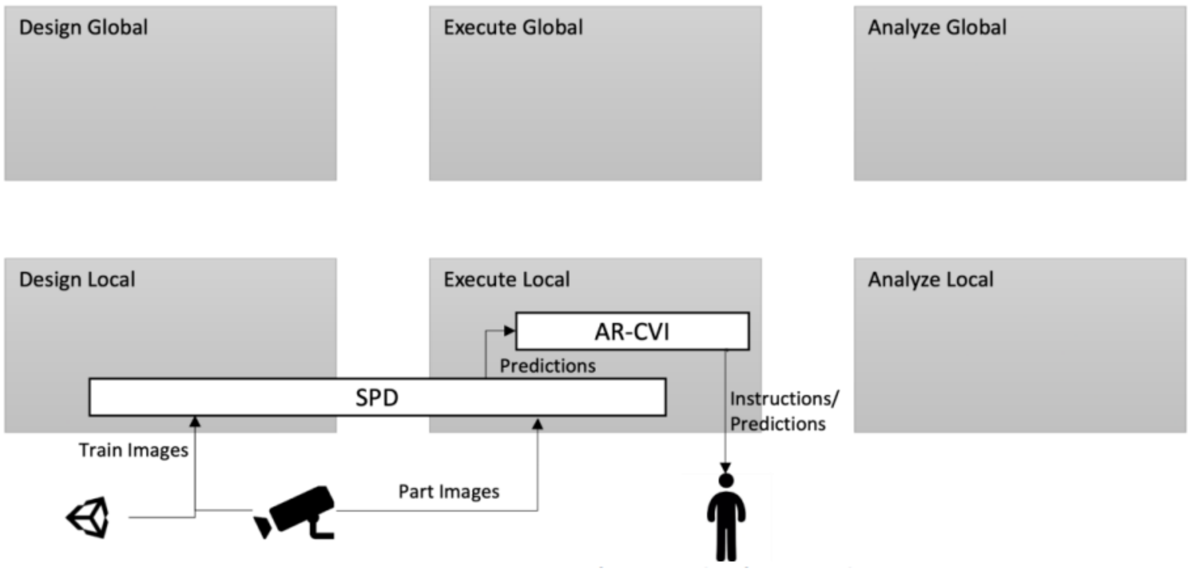

Figure 74 SmartPartsDetector – Logical software architecture

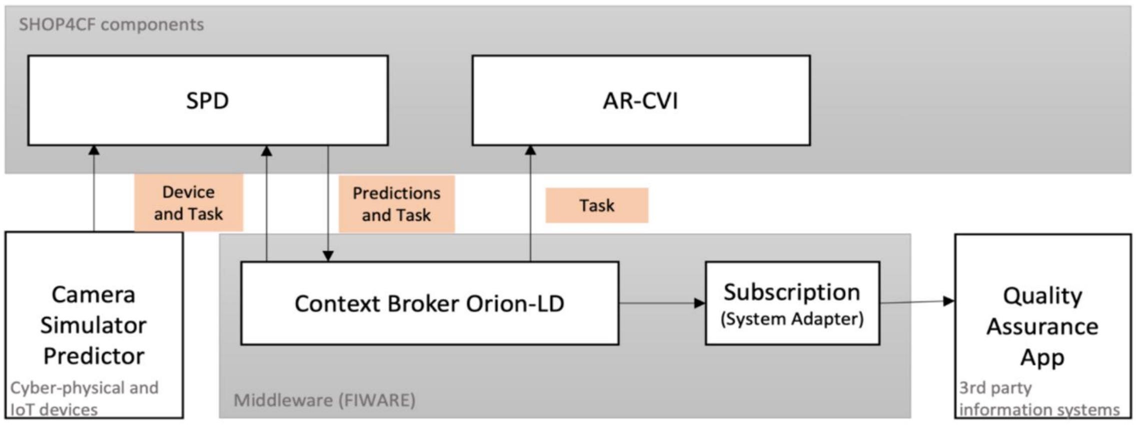

Figure 75 SmartPartsDetector – Logical platform architecture

D.18 SMASH

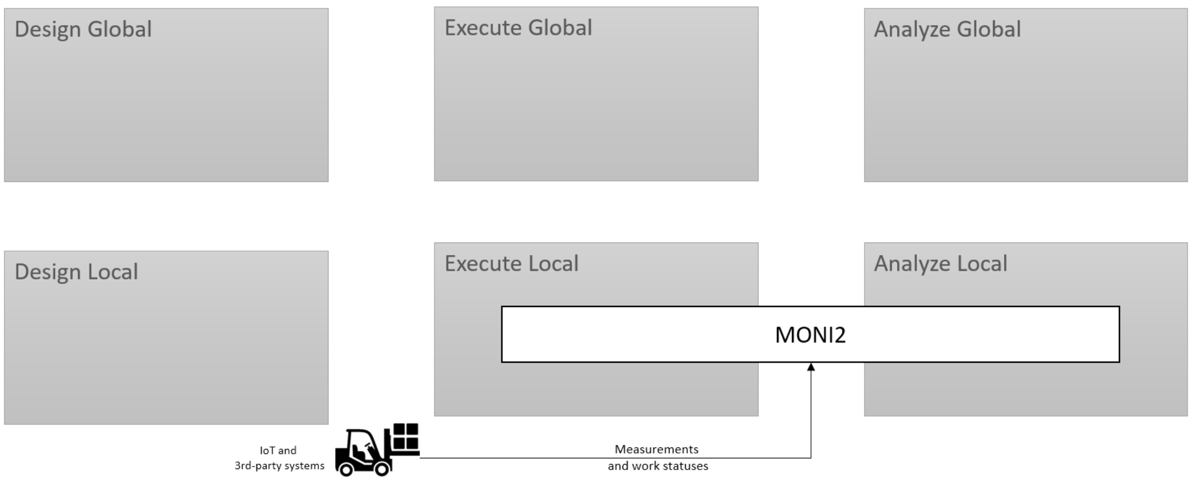

Figure 76 SMASH – Logical software architecture

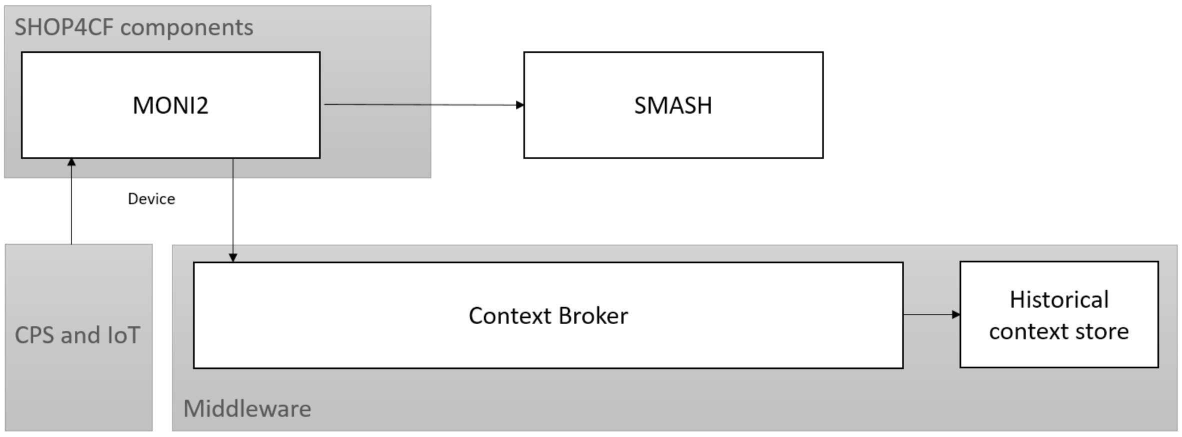

Figure 77 SMASH – Logical platform architecture

D.19 WARNING

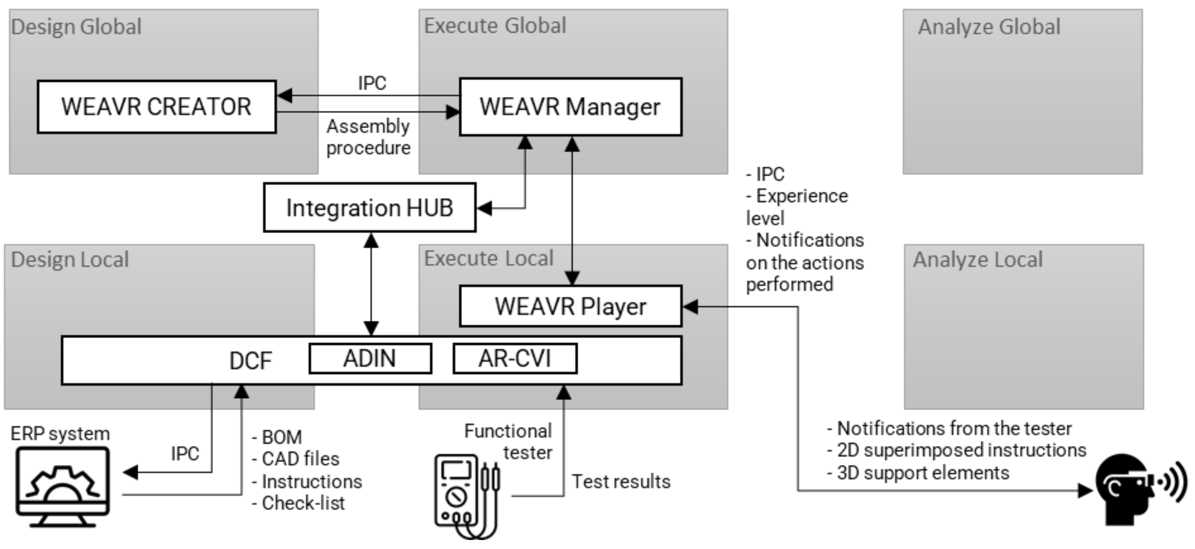

Figure 78 WARNING – Logical software architecture

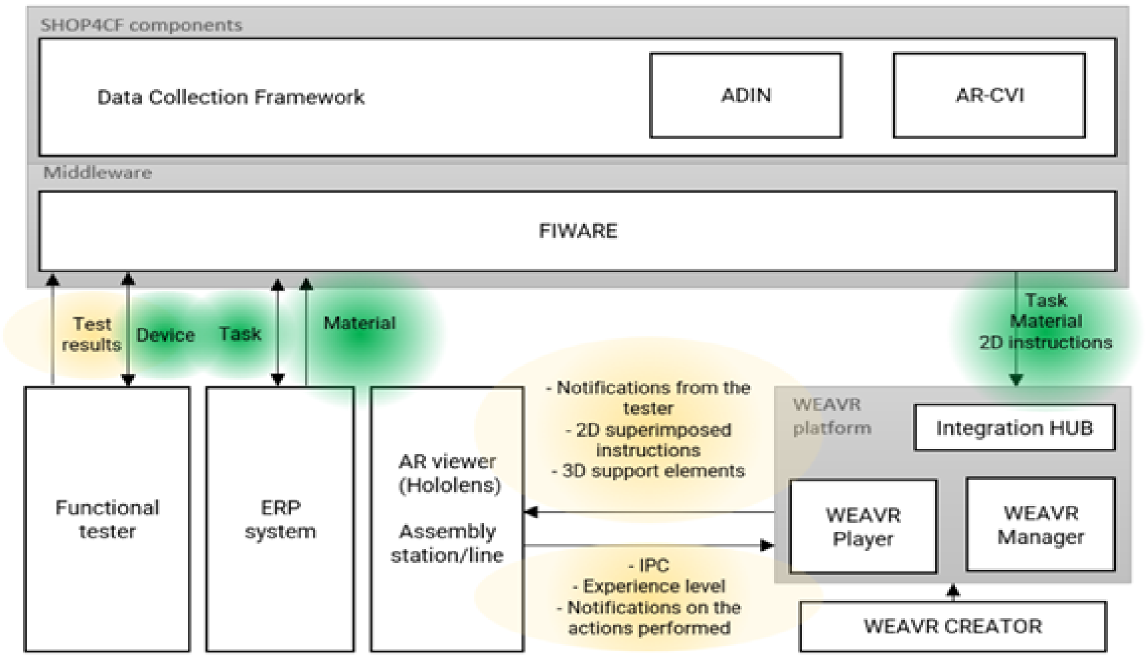

Figure 79 WARNING – Logical platform architecture

D.20 AI4INSPECTION

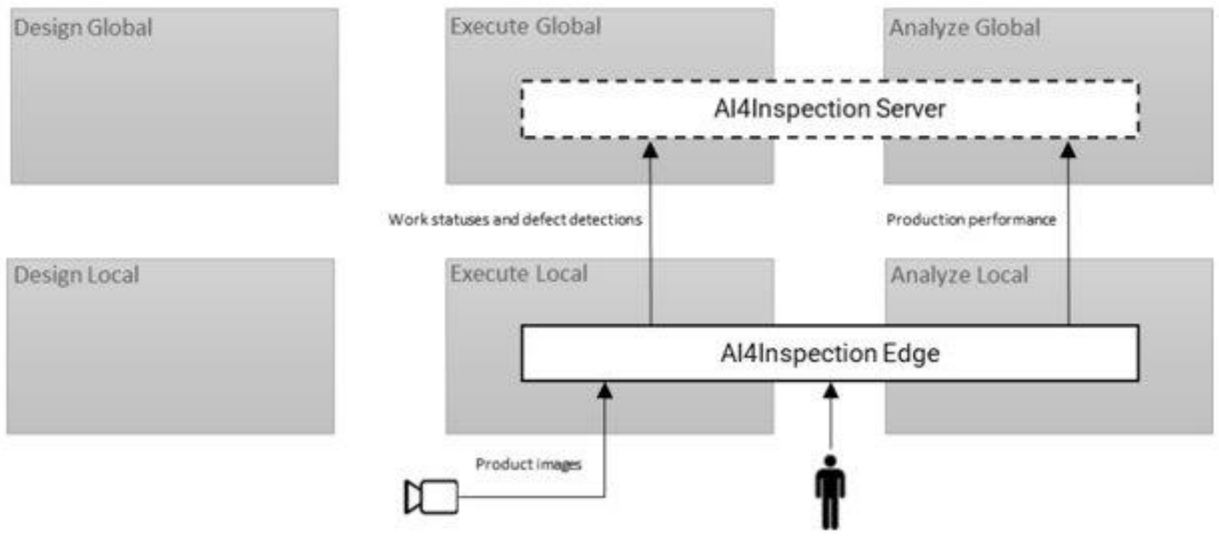

Figure 80 AI4INSPECTION – Logical software architecture

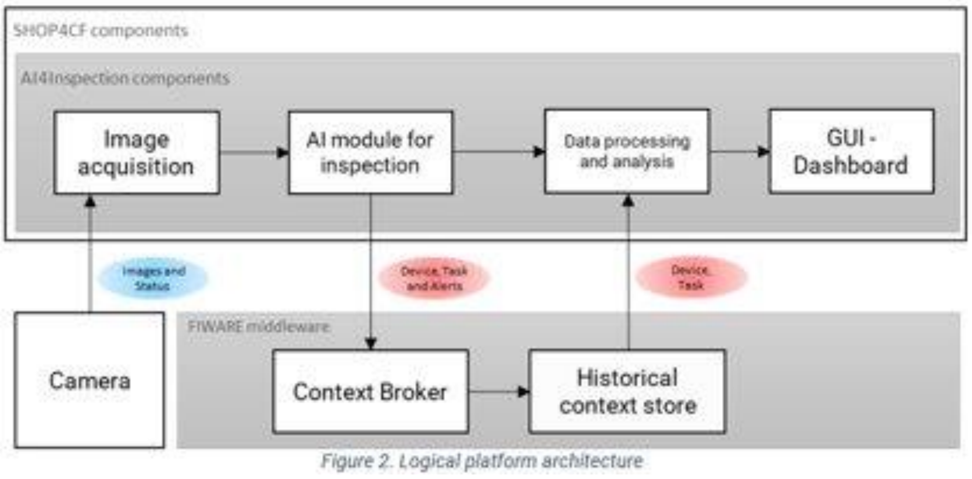

Figure 81 AI4INSPECTION – Logical platform architecture

D.21 ALLOTCODA

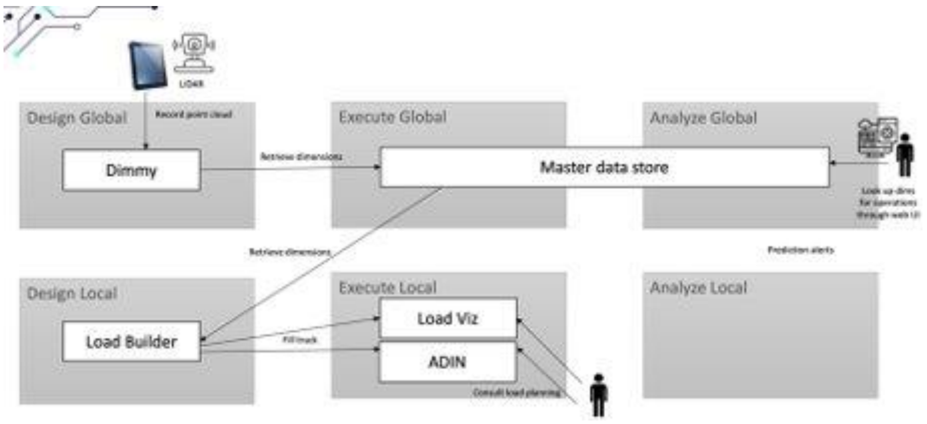

Figure 82 ALLOTCODA – Logical software architecture

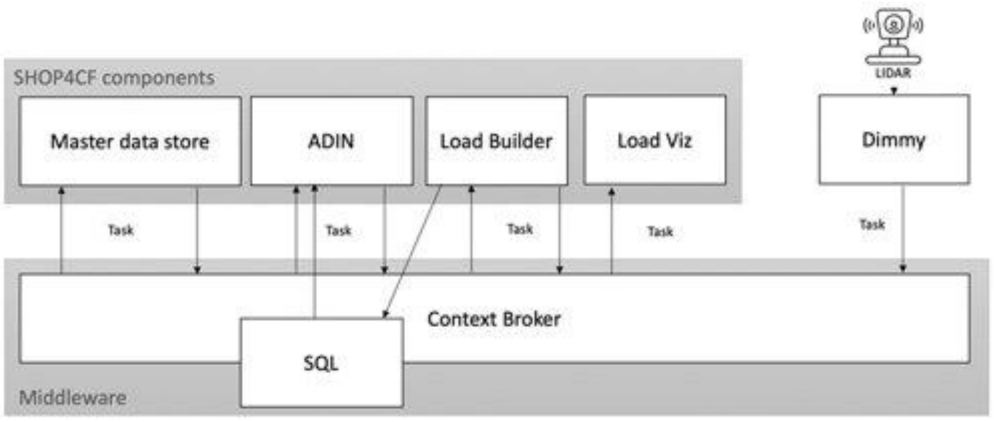

Figure 83 ALLOTCODA – Logical platform architecture

D.22 AR2EMP

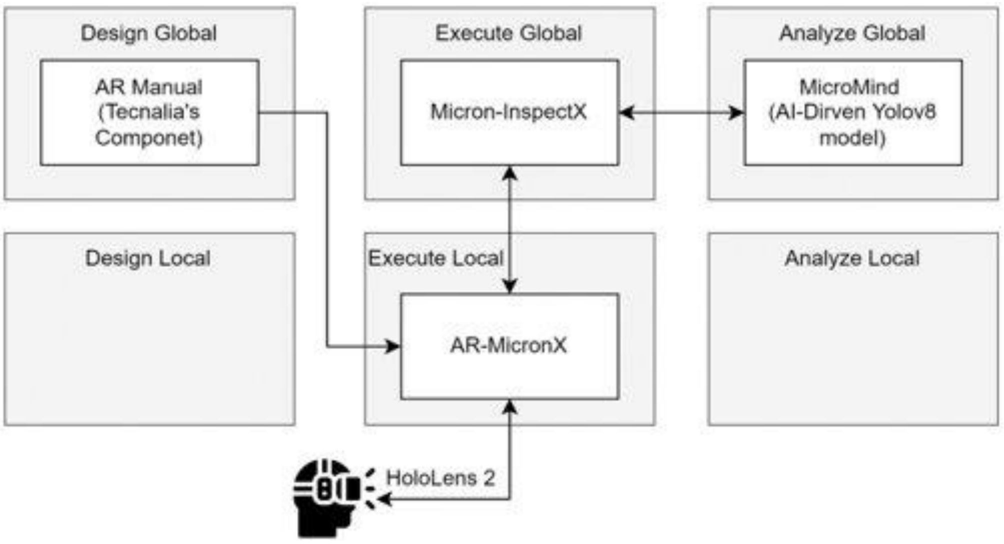

Figure 84 AR2EMP – Logical software architecture

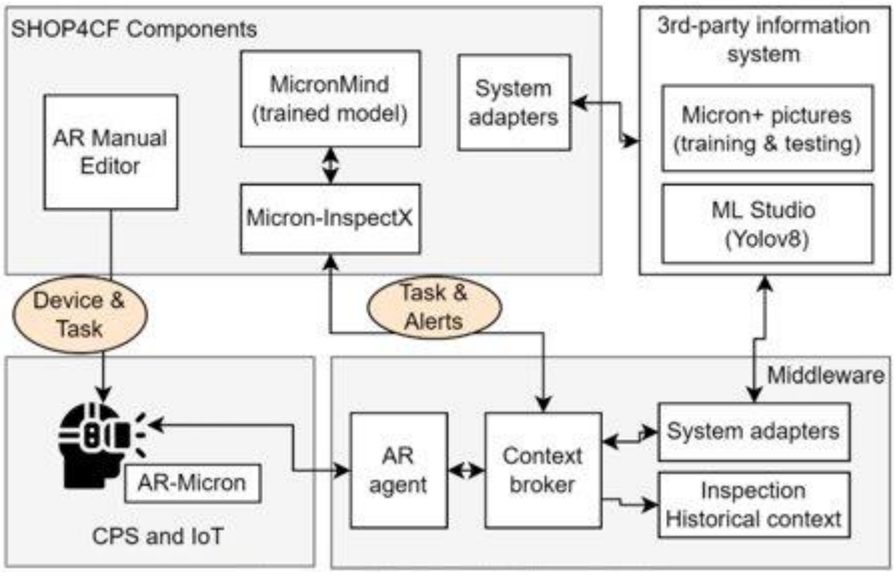

Figure 85 AR2EMP – Logical platform architecture

D.23 CHEEPSH

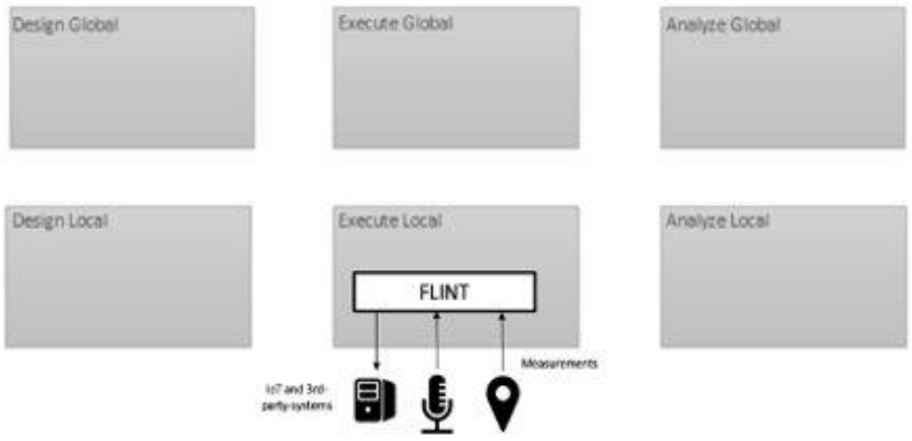

Figure 86 CHEEPSH – Logical software architecture

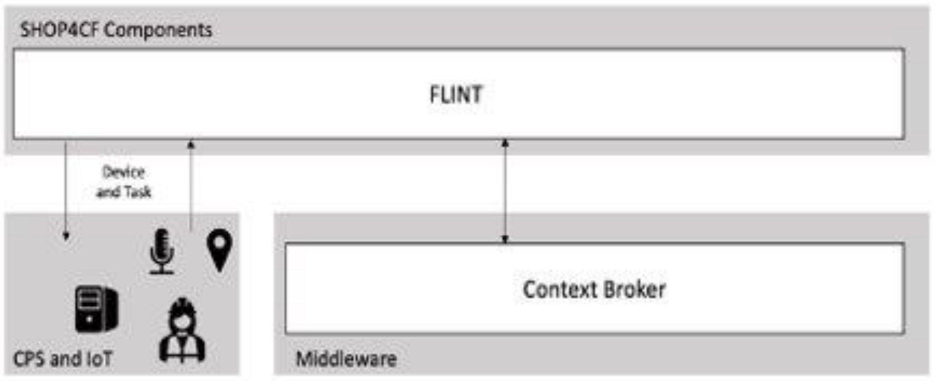

Figure 87 CHEEPSH – Logical platform architecture

D.24 COOP

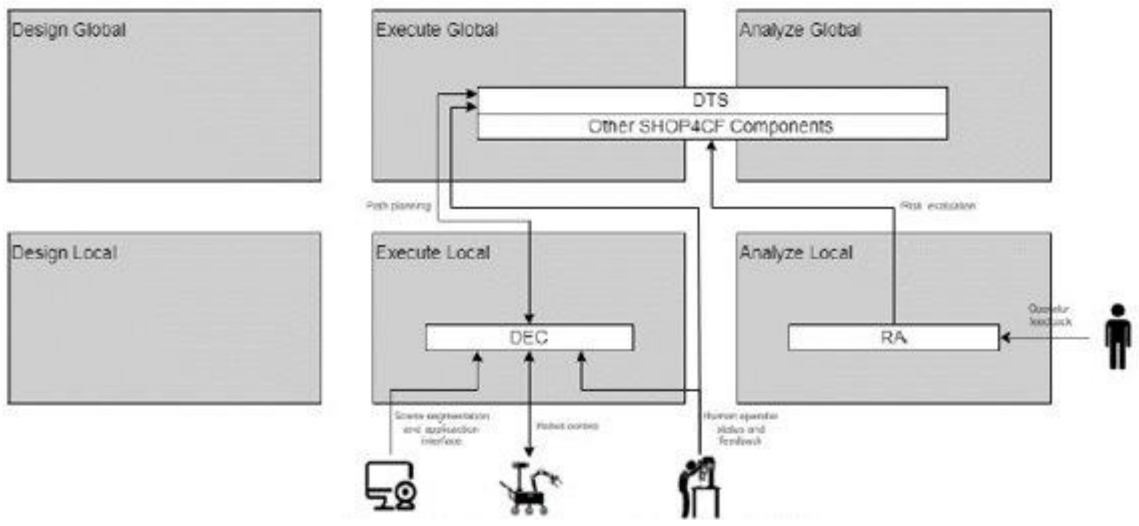

Figure 88 COOP – Logical software architecture

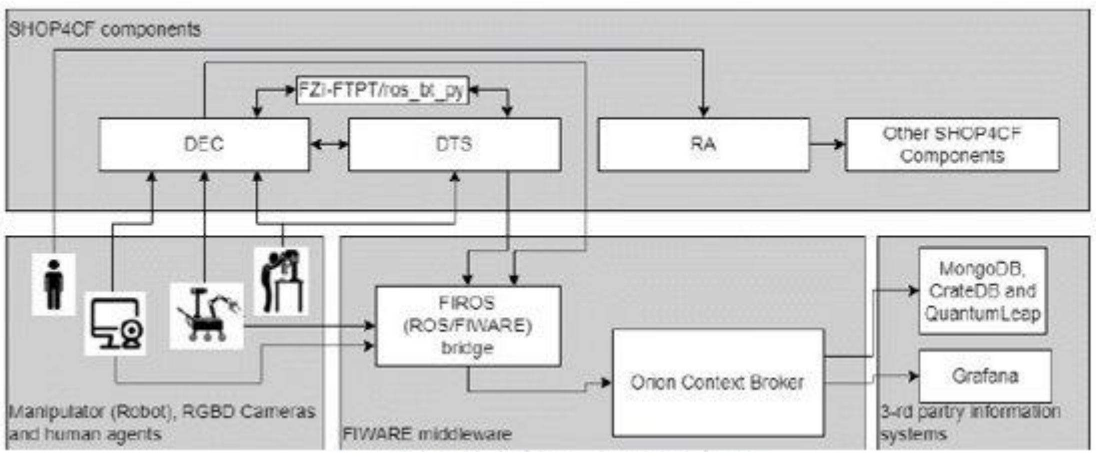
SHOP4CF components
FZI-FTPT/ros_bt_py
DEC
DTS
RA
Other SHOP4CF Components
FIROS (ROS/FIWARE) bridge
Orion Context Broker
MongoDB, CrateDB and QuantumLeap
Grafana
Manipulator (Robot), RGBD Cameras and human agents
FIWARE middleware
3-rd party information systems

Figure 89 COOP – Logical platform architecture

D.25 CUT SHOP

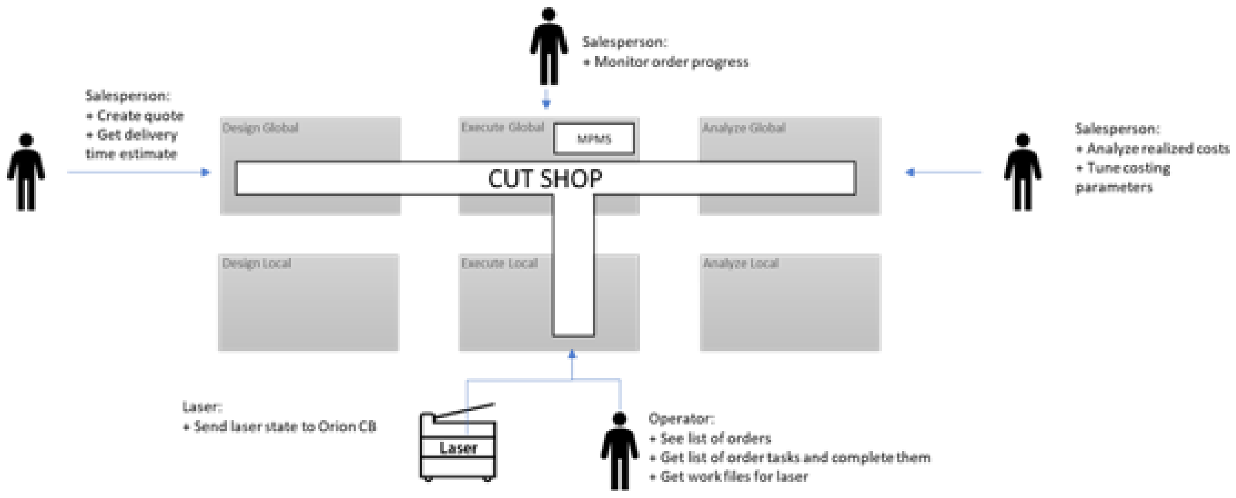

Figure 90 CUT SHOP – Logical software architecture

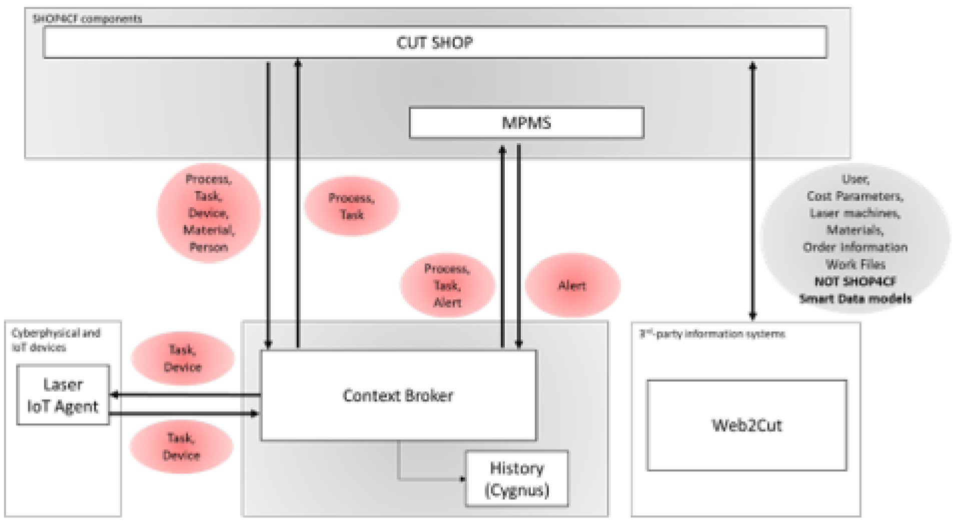
SHOP4CF components
CUT SHOP
MPMS
Process, Task, Device, Material, Person
Process, Task
Process, Task, Alert
Alert
User, Cost Parameters, Laser machines, Materials, Order Information Work Files
NOT SHOP4CF Smart Data models
Cyberphysical and IoT devices
Laser IoT Agent
Task, Device
Task, Device
Context Broker
History (Cygnus)
3rd-party information systems
Web2Cut

Figure 91 CUT SHOP – Logical platform architecture

D.26 Dynanomics

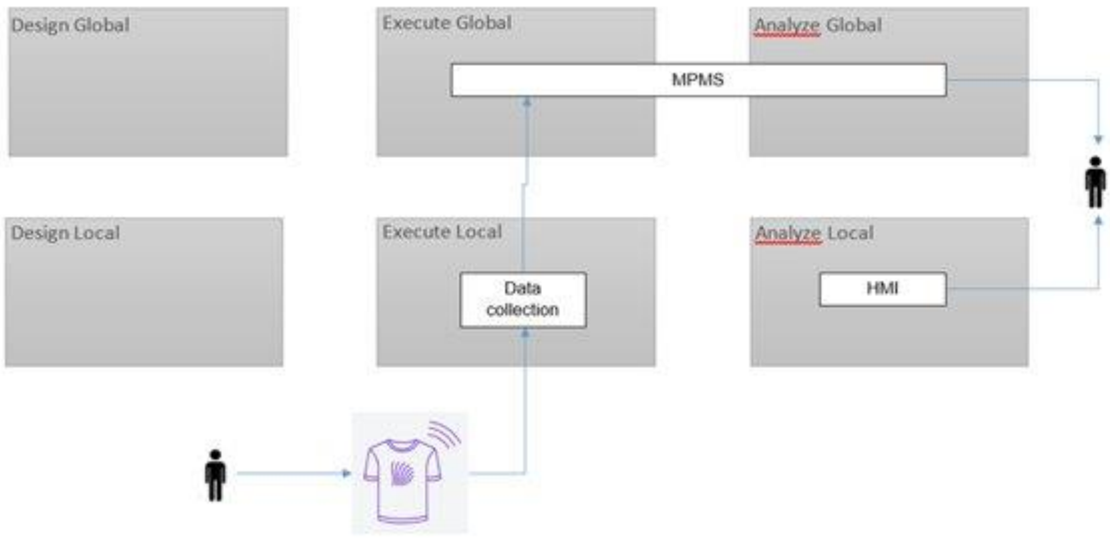

Figure 92 Dynanomics– Logical software architecture

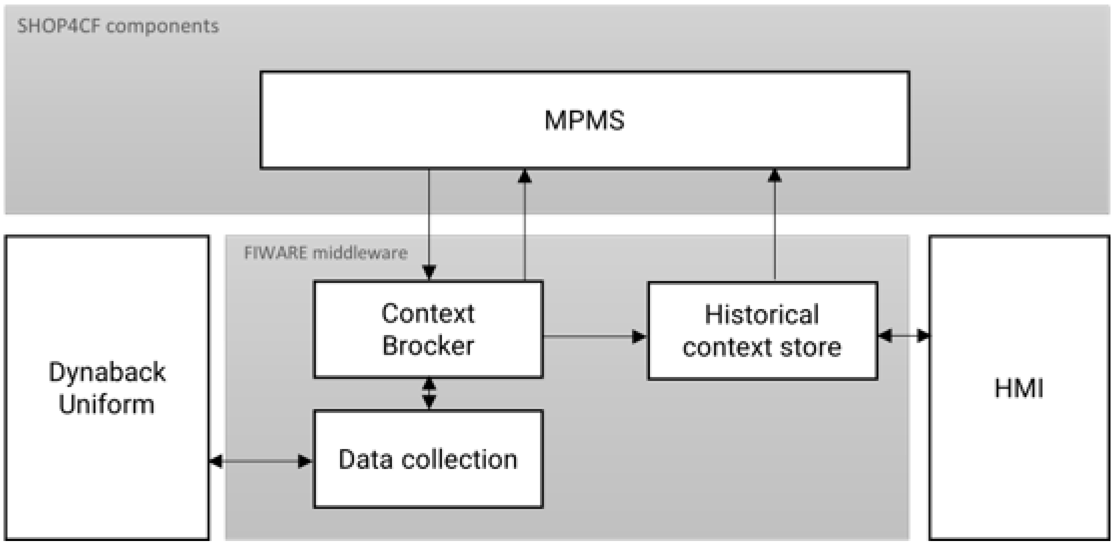

Figure 93 Dynanomics – Logical platform architecture

D.27 MonitorEX

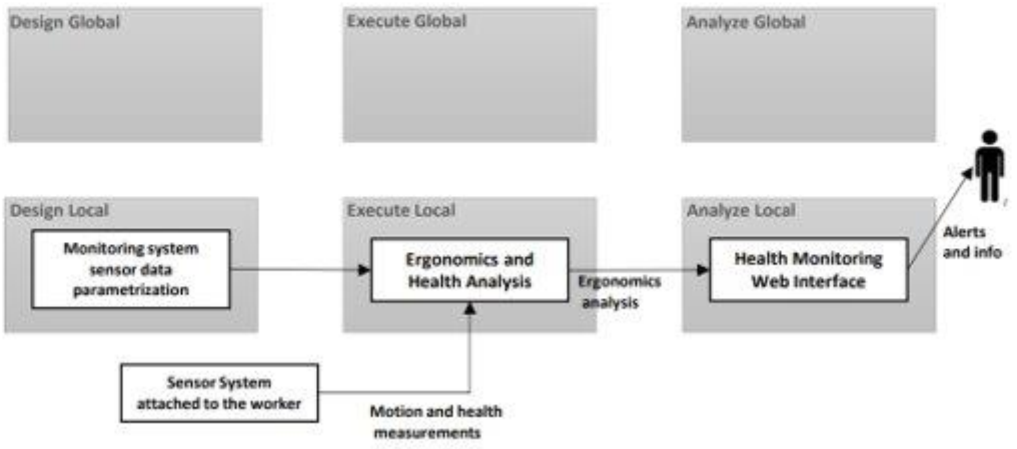
Design Global
Execute Global
Analyze Global
Design Local
Monitoring system sensor data parametrization
Execute Local
Ergonomics and Health Analysis
Ergonomics analysis
Analyze Local
Health Monitoring Web Interface
Alerts and info
Sensor System attached to the worker
Motion and health measurements

Figure 94 MonitorEX – Logical software architecture

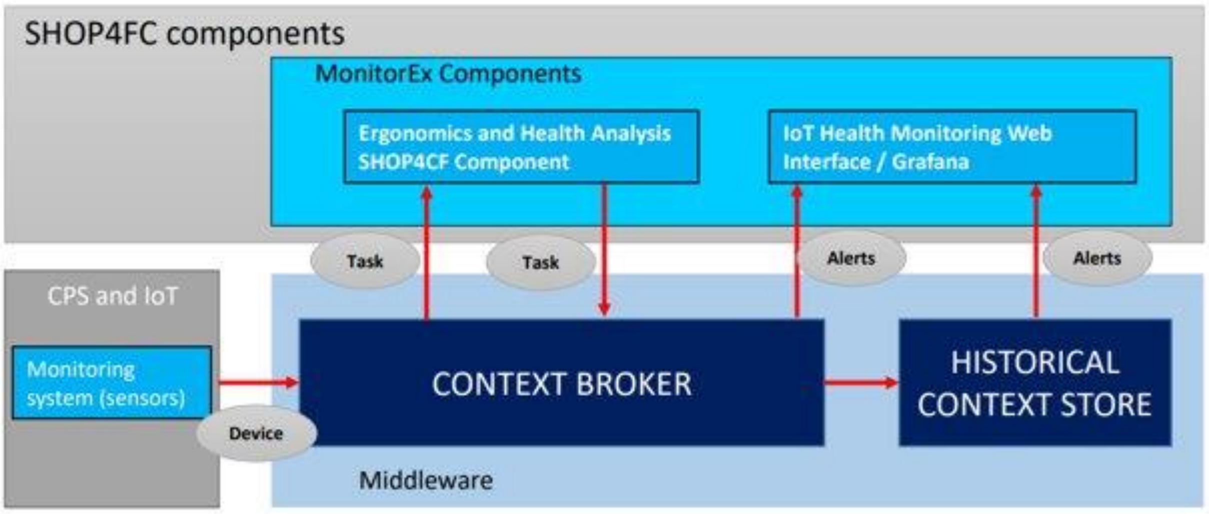

Figure 95 MonitorEX – Logical platform architecture

D.28 RTassist

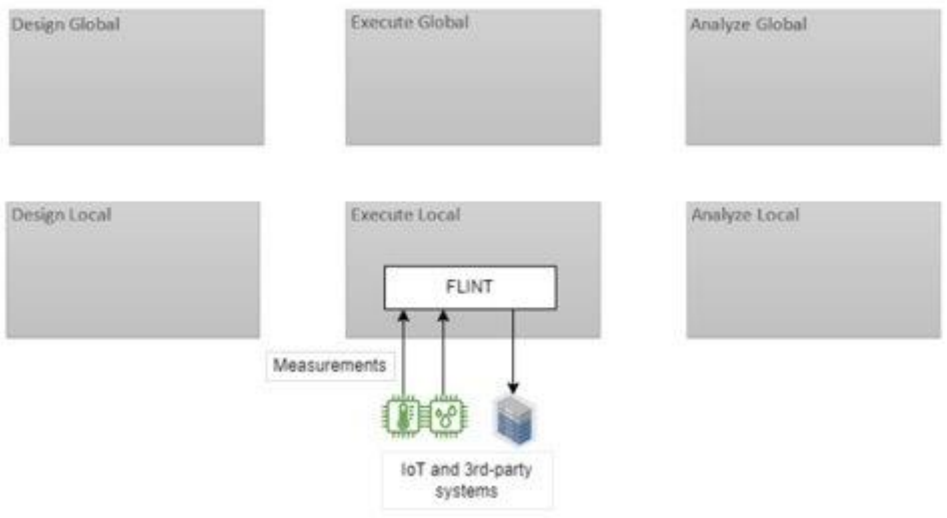

Figure 96 RTassist – Logical software architecture

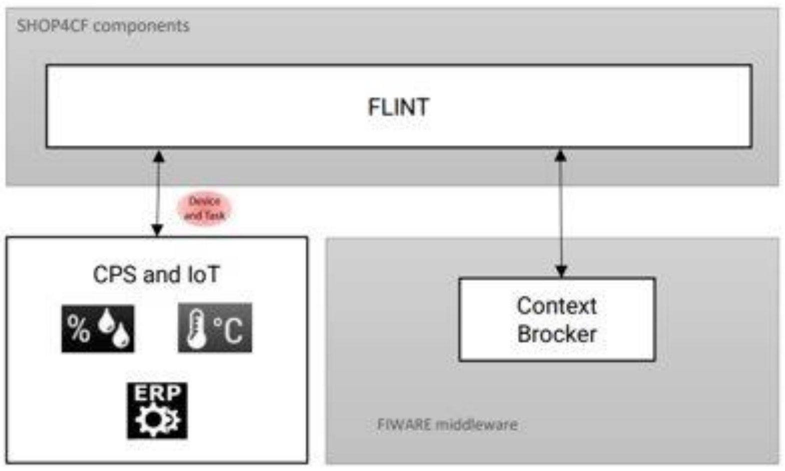

Figure 97 RTassist– Logical platform architecture

D.29 Safaho

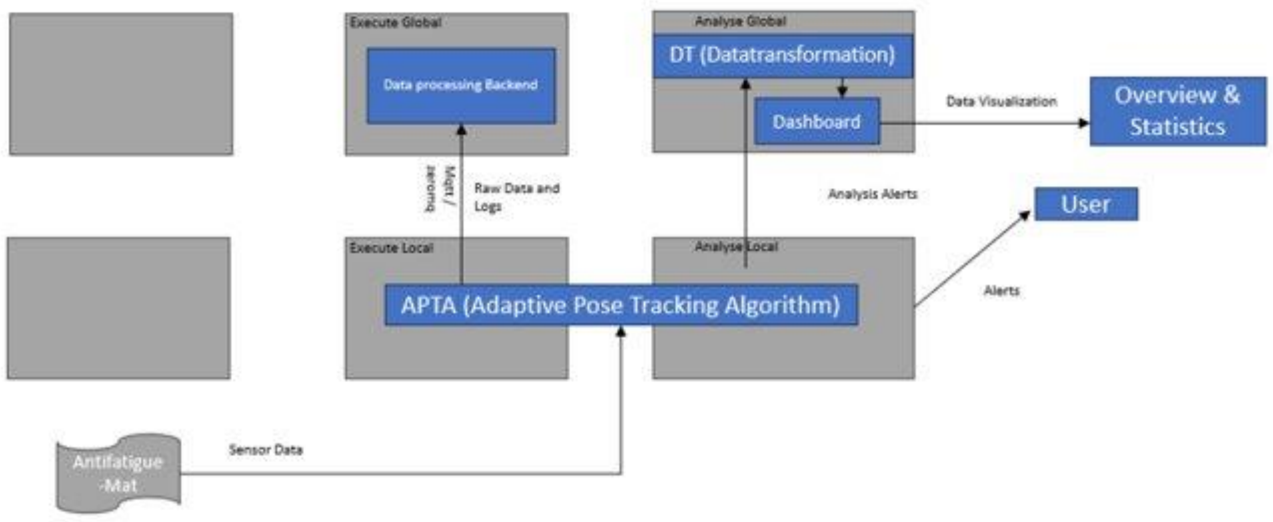

Figure 98 Safaho– Logical software architecture

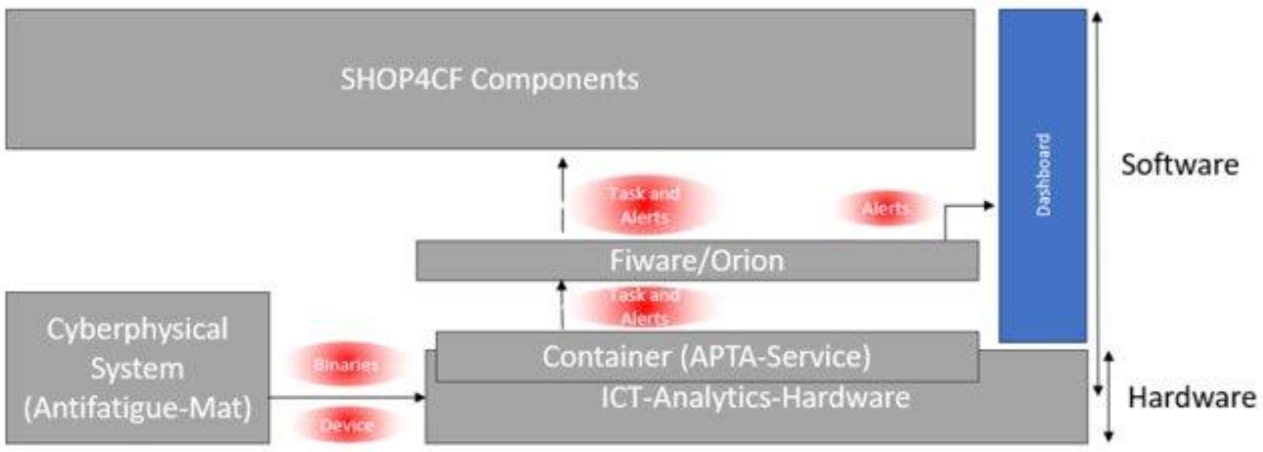

Figure 99 Safaho– Logical platform architecture

D.30 SEA WORDS

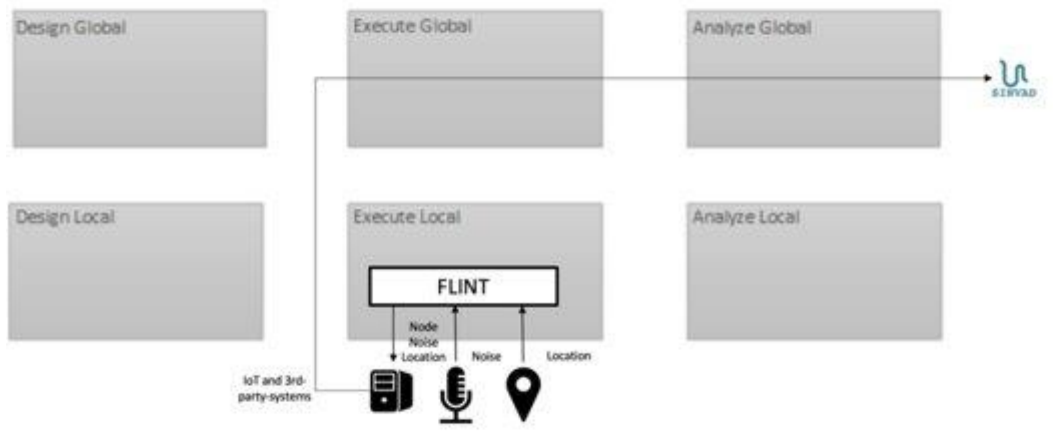

Figure 100 SEA WORDS – Logical software architecture

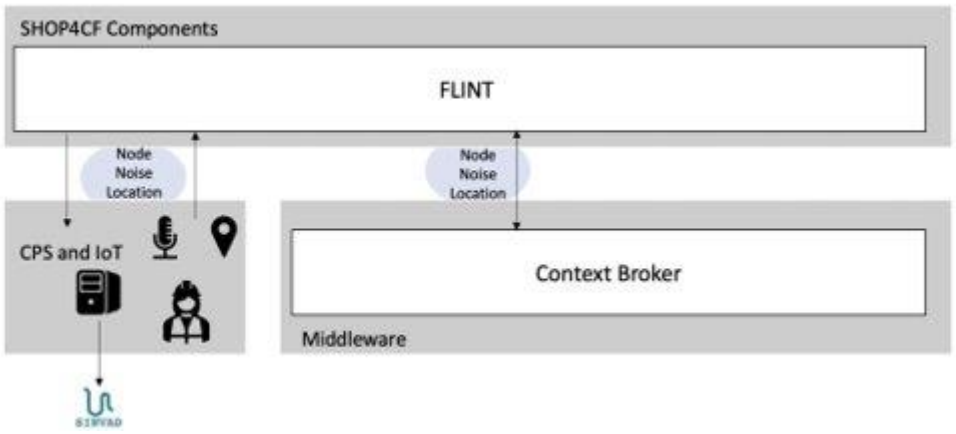

Figure 101 SEA WORDS – Logical platform architecture

D.31 SISTERS

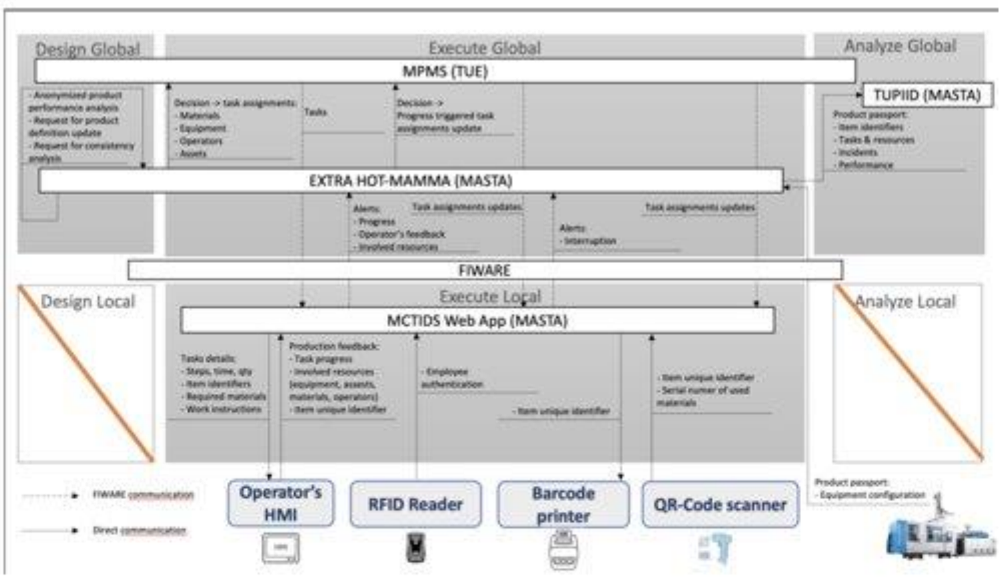

Figure 102 SISTERS – Logical software architecture

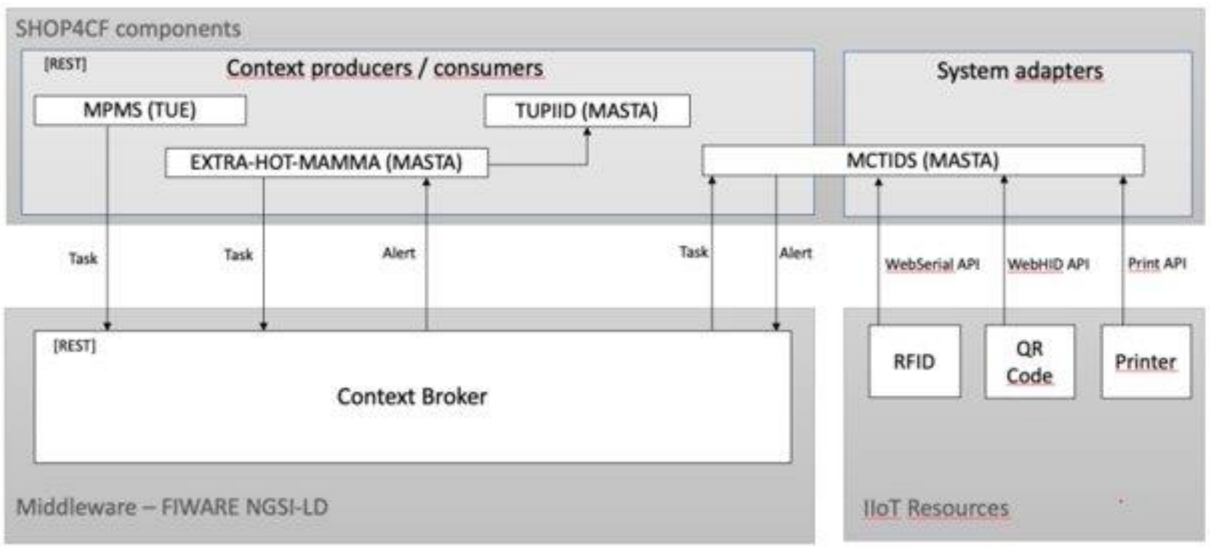

Figure 103 SISTERS – Logical platform architecture

D.32 TRanspoRT

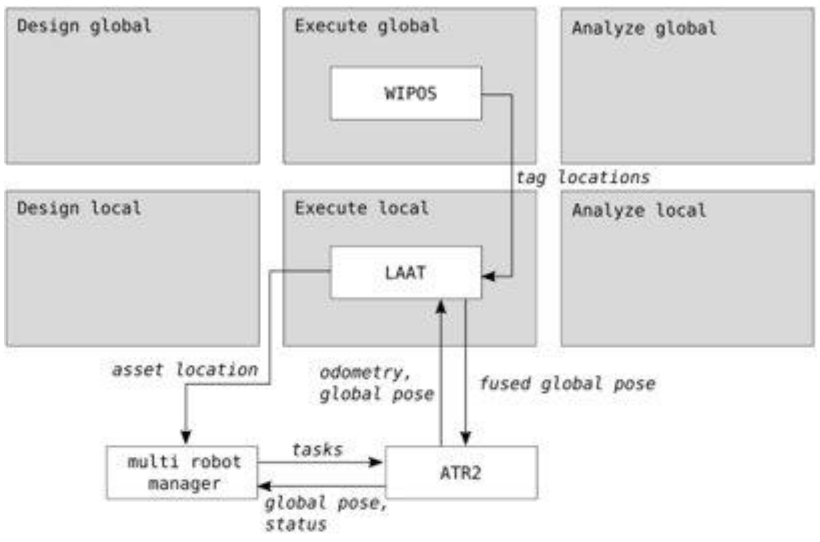

Figure 104 TRanspoRT – Logical software architecture

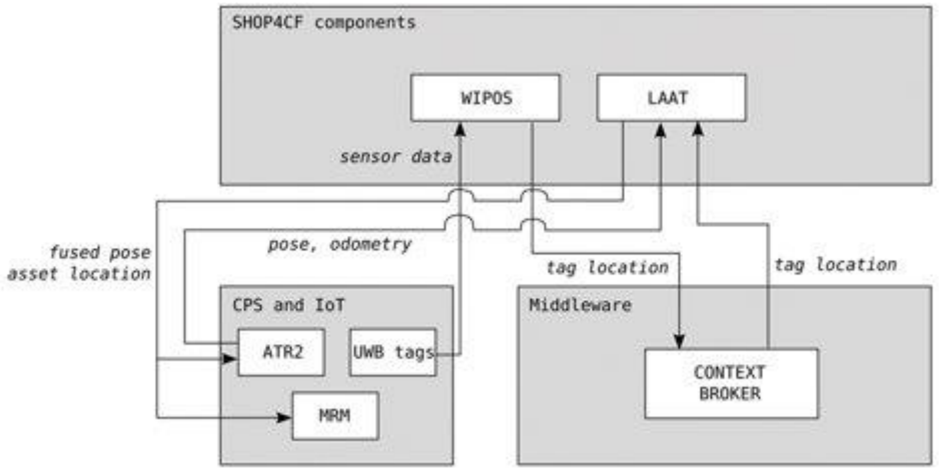

Figure 105 TRanspoRT – Logical platform architecture