D3.2b AUTOWARE Software Defined Autonomous Service Platform development

Document Owner	**FhG**		
Contributors	FhG, imec, JSI, Robo, SmartFactory, Tekniker		
Reviewers	UMH		
Dissemination level	PU	**Dissemination nature**	R
Date	27/02/2019	**Version**	1.0

Version History

Nr.	Date	Author (Organization)	Description
0.1	30/11/2018	FhG	Deliverable structure and ToC
0.2	13/12/2018	IMEC, SFKL	Contribution to Chapter 2.3 and 6, Update Chapter 2.2 and 5
0.3	14/12/2018	FhG	Update of chapter 4
0.4	20/12/2018	JSI	Update Chapter 2.1 and 3
0.5	04/02/2019	TEKNIKER	Contribution to chapter 7
0.6	13/02/2019	FhG	First consolidated version
0.7	20/02/2019	UMH	Internal review
0.8	26/02/2019	JSI, SFKL, TEKNIKER, FhG	Version with int. reviewer corrections
1.0	27/02/2019	FhG	Final Version

Project partners

Software Quality Systems	SQS
Asociación de Empresas Tecnológicas Innovalia	INNO
Technologie Initiative SmartFactoryKL e.V.	SmartFactory
Jožef Stefan Institute	JSI
TTTech Computertechnik AG	TTT
Consiglio Nazionale Delle Ricerche	CNR
imec	imec
PWR Pack International B.V	PWR
Robovision	Robovision
Universidad Miguel Hernández	UMH
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	FhG
Blue Ocean Robotics	BOR
Fundación Tekniker	Tekniker
SMC Pneumatik GmbH	SMC

Executive Summary

This deliverable is the updated version of deliverable D3.2a "AUTOWARE Software Defined Autonomous Service Platform development" which was submitted in M18. The document provides an overview of the development plan of the services that will be made available through the Software Defined Autonomous Service Platform (SDA-SP) and will be used for the implementation of the industrial and neutral use cases in WP5.

The service development partners of WP3 in AUTOWARE are using different infrastructures and platforms (ROS, CloudiFacturing and GPflowOpt) to develop and provide their services to the AUTOWARE project. From the conceptual point of view the SDA-SP can be seen as a generalized reference architecture to realize cognitive applications and spans above individual platforms to integrate all AUTOWARE services in one architectural representation.

Changes to the development plan of the services related to the implementation of the main three WP3 assets (a reconfigurable robotic work cell, a mixed or dual reality supported automation to implement an effective and flexible collaboration between humans and robots, and a multi-stage production line) are presented in this document.

Additionally two new use cases are added:

- Neutral experimentation infrastructure for intelligent automation applications for robotic industrial scenarios and
- Industrial Cognitive Automation Validation

This document describes how each one of these two developments instantiates the reference architecture of the SDA-SP in order to provide use-case specific reference implementations in the project.

The next step will then be the revision of the reference implementation of the SDA-SP, which were documented in the deliverable D3.3a, taking into account the changes made on the development plan described in this document.

Keywords

Service platform, information systems in industrial production, cyber-physical production systems (CPPS), (re)configurability, reference architecture

Table of Contents

List of Figures

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CAD	Computer Aided Design
CATIA	Computer-aided three-dimensional interactive application: a CAD format developed by the French company Dassault Systèmes
CPS	Cyber-physical system
CPPS	Cyber-Physical Production System
DLL	Dynamic Link Library
ERP	Enterprise Resource Planning
FIWARE	Open cloud-based platform for cost-effective creation and delivery of innovative applications and services
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HPC	High-Performance Computing
IIB	IBM Integration BUS
IoT	Internet of Things
JSON	JavaScript Object Notation
JT	Jupiter Tesselation: an ISO-standardized 3D data format used for product visualization, collaboration and CAD data exchange
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport

OPC	Open Platform Communications
OPC UA	OPC Unified Architecture
OS	Operating System
OWL	Web Ontology Language
PMI	Product Manufacturing Information
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
PLMXML	Siemens PLM Software format for facilitating product lifecycle interoperability using XML
RDF	Resource Description Framework
resCNN	Residual Convolutional Neural Network
REST	Representational State Transfer
RFID	Radio Frequency Identification
ROS	Robot Operating System
SDA-SP	Software Defined Autonomous Service Platform
SDK	Software Development Kit
TPU	Tensor Processing Unit
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

1 Introduction

The specification of the reference architecture for the Software Defined Autonomous Service Platform (SDA-SP) was introduced in Deliverable D3.1 and laid the foundation for the upcoming activities in WP3. In deliverable D3.2a this reference architecture was instantiated by the main three WP3 assets:

(i) A reconfigurable robotic work cell,
(ii) A dual reality supported automation framework as an enabler for human-robot collaboration and
(iii) A multi-stage production line system.

Their instantiations and their services were described in detail in addition to their development plans. Based on D3.1 and D3.2a, this document D3.2b provides an updated description of the development plan of the SDA-SP with its services and platforms.

Evaluation of the individual pilots in WP5 and associated KPIs, especially the degree of fulfilment of the business and technical KPIs, gave an indication concerning potential improvements and further developments of the reference implementations of the SDA-SP. Consequently, the updated version of the deliverable D3.2 provides an overall update of the existing reference implementations and additionally includes two more instantiations to complete the overall AUTOWARE approach. These reference implementations are linked with the following AUTOWARE use cases:

(iv) Neutral experimentation infrastructure for intelligent automation applications for robotic industrial scenarios (owned by the AUTOWARE partner TEKNIKER);
(v) Industrial Cognitive Automation Validation (owned by the AUTOWARE partners imec and RoboVision)

Since the AUTOWARE architecture is open and thus extensible, the inclusion of two more use cases gives the opportunity to reveal a statement about the representativeness and applicability of the AUTOWARE architecture.

1.1 Scope of the deliverable

This deliverable is linked with the task T3.4 Reference implementation for the Software Defined Autonomous Service Platform (M13-M30, FhG lead). The task T3.4 is described in the Description of Work (DoW) as

> *At the end of the first and final iteration of the developments, T3.4 will proceed with the integration of the different components into a reference*

implementation of the Software Defined Autonomous Service Platform. This task will also implement the enablers and interfaces needed at this layer to connect the Software Defined Autonomous Service Platform with the Physical, Connectivity & Data Management framework in WP2 and the Open CPPS Ecosystem implemented in WP4.

1.2 Relation to existing platforms

The service development partners in WP3 are using different infrastructures and platforms to develop and provide their services to the AUTOWARE project. Functionalities are implemented as services which can then be combined and orchestrated to provide specific factory solutions. The services to be implemented will be mapped on the AUTOWARE architectural layers to create instantiations of the reference architecture. Each one of these individual reference implementations of the SDA-SP together with corresponding services in the different architectural layers will be used to commission the AUTOWARE pilots in WP5.

Due to advances in recent EU research projects, the AUTOWARE project will use the more recent update of the CloudFlow technology environment. It is developed in the CloudiFacturing project, which is more focused on the manufacturing industry.

Figure 1 depicts the individual instantiations and the different reference service implementation platforms. The existing platforms being used in WP3 are:

- ROS (Robot Operating System) is used by JSI and TEKNIKER
- CloudiFacturing is used by SFKL and FhG
  SFKL will provide core infrastructure components such as Workflow Manager or High Performance Computing (HPC) hardware for the AUTOWARE project.
- GPflowOpt used by imec

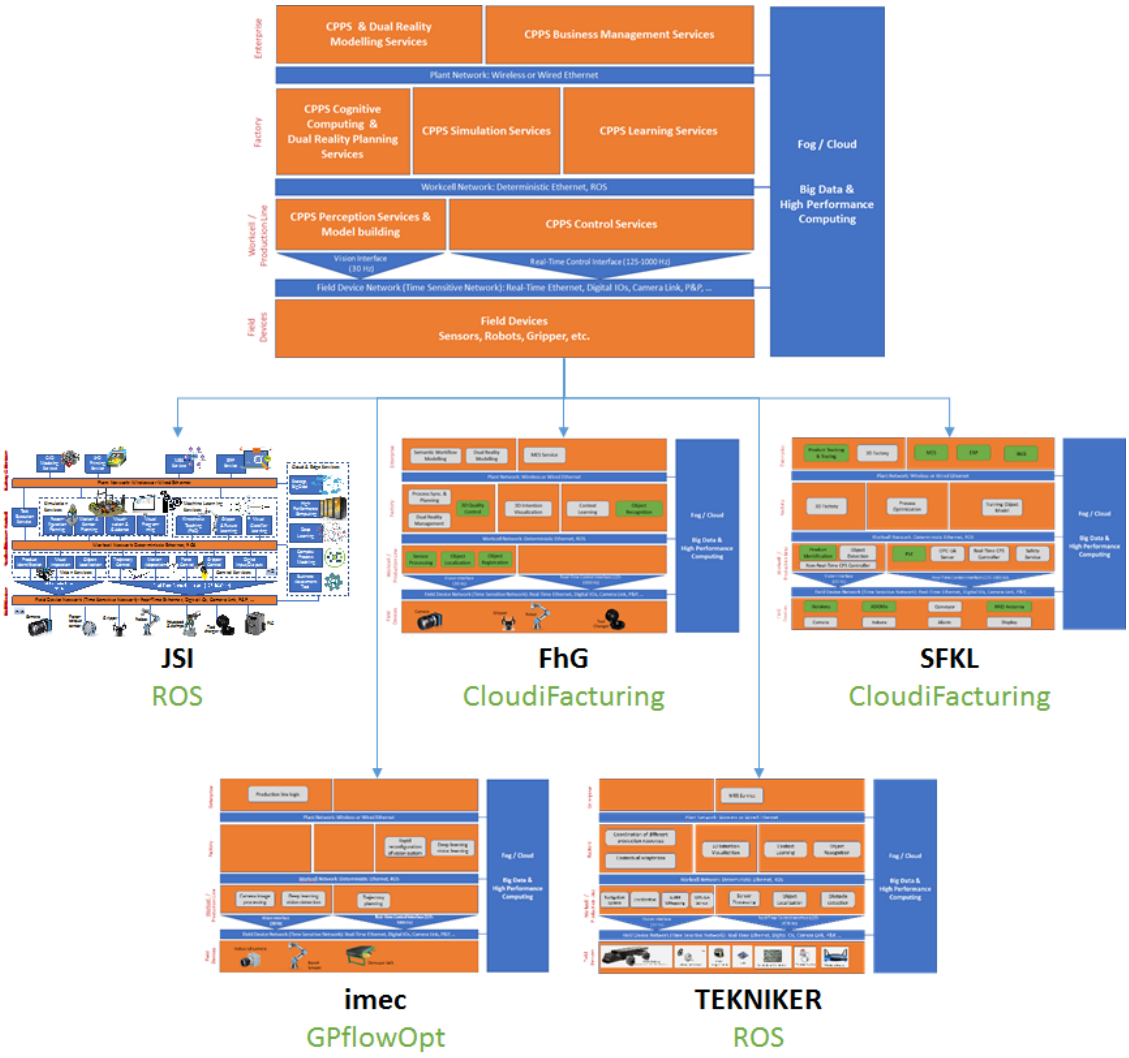

Figure 1: Overview over all instantiations of the reference architecture and service implementations based on ROS, CloudiFacturing and GPflowOpt.

1.3 Document Structure

The document is structured as follows: Chapter 2 introduces and states the newest updates on the existing platforms mentioned above (ROS, CloudiFacturing and GPflowOpt) that are being used in WP3 to implement the AUTOWARE cognitive services. Chapters 3, 4 and 5 continue with dedicated implementation plans for the three main WP3 technology developments: *a reconfigurable robotic work cell, dual reality supported automation,* and *a multi-stage production line by using digital product memory technologies*.

In addition two new use cases: *Neutral experimentation infrastructure for intelligent automation applications for robotic industrial scenarios* and *industrial cognitive automation validation* are described in chapter 6 and 7 with their instantiation of the SDA-SP reference architecture and the development plan for their services related to

cognitive automation validation and collaborative robotics. Chapter 8 gives an overall summary of this document and provides an outlook for future activities.

1.4 Target audience

This document is mainly intended for developers of the software defined autonomous service platform. For example, automation developers and integrators of the use cases should follow the directions given here. It also includes information for the providers of technical enablers (e.g. edge computing, deterministic communication), to support the design of enablers with appropriate characteristics.

2 Existing Software Defined Service Platforms

This section gives an updated overview on the different software infrastructures that are currently being used in the AUTOWARE project to implement the services for the SDA-SP.

2.1 ROS

JSI instantiated the general AUTOWARE software architecture for the case of cyber physical production systems that deal with functional layers ranging from supervision and higher-level control down to hard real-time low control (and sensing) of devices that are involved in actual physical shop-level production. We are dealing with production processes involving manipulation of work pieces and/or tools. Advanced production of this type often – not always – involves robot manipulators, beside others specific production technology devices. We selected **ROS**, a set of tools and libraries that runs on top of an operating system. It is a kind of SDK (Software Development Kit) for robotic applications, thus a middleware software component. In AUTOWARE, we use it as a main robotic software building block/component to design a service based architecture, according to principles developed and described in the deliverable D3.1. It covers functionalities as hardware abstraction, device drivers, libraries, visualizers, message-passing, and package management. In our context, the most important single ROS' characteristics is unified inter-process communication.

In the previous release of this deliverable (D3.2a) we described the requirements of such software systems, explained the selection of ROS and gave an overview of its advantages and shortcomings. In this final delivery that is dealing with the plan to implement the various features of the robotic oriented service platform, we will describe the "institutional" ROS related community activities to advance "standard" or "first" ROS capabilities and address its shortcomings.

2.1.1 ROS-industrial

This is an initiative that bases on "standard" ROS distributions. It address predominantly the shortcomings that made ROS less attractive to industrial users in actual production environments, in contrast with its significant acceptance and use in the academia and in research [Gerkey, 2017, Tellez, 2018]. ROS-Industrial initiative is also an open-source project; its main result is a verified collection of software, the so called "ROS-Industrial repository", that provides modules and libraries that are robust, reliable and simple enough, so that they can be used for actual production and manufacturing applications.

2.1.2 ROS 2

This is the "next" ROS, conceived and developed by some individuals that were involved in the development of the "first" ROS and by a community that intensively uses it. With ROS 2, they address all shortcoming of ROS [Gerkey, 2017, Tellez, 2018]; the last ROS 2 version was released in December of 2018, the next one will be in June of 2019. There is a public roadmap of planned functionalities to be implemented, including porting of ROS packages to ROS 2.

In chapter 3, we will describe the implementation plan for the Autonomous Robotic Work Cell Services. It initially describes the selection among ROS, ROS-Industrial and ROS 2. The plan also position ROS inside the overall AUTOWARE service platform architecture.

2.2 CloudiFacturing

2.2.1 General Description

Due to advances in recent EU research projects, the AUTOWARE project will use the more recent update of the CloudFlow technology environment as it is developed in the CloudiFacturing project. Aimed at the manufacturing industry, it includes the CloudFlow workflows (as described in more detail in Deliverable D3.2a) as a set of calculations, simulations, and analytics jobs for complex engineering and manufacturing tasks which are executed in a cloud environment as shown in Figure 2.

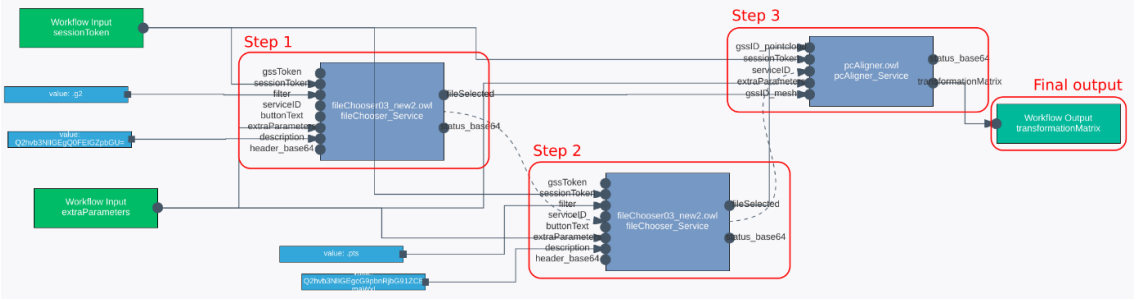

Figure 2: CloudFlow workflow editor

The manager allows arbitrary software to be encapsulated into web services or high-performance computing (HPC) jobs, which can be arranged graphically into workflows and executed and monitored using a graphical web interface (as described below in Section 2.2.1). The advantage of using the CloudiFacturing solution in contrast to the pure CloudFlow solution lies in the improved technical interoperability of the Cloudflow components and the integration into a unified user interface.

This architecture includes the User Management component, the repository for executable artefacts and the workflow and application mediator as well as the data transfer and browsing system.

These components can be described as the following:

1. The User Management component provides a single entry point for the complete solution. Through this logging in point, it is possible to access all integrated technology stacks.
2. The Repository for Executable Artefacts stores information on all applications, workflows etc. including metadata about the technology stacks an artifact is connected to. This allows users to view artefacts relying on different technology stacks in exactly the same way.
3. The workflow and Application mediator acts as a unified execution interface for software artefacts registered in the repository. The mediator translates repository metadata of registered artefacts into execution calls to the different technology stacks.
4. The Data Transfer and Browsing system allows data transfer between storages connected to different technology stacks. It makes it possible to create "meta artefacts" which consist of several artifacts from different technology stacks.

2.2.1 Interface

Based on the architecture described above, an input mask was created that makes it possible to deploy services in a workflow. This deployment has been simplified and is based on Docker [Merkel, 2014] (software that performs system-level virtualization) to containerize components. A screenshot to demonstrate how these services can be implemented is shown in Figure 3.

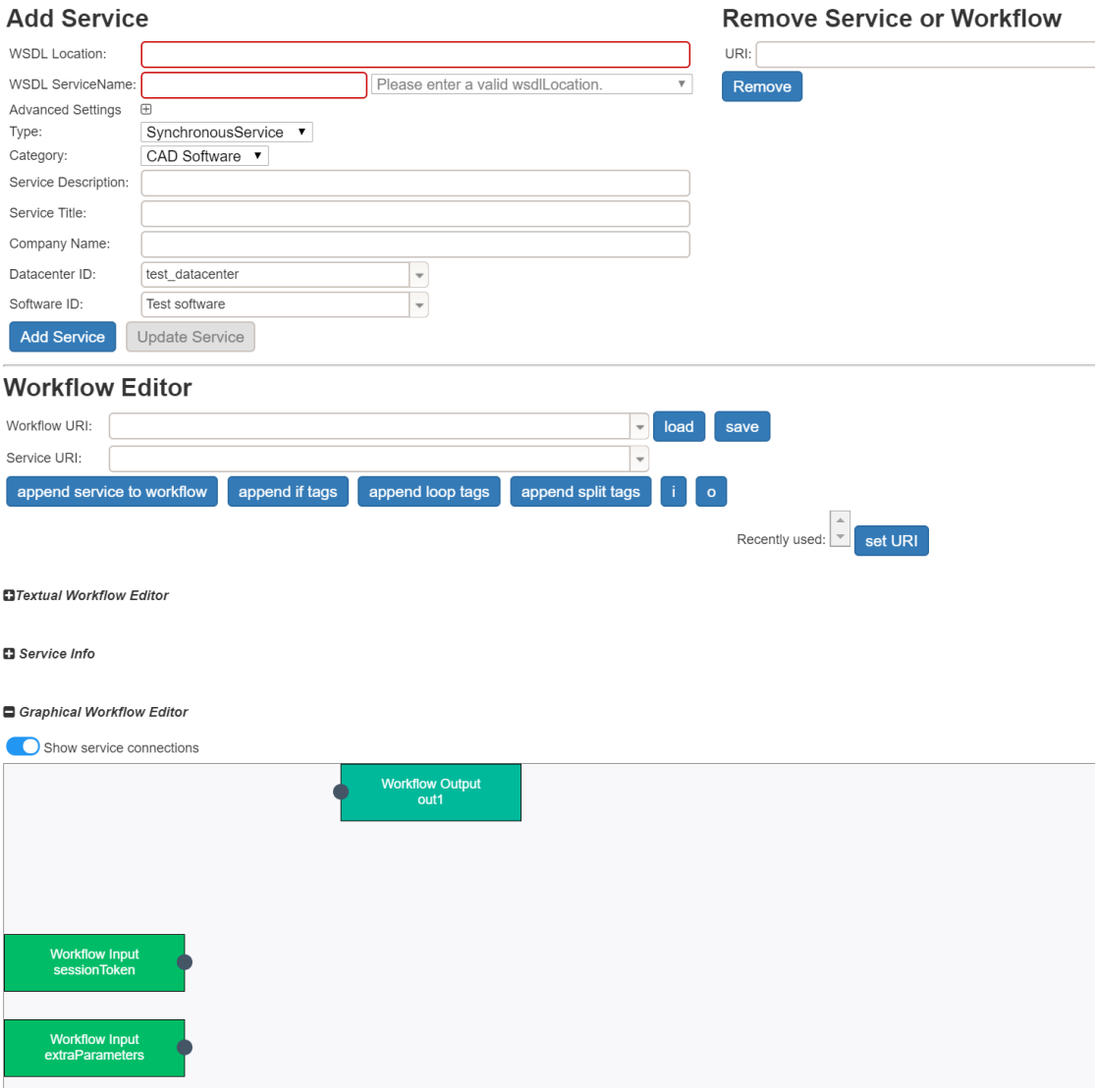

Figure 3: Improved Workflow User Interface

2.3 GPflowOpt

GPflowOpt is a powerful optimization framework, which distinguishes itself in three aspects from other more academic alternatives. First the fact that it builds on modern frameworks allowing for fast computation which is required in a manufacturing setting, reducing the changeover times by rapid reconfiguration of the artificial intelligence (AI) systems. Secondly, the quality of the implementation that reaches 99% code coverage and thirdly, the ease of use, allowing adoption for multiple scenarios without requiring expert knowledge. GPflowOpt offers a representational state transfer application programming interface (RESTFull API) in which JSON (JavaScript Object Notation) objects are used as the standard communication protocol.

With regard to D3.2a, the main work on GPflowOpt has been done integrating several development branches into the master branch to allow for further functionality in the core software. There have been no changes with regard to the services and API as discussed in deliverable D3.2a. In addition, we received several bug reports both internal and external which have been the main focus in the last months prior to this deliverable.

3 Implementation Plan of Autonomous Robotic Work Cell Services

One of the JSI's project tasks is to instantiate and implement the AUTOWARE software architecture, developed in other work packages, for the case of CPPS involving robotics. In the first WP3 deliverable, we defined the principles and structure of the software defined autonomous service platform; accordingly, JSI will implement a robotic work cell control system as part of a services-based framework. Also, JSI hosts, maintains and runs one of AUTOWARE's neutral facilities, a reconfigurable robot cell, where all these principles will be used and demonstrated. The activities related mainly to the mechanical/hardware design and disposition have been mostly (but not exclusively) done in a concurrent ReconCell project; the design and implementation of appropriate service as a software architecture and the design and inclusion of specific enablers (part of the WP4) is being mainly done as part of AUTOWARE.

3.1 Robotic services integration – deployment of ROS in a service based structure

Due to the robotic nature of the CPPS we are targeting, we choose ROS as basic robotic middleware. We described its advantages in the previous deliverable report release. In chapter 2, we described two current additional initiatives, "ROS-industrial" and "ROS 2" that address the "first" ROS shortcomings for use in an industrial production environment. All three versions are currently maintained and being developed or updated. We investigated thoroughly all three versions for use in the AUTOWARE WP3 tasks, taking into account the project's resources and most importantly the project's time schedule. In the following we recap our findings and technical solutions selected for our AUTOWARE implementation.

- ROS and ROS 2 are both excellent as basic framework for task decomposition and distribution by supporting transparently inter-process and inter-computer messaging and other related functionalities.
- ROS does not address the real-time and related lower level robot/device control requirements adequately out of the box.
- "ROS-industrial" does in principle address real-time and control requirements but presently not enough for practical use; from the list of supported robots and functionalities in December 2018 (regarding ROS module functionalities for real-time robot control/sensing), we see that a limited number of commercial robots is supported; the type of robot control commands, e.g. trajectory control and force/torque control is even more limited.

- ROS 2 supports real-time requirements adequately.
- ROS 2 development is presently (December 2018) concentrated and progressing very well on core functionalities; it is lagging in the transfer of useful functionality modules (to name some, "MoveIt!" and "Gazebo") from ROS to ROS 2.

Based on our assessment and findings, our plan regarding the implementation of robotics middleware in the software defined service platform is the following:

- We will embrace "first" ROS as middleware; in this way, we can rely on existing functionality packages mentioned before;
- We will develop specific packages for advanced functionalities, e.g. those related to automated re-configurability and cognitive capabilities; in doing this, we will stick to basic/standard ROS functionalities, so that the foreseen future porting from ROS to ROS 2 will be easier.
- As ROS doesn't support real-time control functionality at the level required for robot control, we will develop a dedicated solution to resolve this issue.

3.2 Implementation plan to provide three scales integration

It is one of WP3 goal to provide integration of CPSS at three scales (edge, cloud, and HPC). In this sense we planned our implementation according to following principles and design choices.

- The main work cell control system will use ROS, thus it will run on a Linux-based operating system. The implementation will run on one or more PC-class computers. The number of computers and their power will be proportional to the requirements of the algorithms to be used for a specific production use-case application. Here we will leverage ROS capability to seamlessly distribute ROS nodes for an application on one or more computers as needed.
- Based on the initial choices, we could also implement the main work cell control system on a number of the "fogNode"s, the fog/edge units provided as an enabler by an AUTOWARE partner. For the time being this is not possible due to the relatively lower computing power on available fogNode models (based on Intel Atom processor) than it is necessary by algorithms for JSI's AUTOWARE use cases. However, as new versions with more powerful processors will be introduced, it will be possible to seamlessly move all or a part of the work cell control system on one or more of those "fogNode"s.
- We established that ROS does not support hard real time that is necessary for the lower level control of manipulation-capable production hardware as robots, positioning devices, etc. (although it is supported in ROS 2 and we will eventually switch to). To achieve the control/sensing real-time functionality, we designed and

devised a so-called "server for real-time robot sensing and execution" (further: SLRT server), one per robot. The additional functionality we implemented in the SLRT server is production hardware abstraction. It functions as an interface between standard ROS node's sensing and control commands and proprietary low level control of these devices (e.g. proprietary motor drives for robot joint motors).

The SLRT software code is generated with our application preparation system for a specific production hardware. For example, in our use case applications, we generated and implemented a SLRT server for Universal Robots UR10, used in our work cell. We presently run them on a PC compatible computer. The SLRT could also run on the fogNode, provided it offers the possibility to boot the machine to the real-time kernel, embedded in the SLRT distribution.

- To provide integration of the work cell local components with other distributed and/or remote components at the edge and in the Cloud scale, we will implement a ROS IoT (Internet of Things) communication layer. In our AUTOWARE implementation it will provide MQTT (Message Queuing Telemetry Transport) and OPC UA (OPC Unified Architecture) based connectivity of ROS nodes to external systems in the edge/fog or in the Cloud. The connectivity will be transparent, that is, the ROS nodes will seamlessly use the same mechanism for ROS-IoT communication as for "internal" ROS-ROS node communication.

  This mechanism could be used for communication with an external service or with an externally implemented application. In a typical CPPS application, this mechanism could be used for duties that in a traditional control scheme qualify as MES (Manufacturing Execution System) or ERP (Enterprise Resource Planning) functionalities; however, in contrast with traditional implementations, in AUTOWARE architecture they do not have to follow through the traditional hierarchical communication chain.
- To provide integration at the HPC scale, we will conceive and implement a connection between local work cell components and a HPC level computing facility that is essentially an edge node providing HPC capable hardware and software.

  Due to the large amount of data that has to be exchanged with the HPC node in our use cases, we will use a suitable transfer mechanism (e.g. a IoT protocol as MQTT would not be suitable).

  As part of WP4, we will provide a demonstration of the implementation on the example of pre-training of deep learning networks. This is essential for cognitive functionalities, e.g. for those involving vision processing.

As described in the first release of the D3.2 delivery, the implementation is planned in two steps:

- The first step is to decompose all functionalities into tasks with a modular structure while observing related principles from traditional software engineering like modularity, abstraction, anticipation of change, incremental development and consistency.
- In the second step, we will implement modules with emphasis on service based operation. The main common building block will be the ROS. We will provide for connection with external platforms e.g. edge/fog, Cloud, and HPC.

The two steps can be illustrated by the schemas in Figure 4.

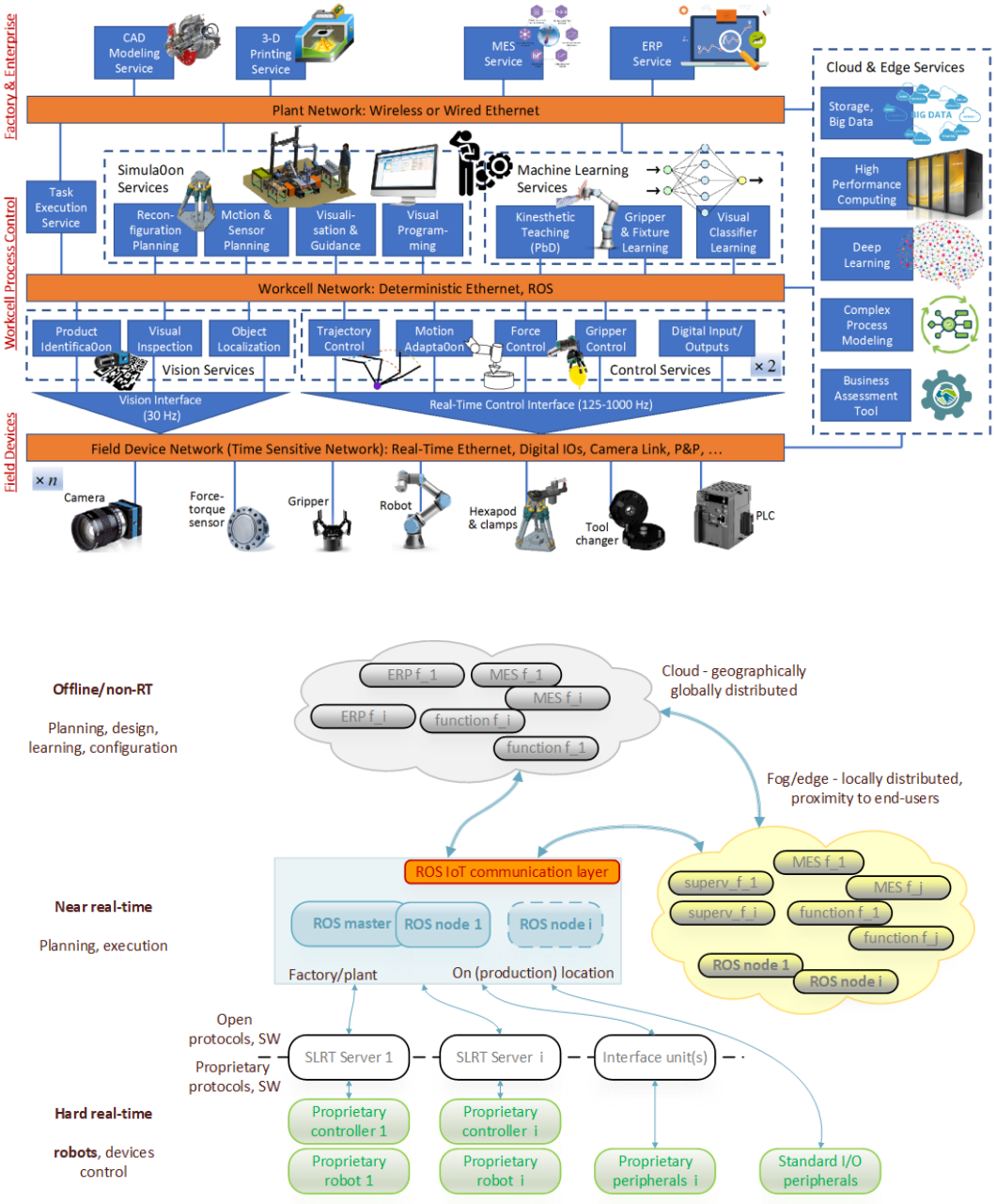

Figure 4: Architectural structure of the planned autonomous robotic work cell implementation. Above: functional view, below: SW and communication implementation view

4 Implementation Plan of Dual Reality Management Services

The implementation plan of the Dual Reality Management Services foresees to implement six high-level services, which were first described in Deliverable D3.2a Software Defined Autonomous Service Platform development. These high-level services consist of a combination of single services, which we listed and described in Deliverable D3.1 *Reference Architecture for Software Defined Autonomous Service Platform*. Descriptions and changes or updates to our services are listed in the following sections.

Four services are instantiated as local services since they are constrained by real-time requirements or do not need external processing. Those are:

- Cognitive Assembly Monitoring and Control
- Dual Reality Control
- 3D Intention Visualization
- Sensor Processing

Two services can be outsourced to run on a cloud server. In this project, we make use of the CloudiFacturing platform to propagate our cloud services:

- Object Recognition
- 3D Quality Control

4.1 Local Services

4.1.1 Cognitive Assembly Monitoring and Control

The Cognitive Assembly Monitoring and Control service foresees to make existing product engineering and production planning data available for a secondary use, namely for cooperative assembly tasks between humans and robots. It can be used to perform automatic extraction of product engineering data (i.e. 3D CAD – computer aided design – files) and production planning data (i.e. structured textual data describing the manufacturing execution and the assembly process) for individualized products / prototypes from a PLM (Product Lifecycle Management) system. In addition, the service performs a consistent semantic mapping of product engineering and production planning data including 3D virtual models of tools and assembly parts on cognitive models. As a result, we get a semantic description of the assembly process incorporating the 3D representations of the assembly steps.
These semantic models can then be queried to extract information about the assembly workflow and related geometry data.

4.1.1.1 Implementation Plan

The implementation plan for the Assembly Monitoring and Control Service foresees to incorporate World Wide Web Consortium (W3C) standards to construct our domain models. We use the Web Ontology Language (OWL), which uses the Resource Description Framework (RDF) as basis, as knowledge representation language for creation and authoring of our semantic model. The semantic model is filled with information about the production planning data extracted from eXtensible Markup Language (.XML) or Siemens PLM Software XML format (.PLMXML) files and additional referenced geometry data (e.g. Jupiter Tesselation (.JT) or Computer-aided three-dimensional interactive application (.CATIA) files). Alternatively the semantic model can be easily viewed and edited manually with a variety of ontology editors, e.g. PROTÉGÉ [Musen, 2015] (Figure 5). To make use of the knowledge stored in the semantic model the service provides a query interface and rule engines. In D3.2a we listed RDF4J as framework to process the queries. However, we revised the query interface to work more seamlessly and more effective with our dual reality application based on C++. We decided upon owlready2 [Lamy, 2017] which is an ontology programming interface. This open-Source software allows ontology-oriented programming in Python, and can be easily embedded in our application.

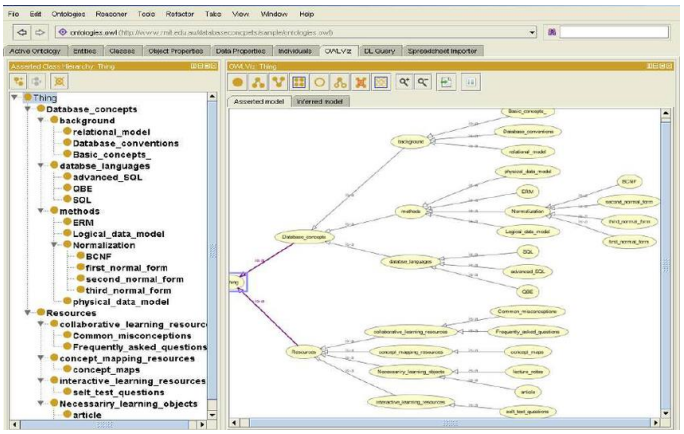

Figure 5: Visualization in PROTÉGÉ of an example of a semantic model

4.1.2 Dual Reality Control

The Dual Reality Control Service is able to model dynamic virtual environments and enrich these with virtualized physical production environments (Augmented Virtuality). Virtual environments and objects (e.g. assembly parts, production tools) are retrieved from the PLM System. The service receives information, which is captured by 2D/3D sensors, about the real world and transforms the input to virtualized real object representations. The service then connects real processes with virtual processes and thus allows a continuous synchronization between the real and virtual environment. It

should be noted that virtual objects could also influence the behavior of real objects. By controlling the behavior of the real physical production environment, the service is able to react directly on virtual objects. As a result, the Dual Reality Control service creates an Augmented Virtual Environment Model containing objects from real und virtual environments.

4.1.2.1 Implementation Plan

To realize the service we developed a data structure to represent the Scene model, which is capable of storing and combining different data sources (CAD models and real world input, e.g. scan point clouds, meshes etc.) in one representation. A CAD loader and converter is incorporated to fill the scene with models. For the virtualization of objects from the real world, we process scans or images of the physical environment, perform detection and classification of objects based on the object recognition service and then translate virtualized objects into the scene model. This guarantees the continuous synchronization of real physical production environments and the virtual world.

4.1.3 3D Intention Visualization

3D intention visualization uses real and virtual world information for the visual communication of the entire assembly process and helps understanding the next steps in the cooperative assembly process by animating and simulating assembly instructions. In this way, users are well informed and able to control the entire assembly process. It also provides the possibility to the manufacturer to explore the whole assembly process by manually stepping through the manufacturing steps and viewing assembling information like part names and handling instructions.

4.1.3.1 Implementation Plan

The service will be integrated into an OpenGL-based application [Khronos Group, 1992] to visualize the collaborative working station and show animations and simulations of assembly instructions. In addition, a visualization of upcoming assembly steps is provided. Also an explorations mode is provided. The augmented virtual environment model serves as visualization input and the *semantic model* provides the information for the subsequent assembly steps.

4.1.4 Sensor Processing

Concerning the task of collaborative assembly of pneumatic cylinders, a sensor is used to support the human during the quality assessment, such as classifying the current assembly state and verifying that construction parts are correctly attached at the right

place and within a given measurement threshold. This process relies on scanned 3D data, point clouds in particular. Depending on type and complexity of the measurement task, the point cloud must be provided in different quantity (e.g., covering larger or smaller parts of the assembly object) and precision (e.g., 1mm or 0.05mm). To match these requirements while minimizing the scanning time, a 3D optical sensor based on structured laser light (line sectioning and space time analysis) is developed within this project. Figure 6 shows the design of the 3D laser scanner.

Figure 6: Design of the Fraunhofer 3D laser scanner

To ensure a high density of the resulting 3D point clouds and a low scanning overall time duration, the scanner operates at high frequency up to 300Hz. Hence, the low-level data processing, such as the laser line sectioning, retrieval and fusion of camera frames into a 3D point cloud must be carried out at corresponding frequency. Therefore, this low-level scan processing service must be locally and synchronously executed on a control PC and has to be directly connected with the scanner hardware and the robotic positioning system. In addition to 3D point clouds, the scanner provides also focused 2D color images for special quality assessment tasks, such as scratch, burr or spill detection from milling and drilling tasks.

Figure 7 shows the assembled 3D scanner with first results of a laser swipe on a target object producing a colored phase image.

Figure 7: Fraunhofer 3D Laser Scanner with first result phase images on a target object

This single image is then further processed to form a 3D scan of a single surface patch and multiple scan patches are combined to the complete 3D model of the object. The accuracy of the laser scanner was further improved by an additional post processing step using a mesh matching algorithm for aligning single scans containing consistent surface patches of the target object. To achieve this we combined an iterative closest point approach with a method derived from mesh less deformation techniques called shape matching. Shape matching fits a geometry mesh into a deformed version and computes the necessary transformation matrix for a best match in a single step.

The resulting algorithm is still iterative, however treating the found closest points as a "deformed" view of the points to be aligned allows to compute the transformation for the next iteration based on shape matching. This way the accuracy of the scanning process can be increased even in bad conditions. Figure 8 shows intermediate scanning results, left: without alignment, and right: after aligning single patch scans producing a more compact and accurate 3D model.

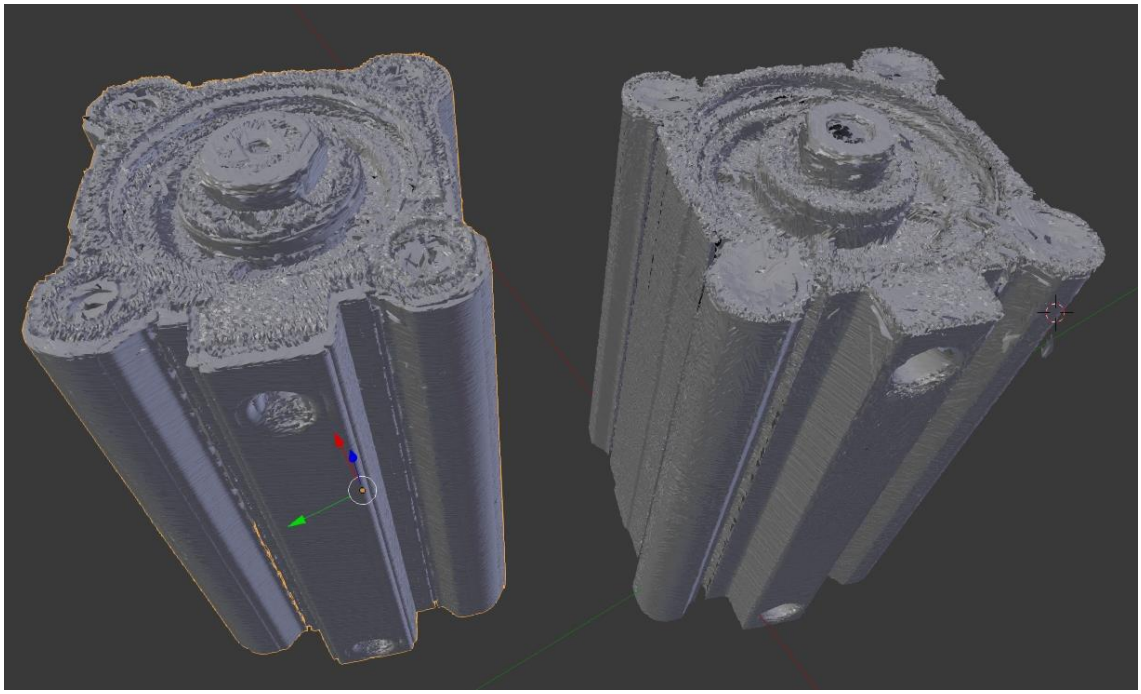

Figure 8: Intermediate scans of a pneumatic cylinder

The real-time sensor data is processed on-site forming a consistent 3D scan of the object that is then forwarded to other services, e.g., registration, classification and measurement, which can then potentially run remotely on the cloud.

4.1.4.1 Implementation Plan

The following implementation steps are carried out in successive order:

- Design of the 3D laser scanner, component selection, hardware interfacing and integration
- Implementation of the scanner calibration method for estimation of intrinsic and extrinsic sensor parameters
- Component for calibration of robotic positioning systems for scanners: hand-eye and turntable calibration
- Higher level interfacing with central workflow management system
- Improve scanning quality and robustness by advanced calibration and 3D matching algorithms

4.2 Cloud Services based on CloudiFacturing

4.2.1 Object Recognition

Many existing shape representations describe actual shapes, with visual and material properties. However, applications often require enhanced shape properties. For example, shape structure information, such as segmentation or label information, can help to relate the parts of a shape to each other. The 3D Object Recognition service can use engineered but also learned features in order to classify and recognize 3D objects. This involves also semantic segmentation by fitting geometric primitives in discrete geometric representations. This service will be used when it comes to recognize the current assembly state in the cooperative assembly scenario. Additionally, we provide the possibility of 2D image classification and training to detect and recognize parts placed in the working environment.

4.2.1.1 Implementation Plan

Our service provides different classification methods. On the one hand, we provide traditional learning approaches, like training a neural network. On the other hand, we examine classification algorithms that bypass required preceded time-consuming training process before usage needed by all learning approaches. This is especially beneficial in industrial scenarios where we have small lot sizes and many context switches. In these cases, it is not practical to spend a lot of time in training a classification system. For that we use our so-called registration-based object recognition service to classify the scanned part. The 2D image recognition is based on the open-source library OpenCV. A so called Cascade classifier is trained with a set of annotated positive and negative images and returns the recognized class.

4.2.2 3D Quality Control

Automatic Quality control by target/current comparison of point clouds with CAD models relieves the user of doing manual check of dimensions after the completion of an assembly. As input, the *3D Quality Control service* receives annotated CAD files with product manufacturing information (PMI) stating desired dimensions and tolerance ranges. The service analyzes a passed scan for the given requirements. As a result, the service provides the distance measurement with a note of measurement accuracy and a statement whether the measurement result lies in the tolerance range.

4.2.2.1 Implementation Plan

For the 3D Quality Control service, we decided upon the .prt and .asm file format for CAD annotations since they provide the functionality to store PMI besides geometry in one file format. In detail, annotations provide the information where, meaning between which elements, the measurement is taken, what distance is requested and what tolerance is accepted. Or they describe via notes on the model additional manufacturing information, e.g. which faces need to be greased during the assembly process. To realize the geometry and PMI extraction we use a CAD converter and importer based on the Spatial ACIS library [Spatial]. The extracted PMIs are then stored in a leaner data format so that they can be easily accessed from our application without loading the whole CAD file again. The extraction is usually performed in a pre-processing step before the actual assembly process begins to avoid longer loading time during the actual mounting. In the process of analyzing the point cloud, we extract higher-order primitives and perform correspondence analysis by registering and mapping detected primitives with CAD elements. Our developed 3D measurement system then provides specialized algorithms to measure in-between different representation formats (e.g. point cloud, parametric representations of planes, cones etc.).

5 Implementation Plan of Smart Production Line Services

5.1 Local Services

5.1.1 Product Identification

The product identification service recognizes identification data, such as barcodes, QR codes, and RFID (radio-frequency identification). The service sends requests to the product tracking and tracing service and reports to machines or information systems. This new digital memory platform is able to handle the data and to store all related data inside hardware memory. Continuous synchronization represented by W3C Object Memory Modelling allows users and machines to track and trace the product history and status.

5.1.1.1 Implementation Plan

We are improving the functionality by shifting the hardware from RFID to a small-embedded system. In the course of this shift, we integrate old-type ADOMe, based on W3C Object Memory Modeling, with FIWARE components so that users and machines can perform tracking and tracing. The second iteration prototype of this active product memory was developed and tested inside the factory line, its components and their relation to the overall structure is shown in Figure 9 and Figure 10.

5.1.1.2 Component Diagram

Figure 9 shows the components of the hardware for product identification service.

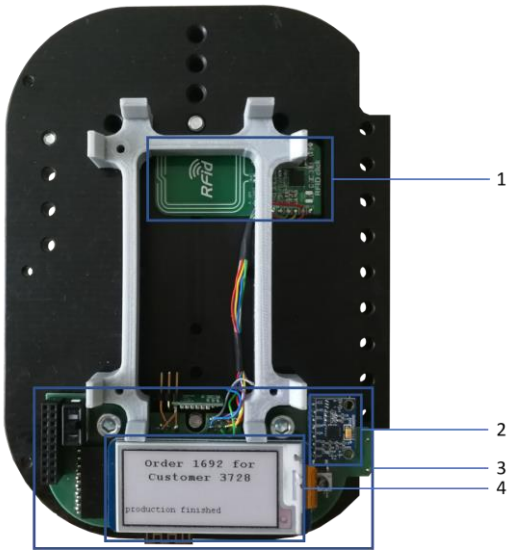

Figure 9: ADOMe second prototype with components

The components and their functions are:

1. RFID Reader/Writer to communicate with the product
2. Acceleration/Gyroscope sensor for dynamic product tracking and locating
3. Embedded hardware system running the IoT Communication Protocols and the Ontology
4. E-Ink Display to dynamically display QR / Barcodes / Human readable information

The usage of these components for the overall architecture of the product identification service is shown in Figure 10.

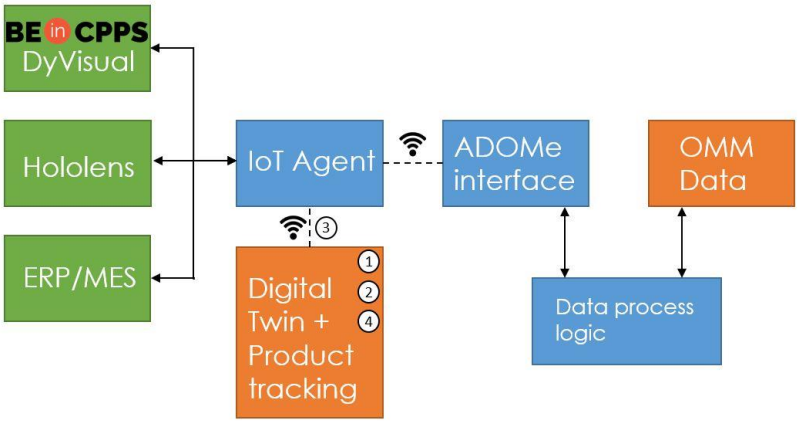

Figure 10: Components for product identification

5.1.2 Object Detection

This service is used to detect objects in an image (for video streams: individual frames). These objects have to be learned by a pre-trained residual convolutional neural network (resCNN) in the pre-deployment training phase. The trained network is then able to recognize the objects and their placement in an image and make them recognizable to the user. In addition, this information can be further used in processes, for example when the processing status of a product to be manufactured can be identified on the basis of the image.

In the production line there are plenty of things that need to be controlled and supported to boost up the speed of Human interaction. A Microsoft HoloLens, a pair of mixed reality smart glasses developed and manufactured by Microsoft, is used in the AUTOWARE system to add mixed reality technology to an industrial environment. Object detection based on our specific training data is not only helpful for monitoring, which is important for safety reasons, but also provides guidance in specific situations and tasks like e.g. manual assembly.

5.1.2.1 Implementation Plan

The following pipeline for creating an object detection classifier was implemented:

- Manually create and upload training data in the form of .jpg or .png images of the objects to be trained.
- Use of a binary program (labelImg) to create the bounding boxes and classes of the objects to be recognized. These are created in xml form.
- Selection of the pre-trained deep network and the corresponding setup file. These networks are based on the Tensorflow framework [Tensorflow] and use functions from OpenCV [OpenCV].
- Conversion of the training and test images with the corresponding xml file into data formats understandable for Tensorflow called "TFread"
- Train and evaluate the network using the Inference Graph
- Creating and outputting the model that can be applied locally

The following activities were selected for the implementation of the Finished Object Recognition:

- Deployment of the model created through the above pipeline to a powerful execution environment that is capable of massive parallel task executions (such as a graphics processing unit (GPU)).
- Transferring images from a HoloLens and from other sensors in the environment to the object detection service and return the result depending on which objects are detected and give relevant Information depending on the context (assembly steps)

5.1.2.2 Component Diagram

Figure 11 shows the data processing workflow of the object detection service.

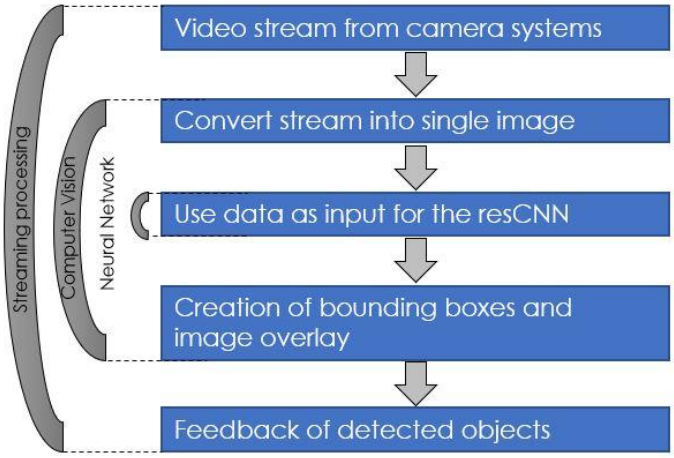

Figure 11: Data processing components for object detection

5.1.3 Non-Real-Time CPS Controller

This cyber-physical system (CPS) controller is used to collect non real-time critical data and preprocess it for later use. The data collected is the information sent by the OPC UA servers (Edge and Programmable Logic Controllers (PLC)) from the production modules themselves and from the Hololens and Active Product Memory, which is collected, pre-processed and forwarded for further analysis.

The edge server is supported by powerful hardware for parallel calculations (like a GPU), which provides it with additional specialized capacities for demanding calculations.

5.1.3.1 Implementation Plan

The system was implemented in several stages. First, access from the Edge Server to the IoT Agent (see Figure 10) was established to enable information connection from the Active Product Memory. A script was also created that accesses the Hololens' camera and sends the image to the server. On the server, Tensorflow and OpenCV, together with a Message Queuing Telemetry Transport (MQTT) protocol, were used to implement the necessary connections and data analysis. The information and bounding boxes created by the data evaluation were sent back to the SmartGlasses via a MQTT broker implemented on the SmartGlasses and displayed in the field of view.

5.1.3.2 Component Diagram

Figure 12 shows the components for the interaction between the Edge Server and the smart glasses.

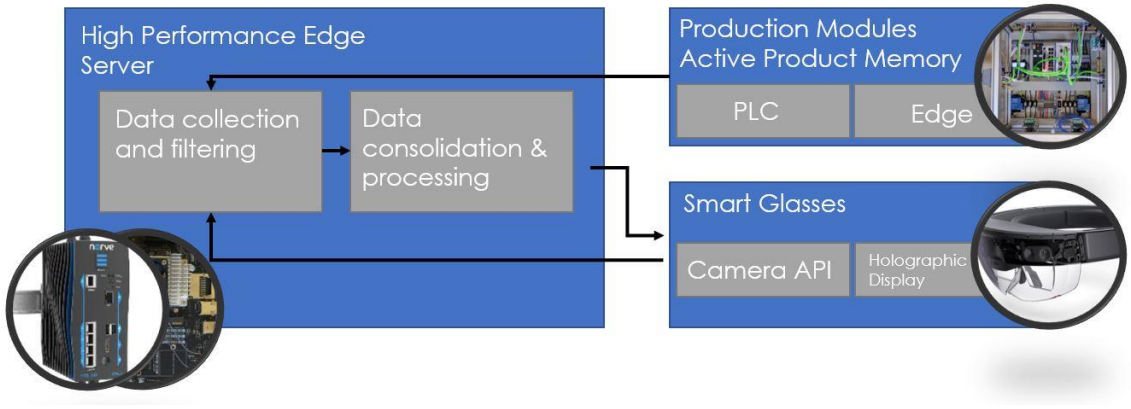

Figure 12: Components of the Non-Real-Time CPS Control

5.1.4 Safety Service

The *SmartFactory*KL production line is organized using individual plug & produce modules. There are several safety concepts to consider such as operation modes,

module detection, and human detection. This service is responsible for keeping the production line and human life safe e.g., stopping the production line if they are close to the moving sections of robots. Internal module safety means as well as cameras in the environment are monitored by this service.

3 cameras will be deployed in the SmartFactoryKL production line in order to detect the location of human operators. The size of the human operator in the captured image is used as reference to calculate the distance and the location. The digital twin records the location of human operators. Any dangerous situation triggers an event, which is connected with an alarm system through this safety service. This service then informs persons standing close to the dangerous machine about the safety risk. The next step would be to use human pose detection so that the safety system can predict possible danger in advance.

5.1.4.1 Implementation Plan

The digital twin updates the location and the distance of each human operator with the result of the object detection service. The result is provided by an edge server, which is connected with the cameras deployed in the shop floor. A single human operator is the target object of the rule-based danger detection. If the service detects any potentially hazardous situation, e.g. when the production module is open during operation, he displays an alarm message to the human operator when he/she is close to the hazardous area. Rule-based alarm detection is implemented in this service with area analysis and alarm interface.

5.1.5 Real-Time CPS Controller

In the field level of the AUTOWARE architecture, it may be necessary to make critical decisions within a limited time. Therefore, controlling CPS in real-time, can be useful in this scope. For example, in case of an emergency stop, the system is responsible to keep a human safe. Additionally, even without emergency stop, the system should be aware of human presence in the area where an emergency reaction may occur.

This kind of controller has to come to an exact-timed and correct decision, realized with proper hardware and software design as well as a real-time capable operating system (OS). For the implementation, it is necessary to have a hardware, which can handle real-time tasks. The software components inside the hardware will virtualize the processors and perform additional scheduling to respond to the requests on time. In addition, it will monitor its resources to keep the Quality of Service (QoS) at maximum and will offload the tasks when necessary. Figure 13**Error! Reference source not found.** shows the required software components for such an Edge Server architecture.

An important extension in the integration of real-time critical data is the use of communication protocols that can guarantee the timely arrival of the data. Time Sensitive Networking (TSN) is a set of standards that can ensure the prioritization of data packets and thus their arrival on the server. However, this requires the use of switches and controllers that support these standards.

5.1.5.1 Implementation Plan

The implementation of the software components will be made using C, C++ or Java. For the OS, a real-time patched or patchable Linux-based system will be used. The hardware is expected to work compatibly with different hardware and is planned to be plug'n'play with minimum adjustments to be ready for the service.

5.1.5.2 Component Diagram

Figure 13 shows the architectural composition of the Edge Server.

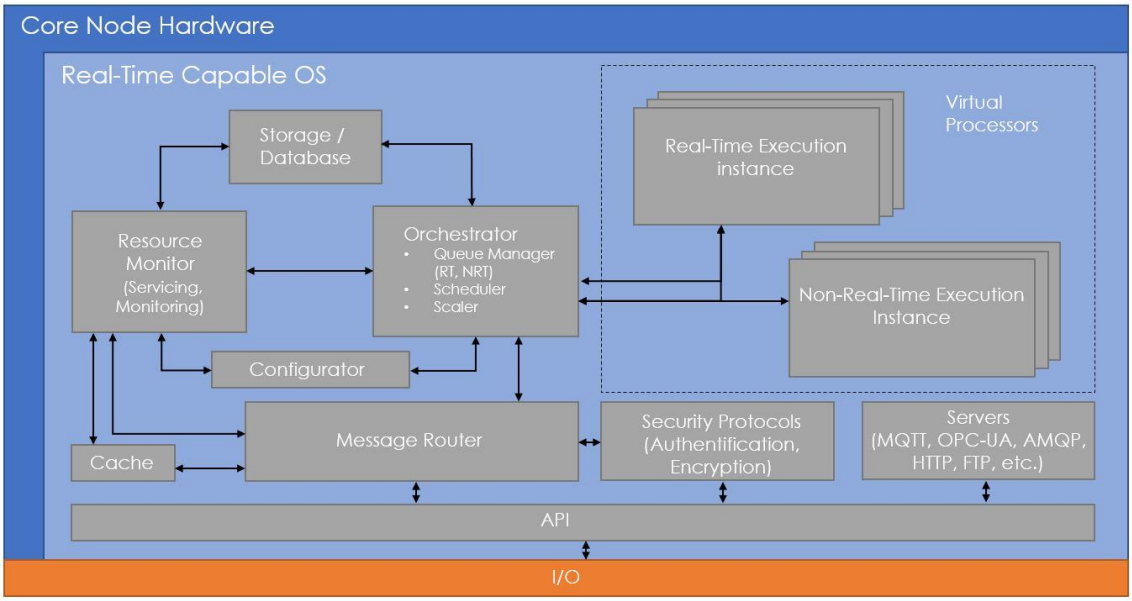

Figure 13: Real-Time capable server architecture

5.1.6 PLC and OPC-UA server:

The legacy PLC system needs a vender-library based client service because the OPC-UA server could be an additional module, which needs to be integrated in some cases. Retrofitting to legacy PLC is a practical approach in the deployment of actual shop-floor environments. Advanced PLC controllers usually have an OPC-UA server integrated. However, typically OPC-UA servers have to be installed additionally or need to be developed.

5.1.6.1 Implementation Plan

The implementation covers the following two components:

- Topology manager: Many of the PLC vendors provide .NET based SDKs and examples.
- OPC-UA server: Multi-vendor PLCs connect with each other via OPC-UA protocols. Each production module uses different PLC systems and OPC-UA servers.

5.2 HPC Services

5.2.1 Product Tracking & Tracing

The product tracking and tracing service retrieves product data, which is stored in a product database, and updates its status when any machine or device requests an item detected at a specific location. The product database manages CAD, bill of material, assembly instruction, process progress, and many more.

The message interface collects MQTT from production modules so that a centralized update can guarantee consistence. The database interface communicates with the central databased tracking. Additionally, this service also provides a search interface.

5.2.1.1 Implementation Plan

The implementation of this service requires the following components:

- .Net framework
- AMPQ and MQTT message
- MySQL database and SQL messages

5.2.2 Integration into the existing infrastructure

The current production line is already equipped with the IBM Integration BUS (IIB) and the ProAlpha ERP system. An example for the IIB is shown in Figure 14.

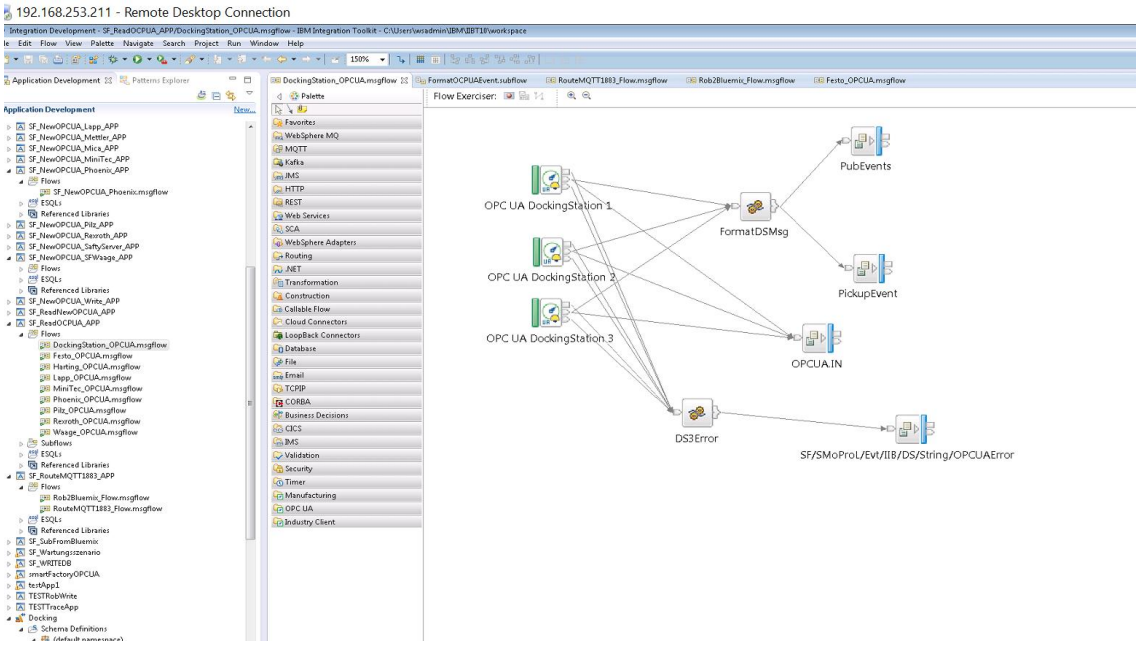

Figure 14: Exemplary implementation of the Integration BUS

5.3 Edge Services

5.3.1 3D Factory Visualization

A digital twin of the production modules is used to animate all events and is visualized on the shop floor dashboard. The 3D model engine of this visualization is deployed as web server to synchronize the virtual model displayed in a web-interface to multiple-users.

A front-end XML3D interface connects with the web server for replacing various Dynamic LinK Libraries (DLL) libraries dynamically. This visualization interface is the message interface DLL library of the webserver.

In the new version of the visualization software, the animations are encapsulated so that individual services can be created and assigned to several modules. Uniform animations for standardized operations (such as the assembly line, the opening/closing of locks) can thus be better implemented.

5.3.1.1 Implementation Plan

The implementation of this service requires the following components:

- XML3D JavaScript library
- .NET framework
- FIWARE architecture
- MQTT message

In addition, modules must be created for individual animations. An example of such an animation is shown in Figure 15 as a sequence of images to insert the bracket.

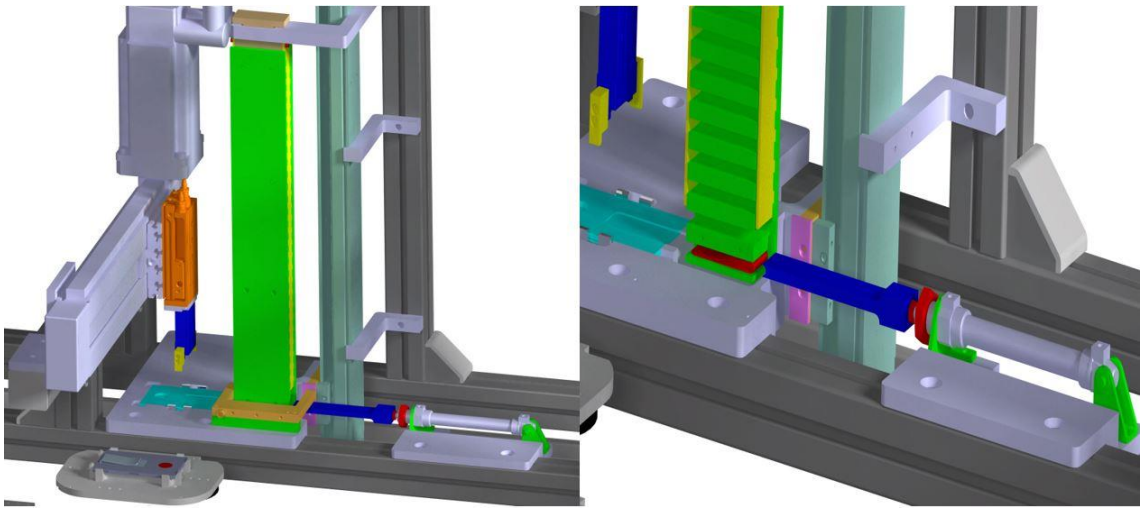

Figure 15: DyVisual visualizing tool

5.3.2 Process Optimization

This service generates alternative plans based on given rules to solve the problem of machine breakdown or quality problems and performs a comparison between different plans. The track and trace service provides the status of the real environment to this service to update the initial condition of the optimization algorithm.

This service is foreseen to realize the lean manufacturing concept in module production lines. In addition, it optimizes the production schedule with given order and sequences by finding the bottleneck in the module in real-time. A capacity analysis algorithm keeps calculating the resource utilization and flow status by collecting product data from ADOMe (active digital memory). In addition, the algorithm can derive a whole order estimation so that process planners can optimize the production schedule.

The Process Optimization Service communicates with the Active Product Memories. If the processing sequence of the products to be processed is changed, a message is sent to the affected products so that they can reorganize themselves.

5.3.2.1 Implementation Plan

This service is a Docker service deployed onto an edge server. A mathematical algorithm runs on an EXCEL-based program and is updated by a message interface. It is displayed on a web-based user interface, which remote users can access. We use Node-red as platform to define message interfaces between external information

(described in MQTT), a simplex-method based algorithm (CPLEX) and the GUI web page.

5.3.3 Training Object Model

This module is a service for improving the training data set of the object detection service. It can contain images of human operators, products, work-in-process states and machines or their parts.

The current production line assembles 3 parts for each business cardholder. To support human operators who will assembly the cardholders manually, meaning each 3 parts of 3 different card holder models, the object model needs to update its training data with new models and new products. This training data will contain different pictures of all parts produced by the SmartFactoryKL production line and will be used for detecting them in the human operator's device. Each edge server focuses on a set of production modules and needs to shift its training object model if the production line is rearranged. The training object model is a Docker based storage, which can be replaced by other ones.

5.3.3.1 Implementation Plan

The object model will be described in a TFread file format with 200 pictures for each training set. The object model is filled with pictures of hand-gripped single parts captured from various angles. This service is a Docker service, which can be accessed from the object detection service and can replace the object model when a new production module is needed. The implementation will be based on a binary program file, which can be utilized to create the classes and bounding boxes of the images to be classified.

6 Implementation Plan of Cognitive Automation Validation Services

The implementation plan of the Cognitive Automation Validation services provides the implementation of the services listed in Deliverable D3.2a *Software Defined Autonomous Service Platform development* provided by GPflowOpt. These consist of services related to the rapid reconfiguration of the cognitive vision systems but do not encompass the related services concerning the deep learning based vision system, the conveyor belt actuation, the image processing by the camera nor the robotic gripper. However these are included in the instantiation service overview (Figure 16) and they will be briefly discussed.

As a reminder the following services in GPflowOpt are provided:

1. Setting up a new Optimization experiment (deep learning training)
2. Adding data (new observations) to the Optimization experiment (the addition of new labeled data)
3. Requesting the current optimal parameter setting according to the currently observed data (= the next potentially best setting for the deep learning system).

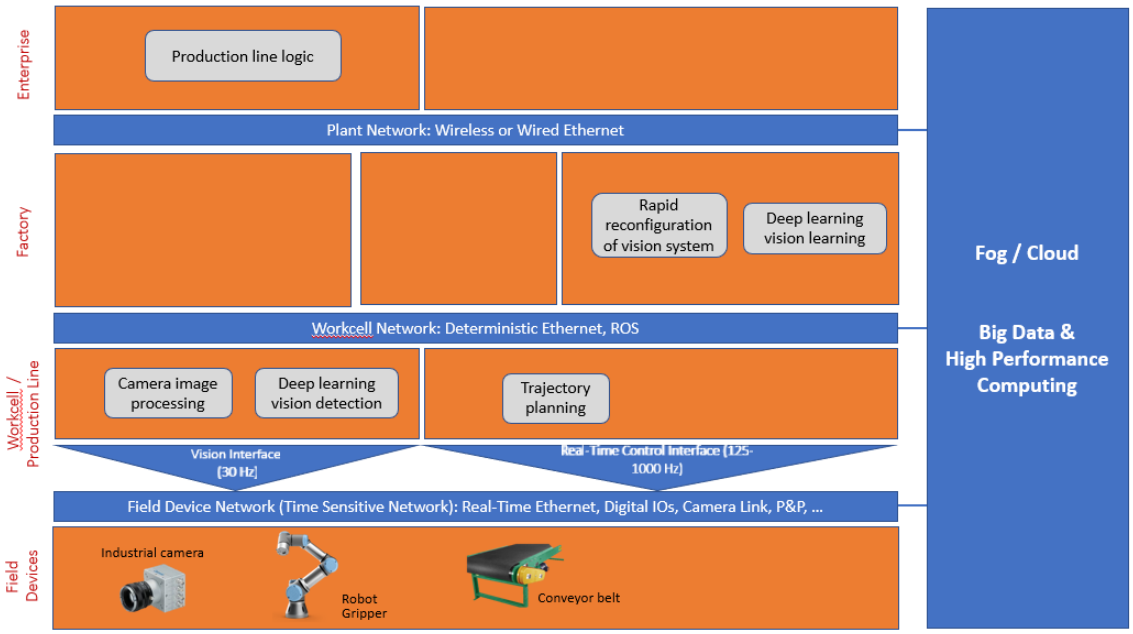

Figure 16: Instantiation of services provided for Cognitive Automation Validation

6.1 Local Services

6.1.1 Trajectory planning, camera image processing and deep learning vision detection

The local services are time-critical as they need to be performed on a fast moving conveyor belt. In a first phase, the camera system continuously provides images directly transformed by wire to a local processing unit containing the necessary hardware and processing power (GPU, TPU - Tensor processing unit: an application specific integrated circuit specifically used for neural network AI) to perform rapid detection based on a deployed deep learning model. The system extracts the relevant coordinates based on the automatic detection, translates these coordinates based on the speed of the conveyor belt and transfers these by ROS or in JSON format to the gripper. At this moment, no specific gripping points are calculated but instead the center of the desired object is returned. The gripper is then actuated to move towards the coordinates and will perform an action. Due to the limited speed of the currently available gripper and the limited length of the conveyor belt, only services related for demonstration purposes have been developed.

6.1.1.1 Implementation Plan

The implementation plan foresees the adoption of ROS or a standard JSON HTTP transfer to communicate the gripping coordinates to the robotic gripper. As input for the deep learning vision, commonly used file formats are used and transformed to images in the RGB color model format and resized to the neural network size.

6.2 HPC Services

6.2.1 Deep learning vision learning services

The initial deep learning vision setup might require processing on HPC infrastructure depending on the use case. As no particular deep learning architecture is put forth within the context scope, any framework can be used. The model can then be downloaded and reloaded locally or reconfigured in the edge node.

6.2.1.1 Implementation Plan

At this moment, Pytorch, Caffe, Tensorflow are the most commonly used technologies to obtain such systems.

6.3 Edge Services

6.3.1 Rapid reconfiguration of deep learning based vision systems services

Rapid reconfiguration is performed in the edge computing facilities. For this the original model parameters are reinitialized within limited bounds to accommodate a changing environment on the production line. Given labeled examples of the new setting, the operator starts a reconfiguration experiment based on the initial model. At the start, several parameters are explored in a space filling design and the accuracy of the obtained systems is compared. This is obtained by the new Optimization experiment procedure of the GPflowOpt software installed on the edge node or by remote access via the Representational State Transfer (REST) interface. In a next phase, the accuracy is further increased by incremental model builds based on the feedback of the system until the required accuracy is reached or allowed time slot has passed. Figure 17 shows an exemplary case for the reconfiguration service. The x and y axis represent the value assignment of two parameters.

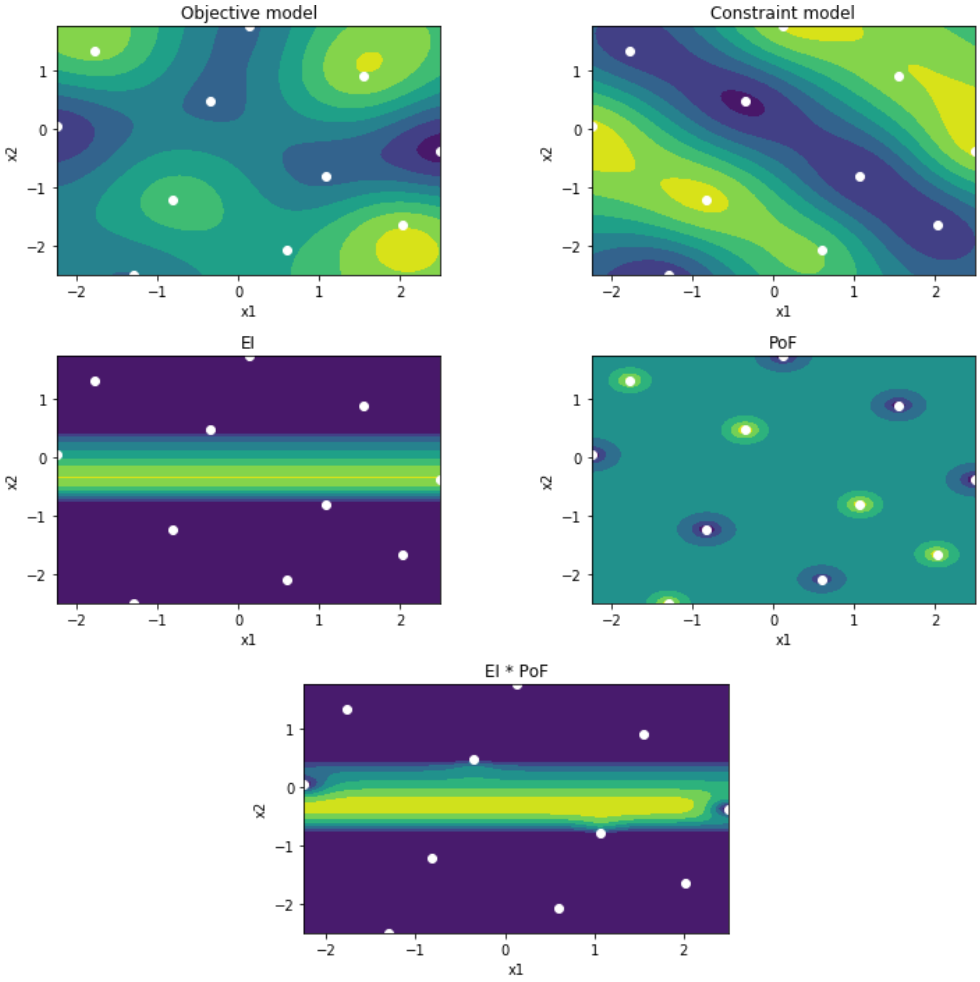

Figure 17: Example of the reconfiguration service in process for a constrained problem.

The samples indicated as white dots represent validated systems. The top left plot indicates the obtained accuracy, the top right one the feasible regions. The middle plots show the regions where the most improvement is possible. While the bottom plot combines the probability of improvement with the probably of feasibility.

6.3.1.1 Implementation Plan

The implementation plan foresees the communication between the deep learning framework used and the GPflowOpt service. Both are available as Python 3.0 frameworks.

7 Implementation Plan of Collaborative Robotics Services

TEKNIKER implements a neutral facility, a collaborative work cell. The main outcome is the development and integration of a mobile robotic platform services for logistics operations. The services are based on the development of autonomous navigation strategies and gasic algorithmic capabilities for environment perception, planning and control of the mobile robotic platforms. Also, the coordination of navigation strategies of mobile platform floats, necessary to implement logistic operations for supplying in different production tasks.

In AUTOWARE these services will be integrated following the reference architecture (see Figure 18). From the software architecture perspective, they will be designed and implemented as ROS services. These services will be the basis

- to provide functionality to the neutral experimentation activity.
- to evaluate the implementation of AUTOWARE components.

In this document, the main functionalities and their integration in the general architecture are described.

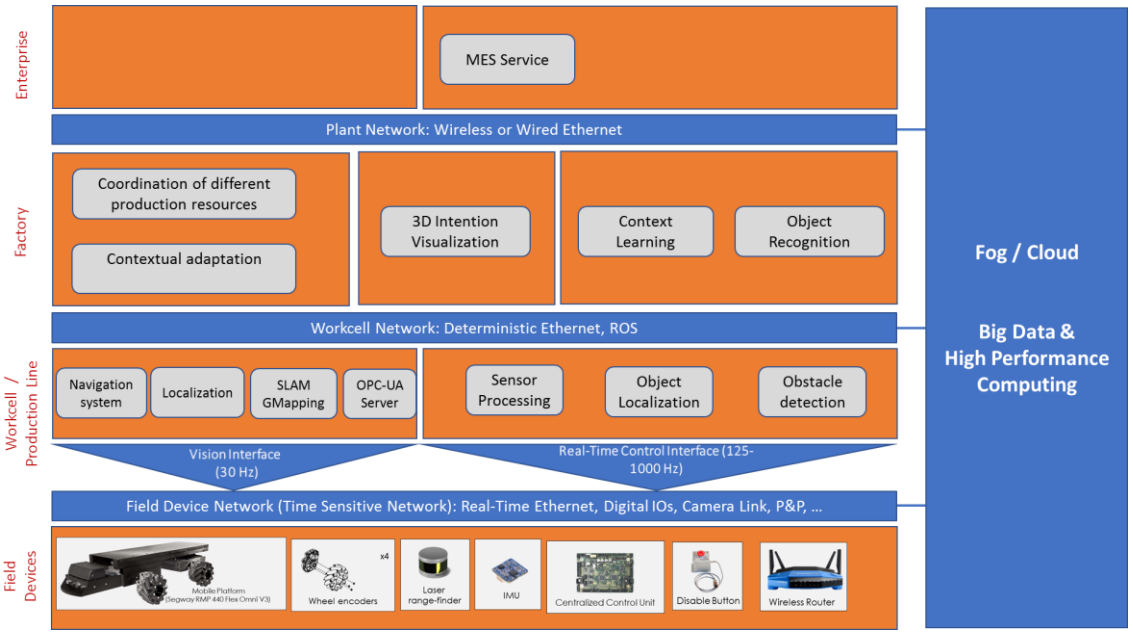

Figure 18: Instantiation of services provided for autonomous navigation of mobile robots

7.1 Local Services

7.1.1 Navigation system

The aim of this service is to provide the logistic basic moving abilities to the robotic mobile platforms. It can be used to assign destinations to the mobile platform and receive feedback about the task status. It is the basic service to build logistic solutions at factory level.

ROS is the core framework for the navigation system. The navigation system will calculate where the robotic platform is, both in absolute coordinates and relatively to a local reference point. Then, for a given target position and based on the current position, it will determine an optimal path. The Navigation module generates lower level motion commands in order to follow the path while avoiding obstacles and re-planning when necessary. These motion commands are sent to the Filed Devices layer. This system is composed of several modules shown in Figure 18 that work together to provide the localization and navigation capability (SLAM, Localization). The component diagram is shown in Figure 19.

The following submodules of the navigation module are used:

- The localization module is responsible for the generation of precise position information based on sensor information, inertial and odometer measurements. The purpose of this module is to generate global localization information, which is based on a global reference frame.

- Global Costmapper: Performs the generation of a global cost map in 2D. A cost map is a grid in which each cell has an associated numeric value regarding the presence of obstacles.
- Local Costmapper: Same as the global version but reduced to an area around the current pose of the robot with a predefined and fixed size.
- Global Planner: Given a goal and a global cost map, it generates a path to follow between the current position in the map and the goal, based on the cost values associated with the traversable and not traversable areas.
- Local Planner: It generates motion commands to navigation on a plane, given a path to follow and a local cost map.

7.1.1.1 Description, Input, Output

As input, the service receives:

- Destination to be reached by the mobile platform.
- The service returns the robot status in terms of: position, status (working, goal reached, error), battery level.

7.1.1.2 Implementation Plan

The service is provided by a ROS node (module). The drivers manage the communication between hardware devices and the modules of the upper layer (Field-Work cell). The main purpose is to provide a hardware independent interface to the upper level. This is achieved by implementing the drivers as ROS nodes that exchange Standard ROS Messages.

7.1.1.3 Component Diagram

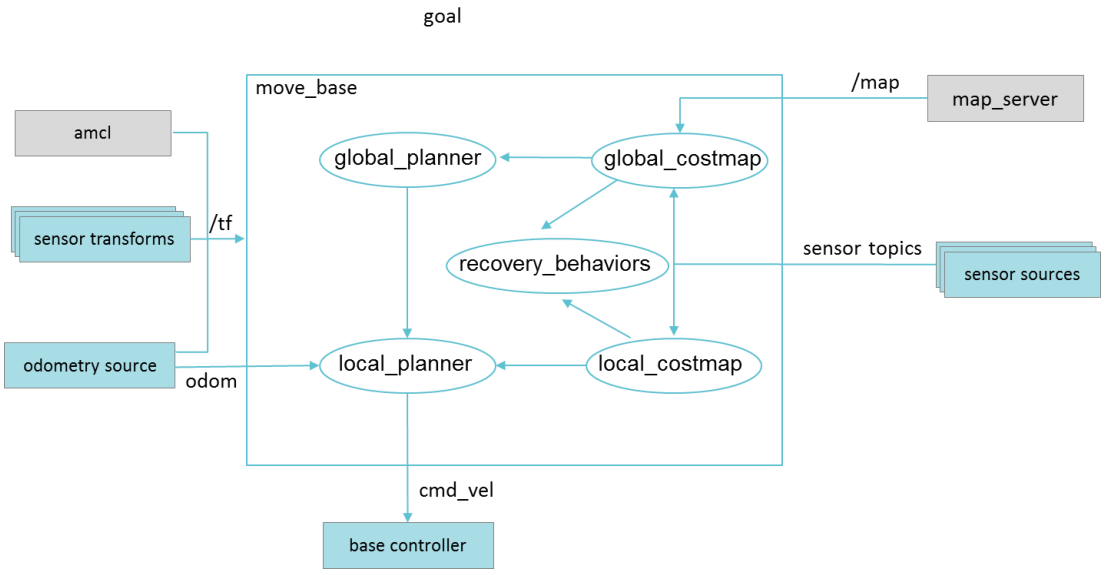

Figure 19: Component Diagram of the navigation module

7.2 Cloud Services

7.2.1 Standard input/output communications to ROS

Most of the modules are implemented using the ROS framework, so they will have access to all ROS communication infrastructure. However, there will be some non-ROS modules that need to communicate with ROS modules by other means.

The OPC UA mechanism is implemented to provide standard communication mechanisms between work cell and Factory Level. The architecture implemented is shown in Figure 20.

7.2.1.1 Implementation plan

The service is provided by a ROS node (module).

7.2.1.2 Component diagram

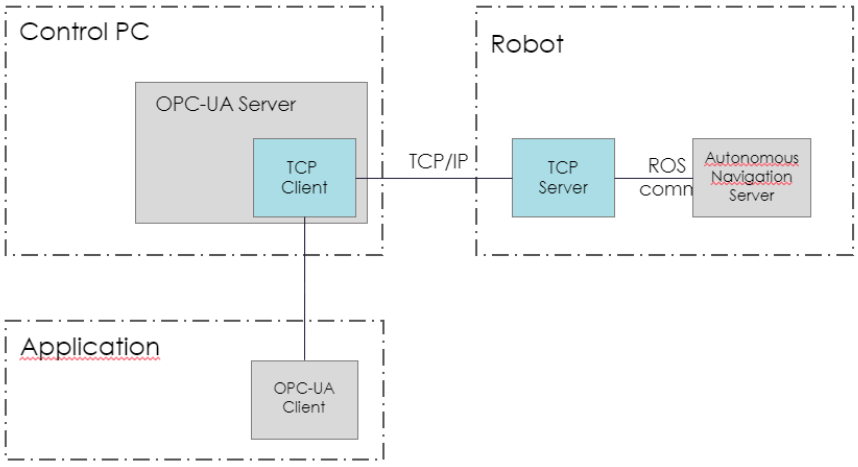

Figure 20: OPC-UA services implementation for Autonomous navigation

8 Summary and Outlook

This document is an update of the development plan described in the previous deliverable D3.2a where a detailed overview of the development and implementation plan of the services placed in the SDA-SP was given. The services provided by the SDA-SP will be used for the implementation of the industrial and neutral use cases in WP5.

The service development partners of WP3 and TEKINKER are using different infrastructures and platforms (ROS, CloudiFacturing and GPflowOpt) to develop and provide their services to the AUTOWARE project. From a conceptual point of view, the SDA-SP provides a generalized reference architecture for cognitive applications spanning over different platforms to integrate and illustrate all AUTOWARE services.

The updated development plans of the services related to the implementation of the main three WP3 assets (a reconfigurable robotic work cell, a mixed or dual reality supported automation to implement an effective and flexible collaboration between humans and robots, and a multi-stage production line) and two newly added instantiations and development plans (cognitive automation validation and collaborative robotics) are presented in more detail in this document.

The next deliverable (D3.3b, due in M30) will provide an overall update of the existing reference implementations of the SDA-SP, taking into account the two more instantiations of the SDA-SP reference architecture, and the description of the integration strategy.

References

Gerkey, Brian (2017) "Why ROS 2.0?" ROS 2.0 Design, Online: http://design.ros2.org/articles/why_ros2.html, Last Accessed 26.02.2019

Khronos Group (1992), OpenGL - Open Graphics Library , Online: https://www.opengl.org/, Last Accessed 26.02.2019

Lamy, Jean-Baptiste (2017), "Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies" Artificial Intelligence in Medicine, Elsevier, 80, pp.11 - 28.

Merkel, Dirk (2014), "Docker: lightweight Linux containers for consistent development and deployment" Linux Journal 2014, Issue 239

Musen, M.A. (2015), "The Protégé project: A look back and a look forward". AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), Online: https://protege.stanford.edu/, Last Accessed 26.02.2019

OpenCV, Open Computer Vision Framework, Online: https://opencv.org/, Last Accessed 26.02.2019

Spatial Corporation, 3D ACIS Modeler (ACIS), Online: https://www.spatial.com, Last Accessed 26.02.2019

Tellez, Ricardo (2018) "On not using ROS for your robotics startup." Learn and Develop for Robots Using ROS - The Construct, Online: http://www.theconstructsim.com/not-using-ros-robotics-product/, Last Accessed 26.02.2019.

Tensorflow, Deep Learning Framework, Online: https://www.tensorflow.org/, Last Accessed 26.02.2019