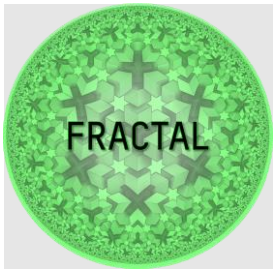

D2.1 Platform Specification (a)

Deliverable Id:	**D2.1**
Deliverable name:	**Platform Specification (a)**
Status:	**Final**
Dissemination level:	**Public**
Due date of deliverable:	**2020-02-28 (M6)**
Actual submission date:	**2020-02-26**
Work package:	**WP2 "Specifications & Methodology"**
Organization name of lead contractor for this deliverable:	Thales Research & Technology
Authors:	Jérôme Quévremont, Thales Frank K. Gürkaynak, ETH Zürich Michael Gautschi, ACP Markus Postl, Virtual Vehicle Martín Rivas, Prointec Christina Schwarz, AVL Bekim Chilku, Siemens Martin Matschnig, Siemens Mikel Labayen, CAF Jordi Mansanet, Solver Machine Learning Tuomas Paso, University of Oulu Lauri Lovén, University of Oulu Ville Niemelä, University of Oulu Panos Kostakos, University of Oulu Arash Sattari, University of Oulu Rouhollah Ehsani, University of Oulu Pierluigi Scarpa, MODIS Tania Di Mascio, Università degli Studi dell'Aquila Enrico Ferrari, Rulex José Ramón Juárez, IKERLAN Antti Takaluoma, Offcode Ester Sola, Zylk Alfonso González, Zylk Iñigo Angulo, Zylk Damiano Vallocchia, Ro Technology Rubén Lorenzo, BSC

	Sergi Alcaide, BSC Jaume Abella, BSC Daniel Onwuchekwa, University of Siegen Nambinina Andrianoelisoa Rakotojaona, University of Siegen Carlos Lua Morales, University of Siegen Matti Vakkuri, Haltian Jyrki Okkonen, Haltian Alexander Flick, PLC2 Giacomo Valente, Università degli Studi dell'Aquila Paolo Giammatteo Università degli Studi dell'Aquila Carles Hernández, UPV Edurne Palacio, IKERLAN Leire Rubio, IKERLAN Kevin Villalobos, IKERLAN Paolo Burgio, UNIMORE Fernando Eizaguirre, IKERLAN Stefano Delucchi, AITEK Sébastien Jacq, Thales Sylvain Girbal, Thales Igor Bisio, University of Genoa Ander Galisteo, IKERLAN
Reviewers:	Leire Rubio, IKERLAN – technical coordinator Iñaki Paz Rey, LKS Antti Takaluoma, Offcode

Abstract:
D2.1 "Platform Specification (a)" captures the requirements of the FRACTAL project. A top-down approach has been adopted, starting from the needs expressed by the eight use cases, spanning through the main features (AI and safe autonomous decision, mutable and fractal communication, safety, security and low-power techniques) and then flowing down to both platforms, the commercial node (Xilinx VERSAL) and the flexible node (PULP).

Contents

1 History 7

2 Summary 8

3 Introduction 9

3.1 Organization of the specification 10

4 Use case requirements 12

4.1 VER-UC1: Improving the quality of engineering and maintenance works through drones 13

4.1.1 Description of the use case 13

4.1.2 Roadmap to achieve use case KPI and objectives 15

4.1.3 Requirements 16

4.2 VER-UC2: Improving the quality of automotive air control 19

4.2.1 Description of the use case 19

4.2.2 Roadmap to achieve use case KPI and objectives 23

4.2.3 Requirements 24

4.3 VER-UC3: Smart meters for everyone 27

4.3.1 Description of the use case 27

4.3.2 Roadmap to achieve use case KPI and objectives 28

4.3.3 Requirements 29

4.4 VER-UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications 33

4.4.1 Description of the use case 33

4.4.2 Roadmap to achieve use case KPI and objectives 37

4.4.3 Requirements 38

4.5 VAL-UC5: Increasing the safety of an autonomous train through AI techniques 40

4.5.1 Description of the use case 40

4.5.2 Specific technical objectives 41

4.5.3 More generic objectives 41

4.5.4 Roadmap to achieve use case KPI and objectives 42

4.5.5 Requirements 43

4.6 VAL-UC6: Elaborate data collected using heterogeneous technologies 45

4.6.1 Description of the use case 45

4.6.2 Roadmap to achieve use case KPI and objectives 47

4.6.3 Requirements 48

4.7 VAL-UC7: Autonomous robot for implementing safe movements 51

4.7.1 Description of the use case 51

4.7.2 Roadmap to achieve use case KPI and objectives 53

4.7.3 Requirements 53

4.8 VAL-UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse 56
4.8.1 Description of the use case 56
4.8.2 Roadmap to achieve use case KPI and objectives 57
4.8.3 Requirements 57
5 Safety, Security & Low Power Techniques 59
5.1 Interconnection Architecture 59
5.2 Low Power Services 61
5.2.1 Node level 61
5.2.2 System level 62
5.3 Safety Services 63
5.4 Security Services 64
5.4.1 Node level 65
5.4.2 System level 65
5.5 Development methods in time-triggered scheduling 65
5.6 Requirements flowing down to WP3 66
6 AI and safe autonomous decision 67
6.1 Communication requirements 68
6.2 Distribution needs 69
6.2.1 Centralization vs. decentralization 70
6.2.2 Hierarchy 70
6.2.3 Opportunism / dynamicity 71
6.2.4 Use case needs for Distributed Artificial Intelligence 71
6.3 AI Performance requirements 77
6.3.1 Efficiency 77
6.3.2 Effectiveness 78
6.3.3 Reliability and availability 78
6.3.4 Use case needs for AI performance 79
6.4 Data & model lifecycle concept 83
6.4.1 Embedded and edge machine learning algorithms 83
6.4.2 Distributed machine learning platform 84
6.4.3 Data management 85
6.4.4 Security and privacy 86
6.4.5 AI ethics 86
6.5 Inference requirements 87
6.5.1 Use case needs for inference 88
6.6 Learning requirements 95
6.6.1 Use case needs for learning 95
6.7 Run & development environment requirements 98
6.7.1 Available tools 99
6.7.2 Usable technologies / technology stacks 100
6.7.3 Interoperability & integrations with other systems 100

6.7.4 Continuous integration / DevOps platform 101
6.7.5 Use case needs for run & development environment 101
6.8 Requirements flowing down to WP3 107
7 Mutable and fractal communications 109
7.1 Edge node design and implementation 110
7.2 Edge center controller infrastructure 111
7.3 Validation of the edge computing architecture 114
7.4 Integration, testing and validation of standalone communication sub-systems 115
7.5 Requirements flowing down to WP3 116
8 Node architecture and building blocks 118
8.1 VERSAL-based node 118
8.1.1 Hardware requirements 120
8.1.2 Software requirements 125
8.2 RISC-V (PULP) based node 126
8.2.1 Hardware requirements 130
8.2.2 Software requirements 135
9 Conclusions 140
10 List of Figures 141
11 List of Tables 142
12 List of Abbreviations 144

Acknowledgement

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Italy, Austria, Germany, Finland and Switzerland.

1 History

Version	Date	Modification reason	Modified by
0.0	2020-10-16	Template	Thales
0.1	2021-02-11	Complete version	Authors
0.2	2021-02-25	Reviewed version	Reviewers and authors
1.0	2021-02-26	Final clean-up, delivered version	Thales
1.1	2021-08-12	Added ECSEL logo, fixed two figures.	Thales
1.2	2021-08-13	Improved previously fixed figure	Thales, UNIVAQ
1.3	2021-12-17	Addressed comments from the first period review, delivered version	Thales

Table 1 – Document history

To cope with the high number of contributors, this document has been edited online. The Microsoft Sharepoint solution has been selected to keep information under EU legislation. This solution offers a reduced feature set compared to a "regular" Word editor. For instance, we have not been able to build a table of references and have instead used footnotes.

Use case and work package leaders have driven the sections that relate to their UCs and WPs. Therefore, formats can slightly differ between sections.

2 Summary

According to the DoA, D2.1 "Platform Specification (a)" captures the requirements of the FRACTAL project in a top-down fashion.

The document starts with a description of the eight verification and validation use cases and the capture of their needs:

- VER-UC1: Improving the quality of engineering and maintenance works through drones
- VER-UC2: Improving the quality of automotive air control
- VER-UC3: Smart meters for everyone
- VER-UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications
- VAL-UC5: Increasing the safety of an autonomous train through AI techniques
- VAL-UC6: Elaborate data collected using heterogeneous technologies (intelligent totem)
- VAL-UC7: Autonomous robot for implementing safe movements
- VAL-UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse

The features needed by these use cases have been devised to the three "features" or "services" work packages:

- WP4: Safety, Security & Low Power Techniques (objective O2)
- WP5: AI and safe autonomous decision (objective O3)
- WP6: Mutable and fractal communications (objective O4)

Finally, use cases and WP4/5/6 requirements have been flown down to the platforms borne by WP3 "Node architecture and building blocks" (objective O1):

- VERSAL-based commercial node
- PULP-based flexible node.

A list of abbreviations is available at the end of the document.

An update, D2.3 "Platform Specification (b)", is planned at M22 (2022-06-30).

3 Introduction

Artificial Intelligence is key for the IoT to enhance existing services and to operate in a more efficient manner. If AI is not implemented in the IoT its scope is very much limited. Cognitivity, provided by Artificial Intelligence, will support the IoT to adapt to surrounding world changes, learning in real-time to improve its performance.

The goal of FRACTAL project is to create a basic building block called the FRACTAL node. This building block is a reliable computing platform node able to build a Cognitive Edge (a network that makes predictions and diagnoses) under industry standards. The FRACTAL node will be the building block of scalable decentralized Internet of Things (ranging from Smart Low-Energy Computing Systems to High-Performance Computing Edge Nodes).

On the one hand, this deliverable focuses on presenting the demonstrators and justify their relevance. The demonstrators are grouped in verification and validation use cases that represent high level class of applications (Transport & Mobility, Digital Life, Digital Industry and Energy). This document details, for each demonstrator, its context and how it allows to address the top-level objectives of FRACTAL project.

In order to have a more precise description of validation and verification needs from use cases, the features the use cases depend upon are expressed as requirements. Those requirements will be further developed and refined in the technical-work package deliverables. Although, they are already effective at defining the demonstrators, their scope and challenges.

On the other hand, this deliverable aims to collect the FRACTAL platform requirements in a high-level format to be able to draw from the very beginning the FRACTAL project concept.

This document provides significant requirements that clarify the following items:

- Definition the structure that should be provided (components, tools, methodology and workflow) to be a key technology enabler for CPS (Cyber Physical Systems) at edge computing level.
- Limitation of the scope of the project and better target the domains considered in the project, the use case requirements will be used as inputs for the specification.
- The high-level requirements, as the outcome of this deliverable, will draw the roadmap for the next steps of the project, against which the ensuing results will be benchmarked.
- Requirement fulfilment will allow to track the progress, the verification and the validation of the FRACTAL Platform.

Solution providers will build the nodes to fulfil the demonstrator missions. This deliverable identifies the initial components that will be integrated and validated in each use case and demonstrator. Hence, each demonstrator defines Key Performance Indicators (KPIs) that will allow this measure. These KPIs will be managed and tracked in WP7 and WP8 devoted to demonstrators.

D2.3 "Platform Specification (b)" will be an update of the document, planned at month M22. Among others, it will address the "TBD" (to be defined) included in the current version.

3.1 Organization of the specification

There is not in the literature a single definition of what a *specification* is. In this deliverable, the *specification* word is understood as "a set of documented requirements to be satisfied by a material, design, product, or service"[1]. Therefore, this document focuses on capturing the needs that shall be satisfied in FRACTAL. This document also serves as a transition to WP3/4/5/6 where solutions are devised to fulfill these requirements.

According to the Part A of the *description of the action* of the FRACTAL project for WP2,

> *"To limit the scope and better target the domains considered in the project, the use cases specifications will be used as inputs for the specification. The high-level requirements, as the outcome of this task, should become a roadmap for the remainder of the project, against which the ensuing results should be benchmarked."*

Therefore, the capture of requirements for this specification will mostly be driven by use cases in a **top-down** approach, according to Figure 1.

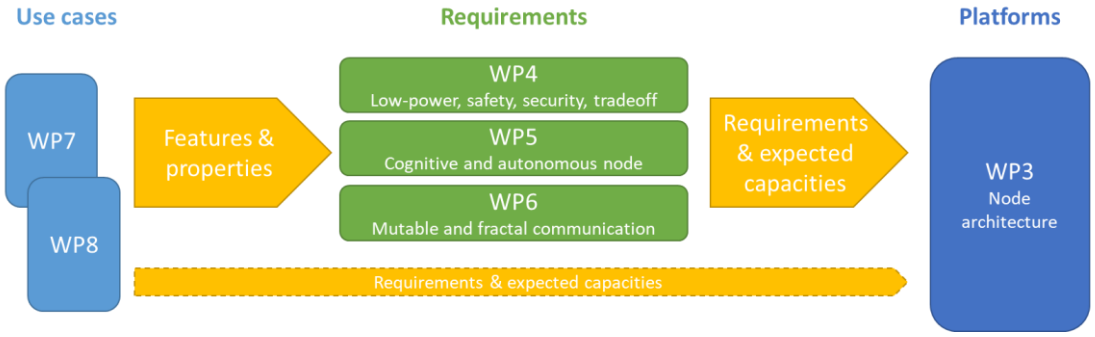

Figure 1 – Top-down approach of the FRACTAL specification

The requirements for the node architecture (WP3) will in most cases derive from "services" (WP4/5/6), but there can be direct inputs from use cases.

To reflect the top-down approach, this document is organized in the reverse order of work package numbers, starting with an analysis at the use case level (WP7/8), continuing with "services" (WP4/5/6) and closing with the node architecture (WP3).

In some cases, a complementary **bottom-up** approach can be used, where technology providers define their solutions not being driven by use cases. This can

[1] Form and Style of Standards, ASTM Blue Book. ASTM International. 2012.

be the case when top-down requirements are not achievable at the bottom level and some *negotiation* is needed between end users and technology providers. This is also the case when use cases do not provide inputs for certain areas that need to be specified.

FRACTAL partners use different wordings in their respective domains. Therefore, the words *requirements, inputs, features, properties, expectations, capacities*… have similar meanings throughout the document.

4 Use case requirements

The use cases will be described later in WP7 (D7.1 due month 18) and WP8 (D8.1 due month 24). As the FRACTAL specification needs to be driven by use cases, this chapter serves as inputs for the next chapters where the FRACTAL platform is specified.

Use case	Name	Relates to
VER-UC1	Improving the quality of engineering and maintenance works through drones	WP7
VER-UC2	Improving the quality of automotive air control	WP7
VER-UC3	Smart meters for everyone	WP7
VER-UC4	Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	WP7
VAL-UC5	Increasing the safety of an autonomous train through AI techniques	WP8
VAL-UC6	Elaborate data collected using heterogeneous technologies	WP8
VAL-UC7	Autonomous robot for implementing safe movements	WP8
VAL-UC8	Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	WP8

Table 2 – List of FRACTAL use cases

After a description of each use case, their requirements are grouped according to the strategic objectives of the project:

Obj. #	Objective	Relates to
O1	Design and Implement an Open-Safe-Reliable Platform to Build Cognitive Edge Nodes of Variable Complexity	Pillar 1 WP3
O2	Guarantee extra-functional properties (dependability, security, timeliness and energy-efficiency) of FRACTAL nodes and systems built using FRACTAL nodes (i.e., FRACTAL systems).	Pillar 2 WP4
O3	Evaluate and validate the analytics approach by means of AI to help the identification of the largest set of working conditions still preserving safe and secure operational behaviours	Pillar 3 WP5
O4	To integrate fractal communication and remote management features into FRACTAL nodes	Pillar 4 WP6

Table 3 – FRACTAL objectives

Use cases are often referred with a shorter name: UC1, UC2…

4.1 VER-UC1: Improving the quality of engineering and maintenance works through drones

4.1.1 Description of the use case

Nowadays, the construction sector is one of the least digitized sectors[2]. This lack of use of technology starts in the first stages of the life of an infrastructure and continues until its final stage of operation and maintenance, resulting in an increasing in costs and in dangerous situations during the work on site. This not only means a loss of business opportunities, but it also reduces the profit margin of the projects, losing efficiency in the works on site, and making difficult to share information between different phases. Projects are becoming more and more complex and hard to manage with the current state of digitalization. In addition, construction must be oriented towards sustainability and reduction of resource use, so some traditional methods must be modified, adapted or eliminated to meet the challenges of the future. Digitalization offers new ways of working that will allow to create knowledge of the construction business with clarity and transparency.

In relation to the construction work itself, it is the least digitalized stage in the lifecycle of infrastructures. On-site analysis/consulting is necessary to improve and integrate processes related to cost, time and supplier management. In this stage, an abundant quantity of information is lost due to the large number of means of communication that are used (telephone, mail, personal, etc.), both internally and externally to the work. The constant monitoring of the work and its workers is also one of the key points to study, which helps to improve the quality of the processes, reduce costs and improve health and safety during the development of the works.

UC1 has been born following the current tendencies in digitalization of the processes through the technology evolution of the systems, enhancing efficiency and reducing costs. Within this UC1, two end-to-end solutions will be developed and tested, which will allow improving economic and operation efficiency and safety conditions in the construction of civil engineering works. The UC1 aims to enhance the management environment by treating the collected data on site, through the use of IoT platforms and the sensorization of the construction areas. Given this context, the UC1 has been focused on two main areas: (1) the maintenance of critical structures to increase the useful life of structures, and (2) the digitalization of the workforce equipment on construction sites; both areas aiming to improve safety and operation conditions in construction works.

From these two areas, two demonstrators will be gathered inside the UC1:

Demonstrator 1 - UAV supervision of critical structures

[2] McKinsey, Imagining construction's digital future, June 24, 2016, https://www.mckinsey.com/business-functions/operations/our-insights/imagining-constructions-digital-future

Demonstrator 1 will be focused in the supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface. Within this demonstrator, PROINTEC will develop an algorithm capable to distinguish between active and non-active cracks of a wide range of pathologies registered in concrete structures. These measurements of the structural status will allow an early detection and classification of the cracks, and which is even more important, a comparison at different times of the evolution of the crack.

The information collected from the UAVs will be processed in the fractal node, and applying the artificial intelligence, the images will be treated in order to create a map of the structural status of the detected cracks.

Figure 2 – UC1 demonstrator 1 scheme

Demonstrator 2 - Wireless Sensor Network (WSN) for safety at construction sites

Demonstrator 2 will be focused in the monitoring of both workforce and machinery within a construction area, by deploying a WSN that provide information about the status and location of the workers in real time. This information will be managed through an IoT platform, registering possible dangers and alarms, apart from establishing a protocol in case of emergency. With this solution, the risk of accidents involving machinery and workers will be reduced, improving traditional health and safety systems, focusing the action in the vision of zero injuries at construction sites.

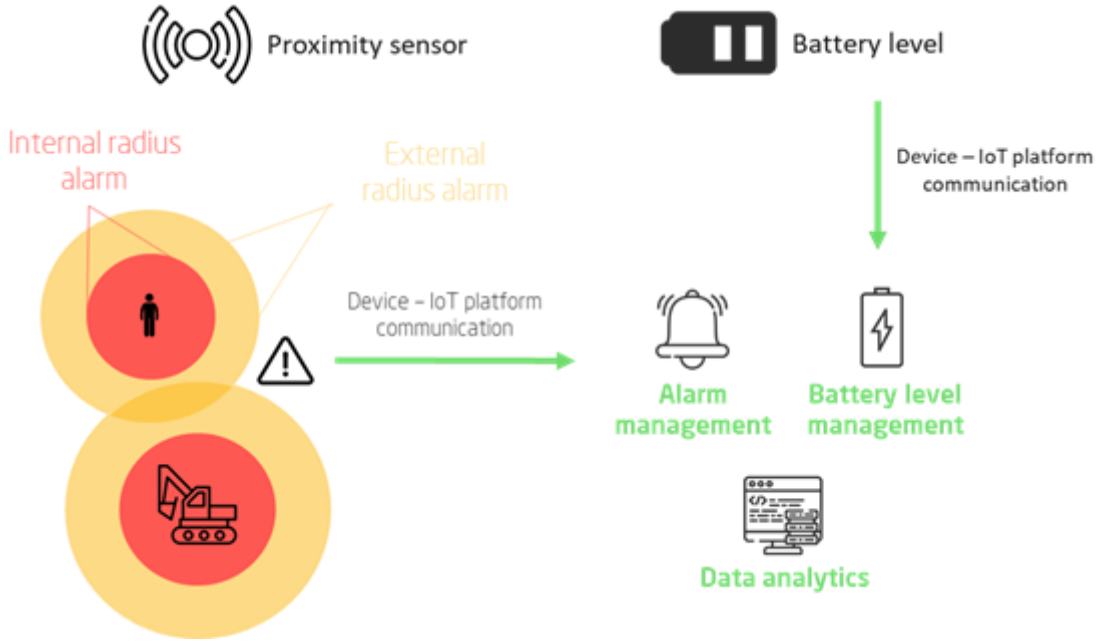

Figure 3 – UC1 demonstrator 2 relations description

In this case, workers will carry personal sensors in their work equipment that will record their position, and several alarms, both to workers and platform, when a vehicle comes too close and vice versa. All this raw information will be collected, processed in the Fractal node and represented in an IoT platform through a user-friendly dashboard. A report will be generated that allows to take measures and reschedule the works on site.

Figure 4 – UC1 demonstrator 2 scheme

4.1.2 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP7 at the end of the project. Further elaborations will occur in WP7.

The present use case can be divided into 4 large consecutive areas of action, which will lead to the achievement of the objectives of the UC1 and obtaining a technological solution for the demands of digitalization of tasks in construction works.

1. Collection of data

Firstly, different data acquisition campaigns will be carried out on the construction sites where the demonstrators are to be located. These campaigns will be carried out

in two ways. On the one hand, for demonstrator 1, images will be collected with UAV drones of a critical structure, such as a viaduct or bridge, so that information can be obtained on the most sensitive areas of cracking due to the loads to which the structure is subjected. On the other hand, for demonstrator 2, once the sensors have been deployed on both the operators and the machinery, a specific time of use will be established in order to find a pattern of incidents and obtain a sufficiently large sample to be able to take corrective measures.

2. Treatment of data

The information collected in the above-mentioned campaigns will be processed in both demonstrators through the Fractal node. This processing will be based on artificial intelligence and edge computing techniques, which will simplify and allow the interpretation of the information in a simple and clear way. In this sense, within the demonstrator 1, a previous study of pathology recognition technologies in structures through artificial intelligence will be carried out, in order to apply the most precise processing techniques. In demonstrator 2, an information filter will be developed through the node, which will allow the selection of key and representative information from the collected data.

3. Visualization of the work status

The results will be presented on different platforms, through dashboards and multi-screen systems in the cloud. In this way, the information will be able to be consulted in real time, both on site and remotely.

4. Adoption of measures

Once the results have been obtained, the necessary measures will be taken to improve safety and efficiency on the construction sites. In this case, within demonstrator 1, a representation of the evolution of the pathologies detected in the structures will be established so as to indicate the need or not to carry out maintenance on the structure. Within demonstrator 2, the need to rethink the work on the construction site will be analyzed, as well as the skills of the operators and their qualifications for carrying out the work.

4.1.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • **Authentication and authorization capabilities**. • **Computing needs**, including required memory, data bus speed and width, processor speed, and potential need for GPU/TPU usage.
Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5. • **Suitable input**, including image resolution, image quality (stability, noise and focus), stable environmental/acquisition conditions (lighting, dynamic range, angle and distance) • **Suitable a priori knowledge**, meaning a proper and large enough dataset of cracks. It should include all the different pathologies and acquisition conditions to be found in the actual operation. • **SW stack** which shall run Tensorflow/Keras/TensorRT/TFLite.
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. • **Communication capabilities** with different data sources, directly or indirectly (WS, API, REST, JBDC, OBDC). • **Communication latency < 2s.** • **Direct communication with the Fractal node via API / WS** with individual device identification.
Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements are inputs for WP3. Target: PULP and VERSAL platforms • **Required processing time** from data acquisition/downloading to generation of the diagnosis. • **Customizability of the fractal node** in order to be useful for a vast set of different IoT-applications and data sources to be used in UC1. • **SW stack**, which potentially includes a stable distribution of Linux compatible. The SO and libraries shall integrate the GPU/TPU mounted.

Other requirements

This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP7.
• **Multi-platform web** frontend based
• **Multi-user display** platform and multi-screen on site (tablet, laptop)
• Data representation in real time through **Dashboard**:
  o Positioning of operators and machinery.
  o Alerts (day and time, internal/external radio, operator, machinery).
  o Battery level of the devices.
  o Alert Concentration Heat Maps.
  o Weather conditions.
• Analytical and operational capacities and **data visualisation and reporting.**
• **Statistical analysis of the captured data** for "in situ" and "posteriori" measurements.
• **Availability of images of the cracks in the structure at different times**, to check the evolution.
• **Definition of the metrics and criteria for evaluating the detection**, including the criteria of coverage (i.e., it is necessary to detect the crack completely, or with a partial detection is sufficient), accuracy (i.e., how many of the diagnosed cracks are actually cracks) and recall (i.e. how many of the total cracks have been correctly diagnosed).
• **Definition of cracks** to be located, including their shape, size, type and variability.

Table 4 – VER-UC1 requirements

4.2 VER-UC2: Improving the quality of automotive air control

4.2.1 Description of the use case

Existing automotive **air-path control strategies** are fully **reliant on model-based control techniques**. The air path of an engine is a highly nonlinear system with dead zones, hysteresis and delays. Typically, the behavior of the system is modeled using first principles that allow physical interpretation, but it is hard to exactly capture the dynamics of the environment while they cannot be incorporated into the controller. The ability of air-path control strategies to perform self-learning through observations of specific situations is therefore minimal. The usage of these systems is **predominantly localized (at the vehicle level)**, but their development might benefit from swarm behavior. Consequently, there is limited scope for sharing of localized learnings. This use case will, therefore, contribute to integrate complex environmental knowledge as a fundamental part of the system, among other benefits, like potentially increased product quality and increased efficiency for the development of customized air-path controllers.

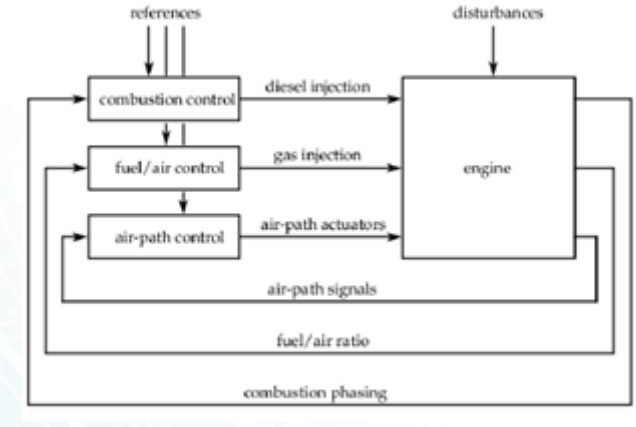

Figure 5 – Engine control diagram

The resulting algorithms are generalized and do not lend themselves to the adaptation to specific variations (e.g. uncertainties due to manufacturing tolerances or aging etc.)

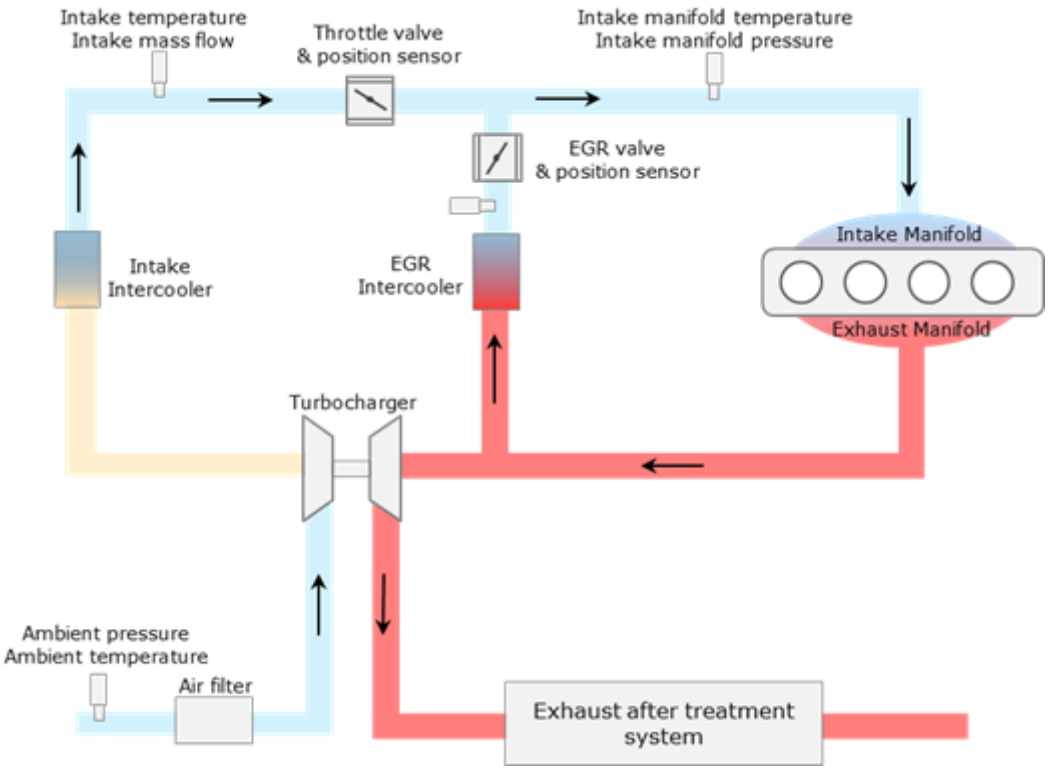

Figure 6 – Physical Air Path diagram

In Figure 6 the physical layout of the Air Path on a combustion engine with an Exhaust Gas Recirculation (EGR) can be seen. The main actuators of the Air Path are the EGR valve, the throttle valve and the turbocharger. The main control variables of the Air Path are the intake manifold boost pressure, the lambda, which is the air to fuel ratio and the NOx emissions of the engine. This is why, the **air-path is a multifaceted system** consisting of components of various complexity levels. Together, they are responsible for mission-critical functionality. Their operation has the potential to benefit from **proactive self-learning**. Such ability would help correct own component and system shortcomings. The localized learnings have a potential for **exploitation further than at a single vehicle level**, which is crucial for commercial exploitation that is possible through **scalability**. The use case aims to demonstrate:

- Implementation of data-driven models aimed at improved energy efficiency and reduction of environmental pollutants
- AI-based self-learning and self-adaptation of the data-driven models, as well as consequential enactment of control strategies
- Identification of potential cyber-security breaches through anomaly detection

The concept of data-driven model application to automotive air path control has the potential to provide **higher precision in control and diagnosis**. The resulting **dependability enhancements** are possible through **simplified development with significantly lower computational effort** over existing controllers. The

ability to self-learn and apply adequate improvements to the control system enable an appropriate response to changes in the environment of CPSoS. The quality of learning and predictions must be raised beyond the current level so that some aspects of the Learning Engine replace the existing model-based control. The collective learnings should improve **resilience and energy-awareness**.

4.2.1.1 Applications in the Area of Predictive Diagnostics

Motivation

This UC investigates analytical solutions that support predictive maintenance of physical entities in powertrains. To develop such predictive capabilities, we primarily focus on using in-use data (in most cases in vehicle-data) of well-known mechatronic powertrain components for the task of modeling. We thereby establish a link between the in-use behavior and the ageing of these components to enable proactive actions to prevent severe malfunctions or damage.

Current Situation

Regular maintenance of products (e.g., components of the powertrain, test beds to verify and validate powertrains) aims to keep availability and reliability of these products as high as necessary. When unexpected outages occur, this causes unforeseen expenses. Preventive maintenance aims to address this on a time- or performance-based cycle. With predictive maintenance, AVL's PTE[3] business unit strives to anticipate and avert such outages. The aim is to detect possible and imminent breakdowns at the right time to help prevent these events by planned maintenance activities.

Challenges

The process of setting up predictive maintenance starts with identifying an upcoming issue that will cause a breakdown in the near future. To begin, reasonable variables must be determined that allow the target state of the product (or parts of it) to be defined. Monitoring the predefined variable means that the current status of the product has to be measured at regular points in time — for example, pressure, temperature or vibration have to be detected to describe and document the current status. Thereby the frequency of measurements also determines the quality of the collected data. It is also necessary to define how the data will be used for prediction, so a model must be developed to monitor the product's condition, with algorithms to analyze trends and better forecast product status and possible breakdowns.

In this UC, we are focusing on analytical solutions allowing predictions about future incidents. In particular we are ***using in-use data of physical products*** **within a powertrain** (in-vehicle data capturing the ***behavior of components such as turbochargers, oil pipes, valves or coolers***) or **in-use data from testbeds for powertrains** (endurance runs, such as ***contactors and fuses*** *used in* ***testbenches*** *and* ***batteries***). In both cases, we consider ***in-use data that in combination with***

[3] AVL PTE: AVL powertrain engineering

the predictive model will allow for ***accurate prediction of future incidents***. In this respect, the following challenges need to be addressed:

- ***Accessing sensor data:*** Gathering sensor data from the product.
- ***Preprocess data****:* Cleaning data by removing outliers aligning time series or filtering out noise.
- ***Feature selection:*** Capturing abstract features that are suitable to be fed into the model.
- ***Modeling:*** Development of models that classify the product or parts of it as healthy or erroneous and can detect anomalies or predict the remaining lifetime using actual in-use data.
- ***Deployment of the model:*** Support the deployment of the models into the final service / product.

Objective

By mastering the above-mentioned challenges, AVL PTE business unit will be able to anticipate problems before they might actually occur. A computational node of the product in consideration is comprised of the entire data from concept to the operation of the product. Besides of establishing connectivity, the computational node provides interfaces for a well-defined data structure that enables data ingest, data storage (e.g. meta-data, data storage in the case of connectivity issues, etc.) and management (e.g. processing, plausibility check, etc.) and supports the dedicated workflow to execute a predictive maintenance model (e.g. model inference).

Figure 7 – Predictive maintenance models link the in-use phase to the development and the workshop/maintenance phase

Developing predictive models is a challenging task, which includes four major steps:

1. Data Preparation
2. Descriptive Statistics/Analytics
3. Feature Engineering
4. Modeling and Evaluation

We envision the development of various models within this UC. Each model makes use on in-use data (e.g. in-vehicle data for powertrain components). In the following we briefly characterize one potential envisioned defect related to the Air-path.

Exhaust-gas recirculation valve: Stuck valve is caused by deposition of unburned hydrocarbons and soot. The erroneous behavior depends on mounting conditions and software configuration.

Observable symptoms:

- Exchange of valve
- Certain entries in the defect code memory

- No electrical fault and no stem fault is appearing

Technical service action: Exchange of the valve and/or software adaption.

4.2.1.2 Proactive Adaptations (OTA)

The air-path service repository, which deals with the mapping of the learning to the situations, also has a prospect of eventually becoming **fog-based**, so that future work could also benefit from the use case (e.g. exploitation of information on driving patterns or **information from the traffic infrastructure**). This hierarchical composable structure lets itself for further expansion of the situation cognition and proactive adaptation at the air path level to be combined with the air quality monitoring and **traffic management systems**. In striving to balance a trade-off between **fog-based** and **edge-based computation**, this use case also investigates crucial issues around massive computational power inside the system.

4.2.2 Roadmap to achieve use case KPI and objectives

To achieve the aforementioned objectives of this use case, the following steps have to be applied.

First the Air Path Control Strategy has to be adapted, in order to be able to communicate with the data-driven models (see Figure 8). Therefore, different sensor and actuator values have to be pre-processed and additional inputs have to be calculated within the ECU[4].

[4] ECU – Engine Control Unit

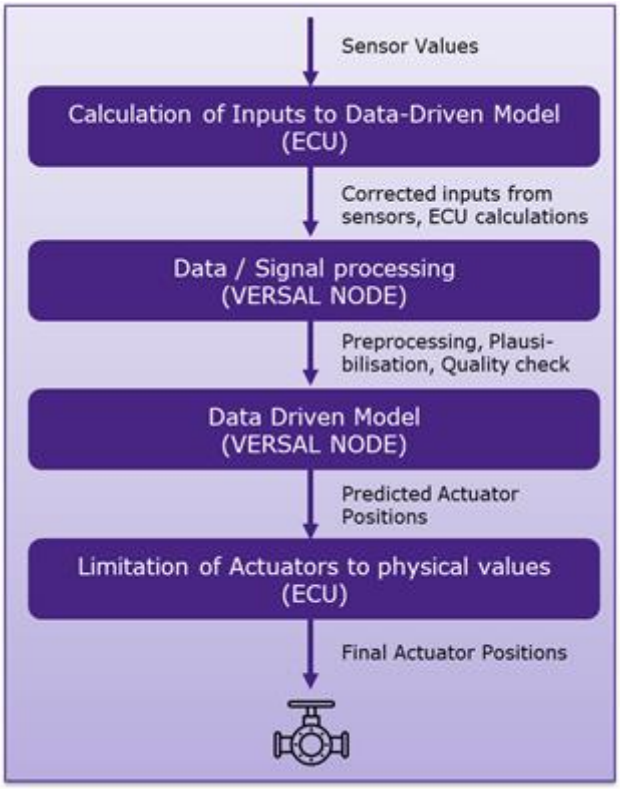

Figure 8 – Overview of UC2

Then a suitable training environment will be established with measurement data from a real engine and the data-driven model will be developed. Different modeling algorithms/techniques for the data-driven model development will be considered and the most suitable will be used for further investigations.

After the model is developed and tested in simulation, the model will be integrated on the FRACTAL node and evaluated on the board. Especially the safe (e.g. correct transfer) and steady (e.g. update rate) communication between the node and the Engine Control Unit will be crucial.

After the verification of a safe control of the actuators, different adaptation and diagnosis concepts will be tested.

4.2.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • The Node shall have the capacity to process time-series input-data in 10Hz update rate
Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5. • The Node shall be able to execute TensorFlow framework models
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. • The Node shall be capable of communicating with cloud services for federated learning as well as diagnosis and adaptation • The Node shall be capable of receiving over the air updates (OTA) for adaptation of the models after the development phase • The Node shall have CAN communication ports, to be able to communicate with the Engine Control Unit and other nodes within the vehicle • The Node shall be capable of communicating via Ethernet, to be able to communicate with additional sensors (e.g. information from the multimedia systems, etc.) and other nodes within the vehicle
Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements are inputs for WP3. Target: commercial node (VERSAL) • The Node shall have a non-volatile memory, to store information also after the engine is switched off (e.g. storage of adaption values, ageing parameters, etc.) • The Node shall have a storage media (e.g. solid state disc, ~200MB) for metadata or offline communication (e.g. driving in a tunnel) • The inputs to the node will be very transient, therefore, the node has to be able to compute new actuator positions every 10 ms in real time • Good performant math libraries shall be available (e.g. possibility to apply filters, or perform simple aggregations like moving-averages, etc.) • The Node shall have the possibility of parallel processing (e.g. at least 4 cores) • The Node shall have at least 16GB RAM

• The Node shall have an uninterruptible power supply • The Node shall have an internal voltage transformer • The Node shall have Linux OS • The Node shall have a C++ compiler
Other requirements
This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP7. To be able to operate the Node in the engine compartment, several different **physical requirements** need to be fulfilled to ensure a safe behavior of the engine control system: • The Node shall be robust to operate at temperatures up to 100°C • The Node shall have a housing that protects it against dust and spray water • The Node shall be robust against vibration

Table 5 – VER-UC2 requirements

4.3 VER-UC3: Smart meters for everyone

4.3.1 Description of the use case

Smart metering is a hot topic and one of the top use cases for the internet of things. The goal is to read the meters remotely by connecting them to the internet. This allows utility providers to remotely read the meters with the benefit that they would no longer need to visit customers to physically read the meters. In order to support smart metering, the meters and its infrastructure around need to be electrified which is often not the case. Especially legacy utility meters such as gas, and water meters often work with pure mechanical principles. Such meters lack power supply and an electronic interface for accessing the meter stand. Electrifying the infrastructure and replacing these meters with a smart device that is connected to the internet is a big investment.

A low-cost non-invasive alternative would be to put a battery-operated device equipped with a camera to take a picture of the meter and run a pattern recognition algorithm directly on the device to identify the meter stand. The extracted values can then be transmitted wirelessly over the cellular network. Such a device must have a small form factor in the range of a 3-5 cm^2 such that it can simply be tagged on a meter. It should consume as little power as possible such that it can be in the field for multiple years. Further, it needs to efficiently and reliably read the meter stand in suboptimal lighting conditions. And finally, it must be capable of transmitting the data over a wireless channel, even if the device is in a location with limited connectivity such as a basement.

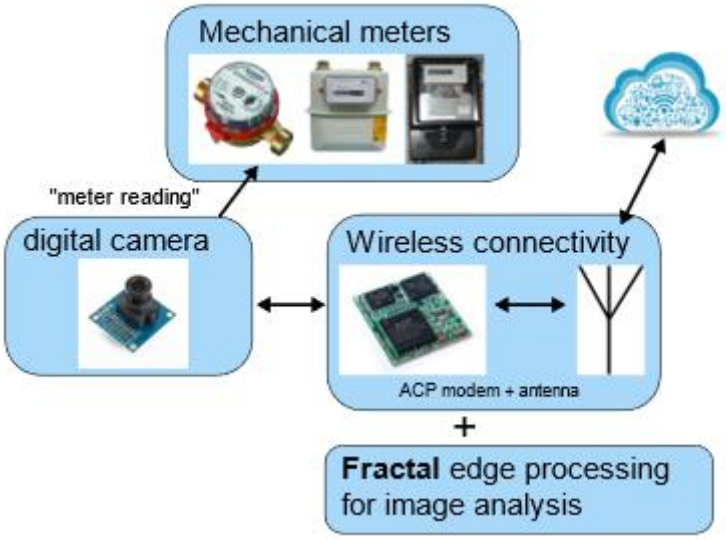

Figure 9 – Smart meter diagram

As a final product, we envision a single chip solution that consists of a programmable platform that can extract the meter stand and run a protocol stack for wireless connectivity on the same chip. For this purpose, a powerful, but energy-efficient compute platform is required. To achieve this final product, a prototype based on the

fractal platform will be developed. The following chapters will highlight how we want to achieve this goal and provide detailed platform requirements.

4.3.2 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP7 at the end of the project. Further elaborations will occur in WP7.

The project can be split in three main parts:

1. Image analysis:

The fractal RISC-V platform will be interfaced with a low power camera that can take pictures of the meter. In a second step, the platform must analyze the picture and extract the meter stand. The main challenges of this task will be to reliably detect digits in an image with a pattern recognition algorithm, on a platform with only a few 100kB of memory and in a power envelope of a few milliwatts such that the device can remain active for multiple years.

2. Wireless connectivity

Transmitting the full image over the internet will take too much time for IoT protocols, and therefore consume too much power. With clever duty cycling, and by only transmitting the relevant information (the meter stand) the power consumption will be greatly reduced. In addition, it will be a challenge to reliably establish a wireless connection in locations with limited connectivity such as basements where meters are typically located.

3. Security features

User data must be stored encrypted on the fractal node to protect personal, sensitive data from external access. The platform must be authenticated by the utility provider, such that it cannot be cloned. Further, the platform must be able to verify its firmware, before reading and transmitting meter readings, such that it cannot be manipulated.

In a final product, these features must be implemented in a single chip, otherwise the solution will not be affordable in prize, and will be too big to be tagged on a meter. For this project however, we can build a prototype of the system based on a FPGA platform with the fractal node and interface it with a camera, and ACPs modem that offers connectivity over the cellular network. Even though a big FPGA is being used in this project, it is important to keep the power consumption and form factor requirements of the final solution in mind during the development of the fractal node.

The prototype that is developed in this use case is targeting smart metering. The prototype's functionality is however not limited to smart metering. A small batter-operated device that is capable of analyzing images, perform edge-computing and establishing a wireless connectivity to communicate with a server is potentially useful

in many scenarios for the internet-of-things. Through the fractal project it will be possible to connect multiple such nodes that are used in different scenarios in one network.

4.3.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

The following table summarizes the specific requirements for the fractal node that is used by the smart meter prototype.

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • **Low power consumption:** VER-UC3 requires smart scheduling techniques to prolong the battery lifetime of the system. Therefore, active times must be kept as short as possible and the system must go in deep power down state when it is not used. • **Encryption services to encrypt user data:** Encryption must be completed within milliseconds, and with a memory footprint of a couple of 100kB due to the limited availability of memory in an integrated circuit. • **Authentication and Integrity:** It is important to guarantee authenticity, confidentiality and integrity of the transmitted data. In addition, VER-UC3 would benefit of an authentication service that can run on the fractal node to authenticate meter reading requests from the operators.
Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5. No specific requirement for autonomous operation and AI
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. • **Communication framework with minimal overhead:** VER-UC3 will transmit and receive only a few hundred bytes per day (meter stand, time, date, unique identifier). For such IoT applications it is essential

that the overhead of the communication protocol is minimal and thus not becoming a main contributor to the power consumption of the device. MQTT could be a suitable protocol.

- **Bandwidth and latency requirements:**

  The smart meter prototype will periodically establish a wireless internet connection over the cellular network (NB-IoT) and transmit its data. Since only few bytes need to be transmitted per day, a bandwidth of 1kbps is sufficient. Latency must not exceed 1s.

Objective 1:

Open-Safe-Reliable and low power node architecture

These requirements are inputs for WP3.

Target: customizable node (PULP)

- **Processing performance:**

  In VER-UC3 it is foreseen to run a convolutional neural network (CNN) on the fractal node to identify digits in an image. To enable a long battery lifetime, it is important that the CNN can be computed in a short time (max. few seconds), such that the system can go back to deep sleep as soon as possible. To be able to reach this goal, a processing performance in the range of 10 MOP/s to 1 GOP/s will be necessary.

- **On chip memory requirements:**

  In order to keep the form factor of the device small (when ported to an integrated circuit), it is not possible to equip the chip with multiple of megabytes of memory. We estimate that 512kB of on chip memory and possibly a 1-2 MB of off chip memory (with higher latency) will be required.

- **Non-volatile storage:**

  The software plus the weights of the CNN must be stored in a non-volatile memory, typically a flash. We estimate that 3-4MB of flash memory will be required to store the weights. In addition, a firmware with a small program to interface the camera, run the CNN and establish a connection with the modem must be stored in the flash. This program should also include encryption, and authentication services and will therefore also require 1-2MB of memory.

- **Small form factor:**

  The goal is to tag the final solution of the smart meter prototype directly on the meter. Hence, the size of the fractal node, together with a battery, a camera and an antenna, must be in the range of a few cm^2 when integrated in an advanced technology. It is therefore important that nothing in the fractal node will prevent the production of such an integrated circuit.

- **Real time clock:**

  The smart meter device of VER-UC3 will be configurable with a wakeup period in which it will autonomously wake up, take a picture and analyze it. The meter stand can then either be directly transmitted to the operator, or it can first be stored in memory together with a timestamp, and once enough readings have been collected, all of them can be transmitted to the operator.

- **Interfaces:**

  VER-UC3 will require only a limited set of interfaces such as UART and a camera interface. The platform should however support other common interfaces like SPI, I2C, I2S, USB, etc. to interface other sensors, or transfer data from one node to another.

- **Low power consumption:**

  The smart meter of VER-UC3 will be in a deep sleep state most of the time. To enable a long battery lifetime, the power consumption of this state is of most importance. Deep sleep currents of <10uW will allow the system to be active for 10+ years when used with a medium sized battery (2000mAh). The fractal node must be portable to an integrated circuit in which the power consumption of the deep sleep state is in this order.

- **Customizable:**

  The fractal node must be customizable in order to be useful for a vast set of different IoT-applications. Some applications require more memory, others more processing power.

- **Software stack:**

  Given the limited amount of memory and the power envelope of the smart meter prototype, it is not feasible to use Linux as an operating system. Running a bare-metal application that reads the meter, analyzes the image and establishes a connection with the modem is sufficient. Using a simple operating system such as littleKernel (lk), Zircon or FreeRTOS is preferable.

- **Real-time support:**

  Image analysis and meter reading are not real-time critical. Hence, no support for real-time execution is required. It could however be that a real-time critical application is running on the same chip (such as the software stack of the modem). It is a plus if the fractal node can run a real-time OS.

Other requirements

This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP7.

- **Reliable wireless connectivity**:

Meters are often placed in basements where there is only limited wireless connectivity. The smart meter prototype requires a modem that offers extended coverage wireless connectivity such as NB-IoT over the cellular network.

Table 6 – VER-UC3 requirements

4.4 VER-UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications

4.4.1 Description of the use case

Real-time object recognition has shown to be an important segment for many industrial applications where automation replaces manual work. This feature enhances the automation process with intelligent capability to detect and recognize objects visually. The whole process is based on machine learning approaches where the inference of a previously trained convolutional neural network (CNN) is used as an algorithm for detection and recognition of the objects from the input data.

The computation capacity nowadays provided by edge computing made it possible to run the object detection and recognition algorithms closer to the location where the object is observed, and with that to eliminate the needs for sending the input video data to the cloud services for data processing. The proximity to the source brings few crucial benefits for this type of solutions. First, it eliminates the need for remote computation in the cloud and with that the need for wide bandwidth, second, it lowers the responds time that would have been imposed due to the network communication delays, and third, it increases the privacy by keeping the video data local. Running the inference locally on the node also enables the edge computing device to perform the process of detection and recognition in real time.

VER-UC4 has for goal to implement a vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block as part of the FRACTAL edge platform. The proposed LLOD building block will have the ability to detect the objects, localize their positions in the image, and categorize them based on pre-defined classification. Figure 10 shows the main component of this use case as well as the flow of the data. The LLOD building block takes as an input a video stream generated from the camera. The stream is handed to a device that runs algorithm for computer vision on top of it. Once the frame processing is finished the device publishes the results on the display. The output is localization of the objects in the image and their classification based on the group that they belong. All this will be performed in real-time as the input video stream flows. The detection, localization and recognition of the objects will be done with the help of inference of a previously trained convolutional neural network model called YOLO, which is described below. The LLOD building block will be implemented in form of a hardware accelerator as part of a larger SoC deployed on FPGA hardware platform. The flexibility of FPGA allows the LLOD building block to be configurable and adaptable for different neural network algorithms. Thus, any change in the inference will simply require reconfiguration of the accelerator in order to improve execution speed and reduce energy consumption.

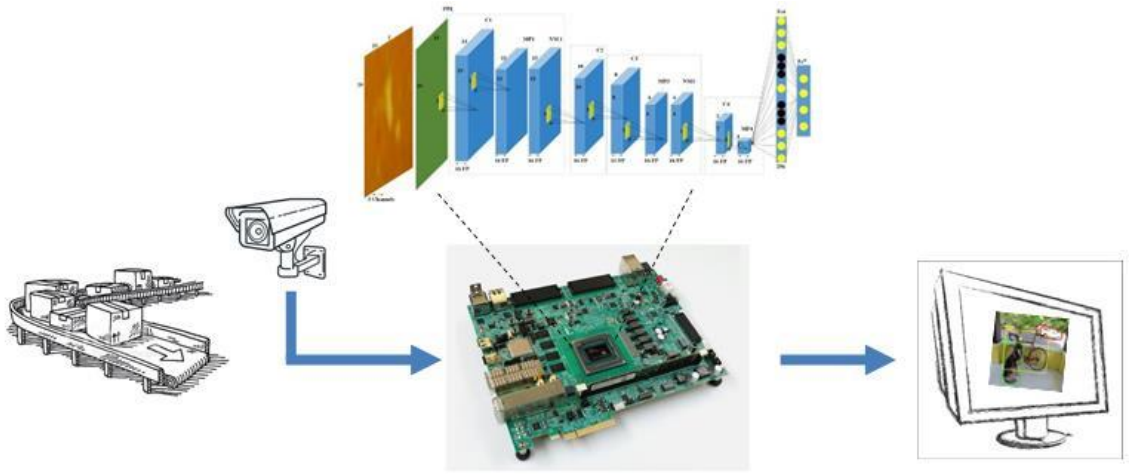

Figure 10 – VER-UC4 Object detection and recognition in industry

VER-UC4 will focus on exploiting the performance and behavior of the new proposed hardware extensions developed in WP3. The inference will be run on proposed hardware and the outcomes will be evaluated. This will allow us to have a better understanding on the impact that proposed hardware extension can have on the execution performance of the inference for visual computation.

Once the prototype is ready it will be handed over to use case VAL-UC8 to be integrated as part of the SPIDER autonomous robot.

4.4.1.1 Object detection and recognition algorithm - YOLO

Object detection and object recognition are techniques used for detecting and identifying objects within an image or a video. Object detection has for goal to localize the objects in the image, while object recognition understands the content of the image and identifies the objects on it.

You-Only-Look-Once[5] (YOLO) is a real-time object detection and recognition algorithm that detects and categorizes objects in extremely short time. It is based on a convolutional neural network that can predict multiple objects and their position on the single image simultaneously. Unlike classifier-based methods, YOLO evaluates the image only once. This is achieved by unifying all components required for object detection into a single neural network. The entire image is inspected at once and all the bounding boxes are predicted instantly. Such an approach makes it possible for the neural network to perform end-to-end training in real-time speed.

The algorithm divides the image into a *SxS* grid, where each grid cell predicts *B* bounding boxes and the confidence score for those boxes. Each bounding box consists of (*x*,*y*) coordinates of the box center, the width *w* and height *h* relative to the whole image, and the confidence which presents the intersection over union (IOU) between a predicted box and any ground truth box. If the bounding box is spread over multiple grid cells, then the grid cell containing the center of the object has the

[5] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection".

responsibility to detect the object. The bounding box uses multilabel classification to predict the classes.

The model is implemented as a set of 24 convolutional layers followed by two fully connected layers. The convolutional layers extract features from the image while the fully connected layers predict the predictability and the coordinates.

The light variation of the YOLO algorithm with less convolutional layers and fewer filters is called Tiny-YOLO. This solution consists of 15 layers on which kernels of size 3x3 and 1x1 are used for convolutional layers and kernel of size 2x2 for pooling layers. The inference has a smaller size (less than 50MB), is a few times faster than the main version and achieves a higher rate of frame processing.

The reason for choosing Tiny-YOLO as inference for edge computing in UC4 are:

- It detects and identifies the objects in the images very fast,
- Has a high rate of processing frames per seconds,
- Has a small size that makes it suitable for embedded devices,
- Achieves a high accuracy on object identification,
- It analyses the whole image at once,
- Has a low rate of background errors compared to other approaches,
- It is a mature solution for object detection and recognition,
- And it is an open source trained neural network.

Apart from these advantages, YOLO has limitation as well. From each grid cell in the image the model can only predict two boxes and can only have one class per cell. The model also struggles with objects that are small and appear in group. These constraints limit the number of objects that can be predicted within a cell.

4.4.1.2 The hardware platforms

The LLOD building block will be implemented on the Xilinx VERSAL Adaptive Compute Acceleration Platform (ACAP) and the Parallel Ultra Low Power (PULP) platform. Such an approach will give us a better understanding on the impact that different hardware designs developed in WP3 can have on the behavior and performance of the neural network inference for computer vision. Xilinx ACAP is described in section 8.1, while the PULP platform is described in section 8.2.

The LLOD prototype to be developed for this use case consists of a camera, the software/hardware platform from the FRACTAL edge node and a display. The camera points to the production line and is used for generation of the input video streams. The frames from the video stream are processed from convolutional neural network that runs on one of the defined hardware platforms. The output results on detection and recognition of the objects are shown on the display.

In order to observe the impact of different hardware architecture on the execution of the inference we propose five solutions with diverse hardware architecture:

- The first solution will run the neural network inference on scalar processor without utilizing any hardware acceleration. For Xilinx ACAP platform this will be an ARM processor, while for the PULP platform a RISC-V processor. This

will demonstrate the difference between ARM and RISC-V processors when they run inference of neural network.

- The second solution will utilize the AI engine provided from Xilinx platform. The neural network inference will be mapped on the AI engine to exploit the benefits that could be gained by using the array of AI engine tiles. This case will demonstrate how the AI engine deals with data-level parallelism and what impact the AI engine has on the performances of the neural network. The whole demonstrator will be built using the SDK provided by Xilinx for ACAP platform.
- The third solution is based on a High-Level Synthesis (HLS) hardware accelerator, which consists on a configurable array of processing elements. The accelerator will be implemented in the programmable logic part of the board for both Xilinx ACAP and PULP platform. In contrast to the second solution where the accelerator is a hardcore solution, this one is an array of processors that is flexible, configurable, and adaptable. This case will demonstrate the impact of adaptability of accelerator on the performance of neural network. The outcome results will also be compared with Xilinx ACAP solutions that uses the AI engine.
- The fourth solution will be implemented on heterogeneous embedded system on chip (HESoC) consisted of a standard scalable processor as host and a cluster of programmable many core accelerators (PMCA). The platform is called HERO[6] and is part of the PULP solutions. The cores of the accelerators are RISC-V processing elements which are adapted to run the neural network efficiently. The software stack provided by HERO will be used for building the demonstrator and the generated results will be evaluated.
- The fifth solution will be a solution that utilizes a RISC-V processor that supports Instruction Set Architecture (ISA) extension for data-level parallelism. The advantage of RISC-V processor is that it has a modular ISA, thus adding or removing a set of instructions belonging to one module will not affect the other ISA modules. Based on the specification of RISC-V[7] there are two types of modules to deal with this form of parallelism called "P" and "V" extensions. "P" standard extension is a packet with SIMD instructions, while "V" extension covers the instructions for vector operations. "P" extension covers the SIMD instructions only for integer operations, while the float-point SIMD operations are dropped in favor of standardizing the "V" extension. Also, the size of the vector on which SIMD instructions can operate is limited. The "V" extension offers more flexibility since the number of registers that it uses to define the size of the vectors is not fixed and also the type of the elements within the vector. All parts of the code where data-level parallelism comes into expression will be executed on the vectoral part of the processor while

[6] HERO: Heterogeneous Embedded Research Platform for Exploring RISC-V Manycore Accelerator on FPGA, Anreas Kurth et al., CARRV 2017

[7] The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA, 20191213

the rest of the code will be mapped on the scalar part. Since inference of a neural network has properties of data-level parallelism it will by transformed from the compiler on a set of vectorial instructions. Extension of RISC-V processor with "V" or "P" instructions has not been considered as part of this project, but a possible implementation would be a good case to exploit the behavior of the neural networks on such architectural solution.

All the development on Xilinx platform will be done with the help of the SDK provided from the vendor. For PULP platform we will use the 64bit version of the RISC-V processor that can run Linux operating system.

4.4.1.3 Applications of LLOD

LLOD as a solution can be used on different industrial applications. In the following we list examples of industrial domains where LLOD can be applied.

- Quality control process – can be a visual task performed to detect faulty products on production lines. The LLOD can be part of an inspection process to detect faulty products and to trigger the next action when such a product occurs (e.g. separate).
- Inventory management – is a complex task in an industrial environment where the value of inventory is changed dynamically because things are added or removed frequently. With help of LLOD the process of keeping track of inventory can be automated.
- Robotics – is extremely useful in the process of automation. LLOD empowers the robots to correctly locate and differentiate the products in the production line.
- Safety – is an important part of human-machine interaction process. LLOD increases the safety of such a process by observing human-machine interactions and triggering safety mechanisms in safety critical situations.

4.4.2 Roadmap to achieve use case KPI and objectives

To demonstrate the behavior of the LLOD building block and to evaluate its performance we will implement the model on both proposed platforms. The goal for this use case is to check for all proposed solutions:

- if the LLOD model runs properly on the proposed solution,
- if the solutions fulfill the objectives, and
- to compare the behavior of the inference on all of them.

The solutions where the inference will run directly on RISC-V and ARM processor will be the first ones to be implemented. The results from such an approach will present a good basement to compare the performance improvements that can be gained with later solutions when special hardware extensions are used.

The second step will be the mapping of the neural network inference on the AI engine of Xilinx platform with the help of the provided SDK. Once we confirm that the implementation runs properly, we will evaluate the behavior of the inference and compare it to the baseline solutions from the previous example.

Next, the inference will be mapped on the PMCA part of the HERO platform with all other operation on the host processor. The results will be evaluated and the behavior of the neural network inference for this case will be compared with the other cases in order to extract the advantages and disadvantages between different hardware solutions.

The last step is the evaluation of the hardware accelerator implemented on the programmable logic part of the platform. The solution will be evaluated for different configuration parameters due to its flexibility to observe the impact of the configuration of parameters on performance of the neural network inference. The results will be compared with the other solutions as well.

4.4.3 Requirements

The following tables summarize the high-level requirements for UC4. These requirements are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • The LLOD controlled device shell response in latency shorter than 1ms. • The LLOD object detection latency shall be shorter than 50 ms. • The LLOD shall have capacity to process at least 30 fps. • The LLOD shall be energy efficient by operating within the limited power envelope of the edge node.
Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5. • The LLOD shall be able to locate the objects in input video stream. • The LLOD shall be able to recognize the objects that have already been detected in the video stream. • The LLOD shall perform detection and recognition operations of all objects in video stream through a single observation. • TensorFlow/Caffe/Darknet frameworks shall be supported by the edge node for generation of the inference. • The inference of LLOD shall be able to process the video input in real-time. • The operation of the inference of LLOD shall be isolated within the edge node.
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. • The edge node shall provide Ethernet interface for remote monitoring of LLOD.

• The edge node shall provide Ethernet interface for remote configuration of LLOD. • The edge node shall provide Ethernet interface for communication of LLOD with other FRACTAL nodes.
Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements are inputs for WP3. Target: PULP and XILINX platforms • The inference of LLOD shall be run on platform consisted of host CPU and hardware accelerator. • The hardware accelerator of LLOD shall be powerful enough to run the inference without any stall as the video streams flows. • The hardware accelerator of LLOD shall be flexible by enabling re-configuration of the hardware for different inference models. • The RISC-V processor shall be 64-bits. • The RISC-V processor shall support M extension for multiplication. • The RISC-V processor should support "V" extension for data-level parallelism. • The compiler should support data-level parallelism for "V" extension on RISC-V. • The RISC-V processor should support "P" extension for data-level parallelism. • The compiler should support data-level parallelism for "P" extension on RISC-V.
Other requirements
This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP7. • The LLOD shall be able to run applications on top of the software system that will control its operation. • The RISC-V processor shall run 64-bit Linux operating system.

Table 7 – VER-UC4 requirements

4.5 VAL-UC5: Increasing the safety of an autonomous train through AI techniques

4.5.1 Description of the use case

CAF Signalling is involved in different research projects related to CV&AI-enhanced systems development in order to:

(1) reach a higher autonomy in urban vehicles and
(2) align them with European railway normative.

The objective is to apply CV&AI techniques to improve different autonomous train operation functionalities, such as precision stop, visual odometry, rolling stock coupling operation or person and obstacle detection-identification in railroads.

Figure 11 – CAF Istanbul's fully automated metro

However, as many companies across the sector, CAF Signalling is facing up different computational capabilities challenges for CV&AI-enhanced autonomous train operation, which needs real-time & safety-critical computing platforms for correct performance. The future of CV&AI breakthroughs in railway sector will require large arrays of memory devices at the same accuracy as a Graphical Processing Unit (GPU)-based system, hardware accelerators and new platforms. These achievements will expand the scale of CV&AI processing-calculations making them larger and faster (this means energy-efficiency must improve dramatically).

CAF Signalling will use the FRACTAL approach on AI-enabled computing platforms to execute some functionalities developed in CV&AI field for autonomous train operation.

More precisely, FRACTAL's project use case will focus on:

- Automatic platform detection: It will detect platform area based on train localization information (odometry sensors, platform beacon...) and different visual pattern (visual sensors) detection/identification (characteristic patterns which identifies train platforms). Platform detection functionality will enable CV&AI based automatic train approximation to accurately stop the train.

- Automatic accurate stop at door equipped platforms aligning the vehicle and platform doors: It will perform precise positioning inside platform area using visual patterns detection, identification and tracking in order to reach accurate stopping point and managing automatic train operation (traction and brake commands, ATO functionality). The visual patterns will be designed and chosen to maximize the detection and identification processes in any possible lightness and meteorological conditions. On the other hand, these patterns will be installed according to predefined precise distances to obtain physical accurate measurement from correctly calibrated visual sensors.
- Safe passenger transfer: It will manage automatic safe door enabling (ERMTS functionality) making sure the train is completely stopped in the platform area (using visual sensors) avoiding a) door opening operation if the train and platform doors are not precisely aligned and b) door closing operation if any passenger is getting in/out the train.

4.5.2 Specific technical objectives

This use case technical objectives aim to:

- Integrate the safety-critical high-performance computing platform within a railway control system.
- Testing and evaluation of CV&AI-enhanced autonomous train operation processes over safety-critical high-performance computing platform with actual in-the-field data and operating in the real railway vehicle environment. The use case will perform CV&AI based:
    - Correct automatic platform detection.
    - Accurate automatic stop at door equipped platforms, aligning the vehicle and platform for correct passenger transfer.
    - Correct detection of the passengers who are getting in/out the train (in platform area) avoiding any door closing operation before all train's doors are free of crossing-passengers.

4.5.3 More generic objectives

The CV&AI-enhanced algorithms for (driverless) autonomous train operation will need a further substantial effort to increase the technical readiness level (TRL) before bringing it to the market. CV&AI-enhanced technology must fulfill with strict standards and safety regulation in order to be certified. In addition, regarding the certification process of railway systems and according to EN-5012x standards, CV&AI-enhanced techniques are not currently recommended, so the adoption of this kind of solutions in such a domain is still a challenge. For this reason, the main barrier for exploitation will be increasing the TRL for system certification carrying out all safety requirements.

Safety-critical high-performance computing platforms (with integrated algorithms for CV&AI-enhanced driverless automatic and safe train operations) will help in increasing the TRL for a future possible system certification, thus, bringing expected benefits of AI based technologies to the autonomous railways sector and supporting

an easier marketing of these technologies. The algorithms developed by FRACTAL in the context of the CAF Signalling use-case will be transferred to European railway companies, which will increase their competitiveness and contribute to the Europe 2020 strategy for smart, sustainable and inclusive growth. On the other hand, FRACTAL will establish CAF Signalling in a preferential position as an innovative technology provider in autonomous and safe train operation.

CAF Signalling's general objectives with VAL-UC5 are:

- Give autonomy and decision-making capabilities to vehicles so they can observe and interpret the environment in an independent manner, complementing the information already received from railroad signaling modes.
- Reduce installation and maintenance costs by lowering both complexity and price with new optical sensors and increasing installations' lifecycle.
- Increase flexibility in different railway operations that are attached to delimited areas and delimited time slots depending on the type of railroad and its configuration.
- Enhance variable calculations and operations both in precision and speed with new optical sensors information.
- Increase railway systems safety level.
- Increase railway exploitation capacity and flexibility by CV&AI based more precise measurements (optical metrics, object detection/identification…)

4.5.4 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP8 at the end of the project. Further elaborations will occur in WP8.

Apart from the technical objectives of VAL-UC5 listed in the use case description, in the scope of FRACTAL three objectives are to be fulfilled when integrating CAF's CV&AI-enhanced system in an embedded platform based on FPGA:

- Verify that the algorithms and its dependencies running over office-lab computer can be addressed in an embedded platform.
- Verify that the algorithms running over an embedded platform generate the same outputs (accuracy) when receiving the same inputs as in office-lab tests.
- Verify the CPU, HW accelerator and memory budgets required by the different algorithms on an embedded platform.

The Roadmap first-schema to achieve these objectives is the following:

- Develop VAL-UC5's functionalities over Linux x86-64 machine + Nvidia GPU workstation in office-lab environment.
- Prepare the VERSAL board for future porting:
    - Develop FRACTAL's libraries abstracting inference to final user level.
        - Integrate/compile OpenCV libraries into VERSAL.
        - Integrate/compile ONNX interpreter libraries into VERSAL.
    - Develop FRACTAL's HW Accelerator.
        - Integrate Vitis AI DPU module into VERSAL.

- Port solution to the VERSAL board
- Test solution (compare office-lab environment results with real-time embedded VERSAL HW platform)

To success in VAL-UC5 development and achieve KPI and objectives, CAF Signalling will contribute with:

- Real railway videos:
  - Videos recorded from train cabin (front view)
  - Videos recorded from rear mirror camera.
- Computer Vision based measurement algorithms (C++ over OpenCV):
  - Stopping distance estimation.
- Trained AI models for detection (ONNX):
  - Platform detection.
  - Passenger detection.
- Performance measurement strategy

4.5.5 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**

These requirements are inputs for WP4.
• The platform shall be compliant with non-functional railway equipment requirements:
  ○ EN 50155 (Electronic equipment).
  ○ EN 50125 (Environmental issues).
  ○ EN 45545 (Fire protection).
  ○ EN 50121 (Electromagnetic compatibility).
  ○ UNE EN 61373 (Equipment vibrations).
• The platform shall support real-time performance for UC5 functionality. 10fps (100ms cycle) will be considered as real-time.

Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5. • The platform shall support OpenCV library. • The platform shall have ONNX interpreter. • The HW accelerators shall be compatible with TensorFlow's framework outputs (nice to have).

Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. No specific requirements.
Objective 1: **Open-Safe-Reliable and low power node architecture**

These requirements are inputs for WP3.
Target: commercial node (VERSAL)
• The board shall provide multi-core technology with at least 4 cores.
• The board shall handle multi-threading applications.
• The board shall have at least 60 GFLOPS.
• The board shall provide at least 16GB DDR RAM.
• The board shall incorporate HW acceleration:
  ○ The board should incorporate HW acceleration based on GPU (nice to have)
  ○ The HW accelerators should be programmed with OpenCL (nice to have)
• The board shall incorporate different interfaces (and their Linux drivers):
  ○ The board shall have 2xGbit peripherals
  ○ The board shall have 2xUSB3.0 peripherals
  ○ The board shall have 1xHDMI peripherals
• The board shall have Linux OS.
• The board shall have C++ compiler.

Other requirements
This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP8: • No specific requirements.

Table 8 – VAL-UC5 requirements

4.6 VAL-UC6: Elaborate data collected using heterogeneous technologies

4.6.1 Description of the use case

Nowadays, several emerging edge technologies are combined together within a new disruptive retail paradigm, called *Sentient Spaces*. It represents an advanced ICT based space that has sensing capabilities, an *Artificial Intelligence* (AI) based brain to process information and data collected, and a large amount of actuation capabilities to interact with customers. It is a dynamic space able to adapt itself promoting products according to individual's preferences. Leveraging on it a double positive impact will be possible: the consumer will experience accurate guidance and product information and retailers will be much more efficient, making marketing more targeted and effective.

Figure 12 – Smart Totem illustration

In a sentient space, a smart totem is equipped with intelligent sensors and actuators, such as cameras, that collect data and implement AI based content analysis providing output and actuations. It is then clear that such a totem could have a disruptive impact on retail and shopping mall business, providing personalized advertisements and product recommendations and driving customers towards products.

In such a context, our aim is to make these devices more accessible and faster to use. Leveraging on Fractal methodologies and approaches, studied and defined during the project, advanced AI approaches will be developed and deployed on the edge to process collected data to extract meaningful proximity information and detailed understanding of its surrounding.

The inference is provided by a neural network and rule based approaches, optimized for running on the embedded device installed in the smart totem. Different embedded technologies will be investigated and compared by following the FRACTAL approach and platform.

Information can be detected in terms of customers' gender and age range, effectiveness of marketing campaigns inside the store determining customers attention time for each content promoted. Not only video but also audio processing will be used to detect meaningful data that can be further elaborated providing useful support for targeted advertisement and a personalized marketing strategy. Moreover, audio processing algorithms for in store context awareness exploits audio signal collected to provide user tailored information, contents and services, delivering shopping experience that meets consumer expectations. Both audio processing and video content analysis are based on innovative AI approaches that can be deployed on edge devices without requiring to upload data collected (i.e., video streams and audio signals) to a centralized cloud infrastructure.

In more detail, our scenario is shown in the figure below:

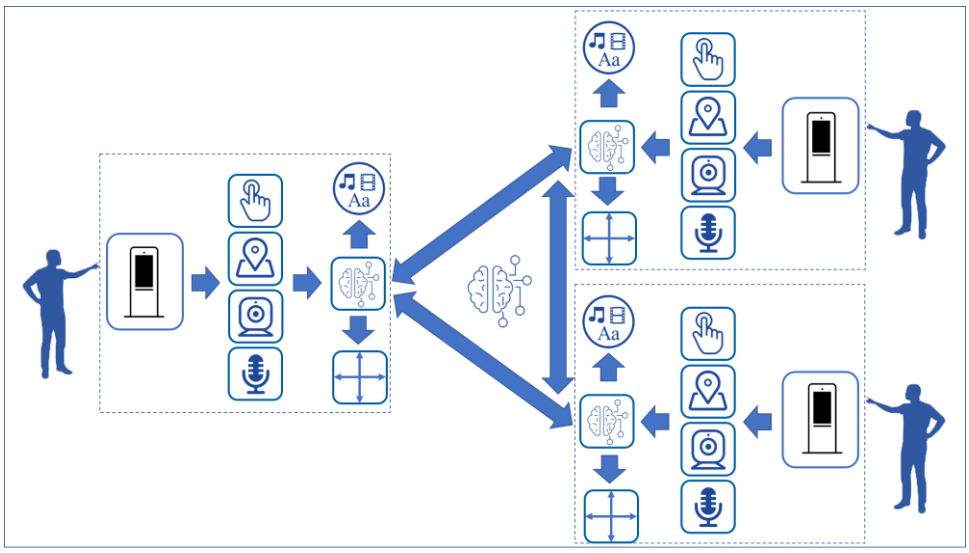

Figure 13 – Schematic representation of the VAL-UC6.

The use case will be composed of a set of building blocks that expect to reach the following TRL:

Use Case Building Blocks	TRL (Expected)	Comment
Smart totems actuators	4	Tailored user support (e.g., content displayed and actions done) automatically selected according to information detected on totems proximity and cooperatively by other totems
Smart totem sensing and local processing	4	Integration of edge computing devices for AI algorithms to infer relevant information by means of audio and video processing
Communications and distributed cooperative strategy	4	Totems will cooperate to coordinate their functionalities and the contents they provide sharing the information locally detected
Age&Gender Classifier	4	Implementation of accelerated CNNs for edge devices
Audio processing algorithms for totem proximity context awareness	4	Exploit audio context awareness to provide user tailored interface, contents and services
Intelligible analytics for context awareness of totem content (e.g., content selection, knowledge extraction from image, audio)	4	Specialized algorithms for black box vs clear box machine learning models
On-board resource management system	4	Deliver real-time performance by adequately managing shared resources in a predictable manner on the target board

Table 9 – Target TRL for VAL-UC6

4.6.2 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP7/8 at the end of the project. Further elaborations will occur in WP7/8.

In particular, the objectives are:

- To verify that the algorithms and their dependencies running over test plant can be addressed in the embedded platform.
- To verify that the outputs provided by the embedded platform in the live environment are consistent with those of the test facility.

- To verify the CPU, HW accelerator and memory budgets are consistent in order to secure node communication and secure data store.

The first Roadmap release to achieve use case objectives will be as follows:

- Develop the UC6's functionalities over Linux machine in office-lab environment.
- Prepare the AI Core series board abstracting inference to final user level:
  - Integrate C++ compiler;
  - Integrate/compile OpenCV libraries;
  - Develop FRACTAL's HW accelerators for the TensorFlow – Keras' framework outputs.
- Integrate user interaction systems (such as camera, microphone, touchscreen display, audio speaker, sensors, etc.).
- Dataset definitions for the AI algorithm learning process.
- Develop the AI solution to achieve the objectives set on user experience, correct outputs and user's interest in the advertised product.
- Test solution.

4.6.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. Considering the defined scenario, the system should guarantee "firm" Real-Time response timing bounds, as established by Miller, 1968 and further explored by Nielsen: o 0.1 seconds is the limit for the user to feel that the system is reacting instantaneously to their direct manipulation – the only necessary feedback is the display of results. o 1.0 second is the limit for the user's thoughts to remain uninterrupted, although they will notice the delay. o 10 seconds is the limit for keeping the user's attention – in other words, the general point of abandonment. Further requirements need to be considered: • Computing needs, including required memory and bandwidth, data bus speed and width, processor speed, and potential need for hardware acceleration. • Authentication and Integrity, it is important to guarantee authenticity, confidentiality and integrity of the transmitted data.

Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5: • The Node shall be able to execute TensorFlow-Keras framework models, or other standard Machine Learning APIs. • The HW accelerators shall be compatible with TensorFlow-Keras framework outputs. • OpenCV Library. The node should support AI solutions to process images collected by cameras to: • Detect user age. • Detect user gender. • Detect people at totem proximity. • Count people, or track people density, in totem proximity. • Compute heatmap. • Detect crowd intensity and variation. • Detect (nice to have) level of attention. The node should support AI solutions to process the audio signal collected by microphones to: • Detect speaker Age. • Detect speaker Gender. • Detect speaker Language. The node should support AI solutions to process heterogeneous data to: • Select content/info to be provided. • Select the output channel among those available (e.g., video, audio, etc.). • Select eventual further output/actuations. The node should/could support distributed learning approaches (e.g., federated learning).
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6: • The node shall support TCP/IP protocol; the ideal network protocol to transport messages among the devices shall be MQTT or any other publish/subscribe communication protocol. • The node must expose a set of APIs which shall allow HTTPs REST calls to and from other nodes, central application and user devices.

Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements are inputs for WP3. Target: commercial node (VERSAL): • The node must acquire images from at least one HD camera. • The node must acquire audio signal from at least one microphone. • The node must support programmable accelerator engines, such as for instance FPGA/programmable logics, or AI engines (e.g., VERSAL's), or in case, GPGPUs. • The Node shall have Linux OS such as Ubuntu or Petalinux. • The Node shall have a C++ compiler and related standard libraries. • The node must support wired connectivity (e.g., Ethernet) in order to ensure network stability, it shall have at least 1GHZ Ethernet connection. • To have a hardware computing node that allows accelerating convolutional neural networks applications. • The node should have a modular and scalable architecture to allow an easy and quick integration of new data sources without changing the architecture. • The node must store data locally in a secure manner. • The node should control an interactive touchscreen display. • The node should control an audio speaker. • To have a monitoring system able to measure response time of tasks, both implemented on microprocessors and accelerators.
Other requirements
This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP7/8: • No specific requirements.

Table 10 – VER-UC6 requirements

4.7 VAL-UC7: Autonomous robot for implementing safe movements

4.7.1 Description of the use case

The "Smart Physical Demonstration and Evaluation Robot" (SPIDER) is an autonomous robot prototype. Within this use case, the Cognitive Edge Node developed in FRACTAL will be integrated in the autonomous robot SPIDER and evaluated against its applicability for performing computationally intensive relevant vehicle functions of variable complexity at the edge of the network (near the source of the data) while still being able to guarantee extra-functional properties (dependability, timeliness) for preserving safety and security operational behaviours.

Figure 14 – Smart Physical Demonstration and Evaluation Robot (SPIDER)

The use case targets two main objectives:

O1: Co-execution of safety-relevant, security-relevant as well as AI based tasks.
Co-execution of safety-relevant, security-relevant as well as AI based tasks without compromising any of the requirements of these functions.

O2: Implement fail-operational capabilities.
Fail-operational capabilities with a single computing device even in the presence of common-cause faults.

The (user-task dependent) computationally intensive relevant vehicle functions might be task dependent, like for instance: enhanced AI-based computer vision and AI based decision making techniques, sensor fusion, the creation of an occupancy grid. All of these are applicable for the demanding requirements of the automotive market. By performing the computationally intensive data processing at the edge of the network, so that the SPIDER robot only sends aggregated data to the cloud, reduces communication bandwidth requirements, and thus fosters node autonomy by reducing the cloud functionality to management and control.

Two SPIDER functions will be deployed on the Cognitive Edge Node platform and use the fractality of the nodes for maintaining safety and security while providing high availability at the same time.

The collision avoidance function (CoA) is a safety-critical function and prevents collisions of the SPIDER with surrounding environment objects to avoid damage and most importantly human harm. The CoA uses four lidar sensors, which constantly measure the distance to environment objects. If one of the objects gets too close to the SPIDER, an emergency brake is initiated.

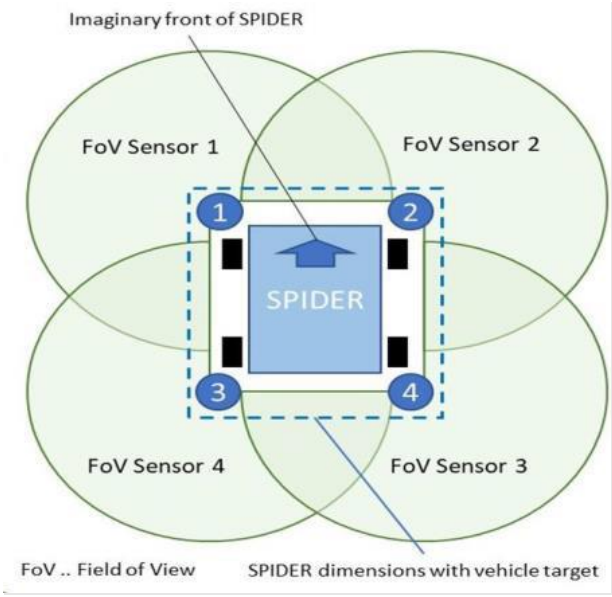

Figure 15 – Sensor setup for collision avoidance function of VAL-UC7

The SPIDER is capable of omnidirectional driving (moving in all directions). Thus, a 360° environment perception with high accuracy of position and range is required. The SPIDER is intended to be operated in a closed environment like a proving ground, where the access of humans is prohibited. However, to ensure maximum safety, the CoA shall detect humans (or objects) approaching the SPIDER from an arbitrary angle and reduce speed or initiate an emergency brake if they come too close. Since the SPIDER can move by its own, the area in which the movement is directed is particularly safety critical. Therefore, if an environment object is detected within this area – called movement zone – an emergency brake shall be initiated.

The hardware platform to be used will be the medium performance node (RISC-V). VIF will run (user-task dependent) computationally intensive applications (like enhanced AI-based trajectory planning, or creation of an occupancy grid), on the FRACTAL Cognitive Edge Node's platform to demonstrate its applicability for the automotive market, where the applicability will be verified by the execution of predefined demanding tests, designed to stress the component. The separated implementation of the functions ensures that neither security issues nor erroneous decisions made by (uncertified) AI algorithms can impact the functional safety.

4.7.2 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP8 at the end of the project. Further elaborations will occur in WP8.

The use case will be composed of a set of building blocks that expect to reach the use case KPI and objectives.

1. **Requirements and specifications**
   Definition and specification of the use case requirements- and technical specifications for the autonomous robot SPIDER. Alignment with other use cases.

2. **Safety concept**
   Providing a safety and security analysis for the integration of the FRACTAL node into the use case. Providing a safety concept with respect to use case. Ensure the compatibility with safety standards.

3. **Development and integration of vehicle functions**
   Develop and integrate vehicle functions and AI algorithms on the target platform (CoA, PTF, AI).

4. **Verification and demonstration**
   Execution of vehicle functions and AI algorithms under real-world conditions. Designed to stress the components.

4.7.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • **Authenticity** VAL-UC7 requires to combine function from different devices or nodes. Thus, the originator of a message needs to be authenticated. The FRACTAL communication framework shall implement methods for verification of messages as well as signing messages (e.g. with a cryptographic key or certificate). • **Integrity** The FRACTAL framework shall provide a cryptographic function to verify integrity of a received message. This function needs to be executed within short process cycles leading to cryptographic functions that allow fast encryption and decryption (e.g. symmetric key rather than asymmetric). • **Predictable timing with 10 Hz loop rates**

The collision avoidance function is based on input from lidar sensors running with 10 Hz each. The CoA functions running on the edge node shall allow a similar rate while the time deviation between processing loops needs to be kept small. Maximum allowed deviation per loop is the half of the processing rate, thus 5 Hz.

- **Global time service**
  The FRACTAL platform shall provide a service for receiving a synchronized time between nodes and give the possibility to synchronize the time with external devices.

Objective 3:

Cognitive and autonomous node requirements

These requirements are inputs for WP5.

- **C++ or Python API**
  The path tracking function of the VAL-UC7 uses AI functions which can be implemented with C++ or Python. Therefore, the FRACTAL framework shall provide a C++ or Python API to access the FRACTAL AI toolkits.

- **OpenCV Library (not mandatory)**
  The support of the OpenCV library may help in developing functions for the CoA as well as the PTF of VAL-UC7. However, the library is not necessary to achieve the main goals and therefore listed as not mandatory.

Objective 4:

Mutable and fractal communication requirements

These requirements are inputs for WP6.

- **TCP/UDP Protocol Implementation**
  The VAL-UC7 uses TCP and UDP for communication with other devices in a distributed network. The FRACTAL node shall support the TCP and UDP allowing to access the communication stack from C++ or Python nodes.

Objective 1:

Open-Safe-Reliable and low power node architecture

These requirements are inputs for WP3.

Target: customizable node (PULP)

- **Floating point unit**
  The FRACTAL platform shall provide a FPU for double precision operations.

- **Multicore**
  Minimum of 2 CPU cores (nice to have).

- **2 GB RAM**
  Minimum CPU memory of 2 GB to support Linux as operating system.

- **Linux**
  Main functions of the VAL-UC7 shall be implemented on a Linux platform. Most suitable distributions would be Ubuntu 16.04, Ubuntu 18.04 or derivates.

- **C++ Suite**
  Main functions of the VAL-UC7 shall be implemented with C++ on Linux. Thus, a compiler, and a debugger are necessary and not mandatory a profiler. The compiler needs to support at least C++11 functions.

- **1 GHz Ethernet**
  The collision avoidance function of the VAL-UC7 uses sensor input from four lidar sensors which are producing 30 MB/s with 10 Hz each. To ensure network stability and keep the transmission delay short the FRACTAL framework shall provide at least a 1 GHz Ethernet connection.

Other requirements

This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP8.

- **Testing ground**
  For testing safety critical functions like path tracking and collision avoidance of the SPIDER robot for VAL-UC7 a sufficient testing ground needs to be prepared. The ground shall be flat, asphalt or compacted gravel, and shall provide sufficient run-off area (5m for low speeds).

- **SPIDER core functionalities running**
  SPIDER robot needs to be prepared for VAL-UC7 to have all base functionalities running that are required for making test drives on a proving ground. Those composes for example the hardware interface and safety controller, motion controller, sensor interfaces, or user interfaces.

- **Lidar sensors available**
  For the collision avoidance function of the VAL-UC7 four lidar sensors with at least 16 lines each and a range of minimum 50 meters are required. The sensors need to be arranged on the four edges of the SPIDER robot to ensure an overlapping field of view of at least 2 sensors. The sensors need to be installed and integrated to the base software stack of the SPIDER.

- **3D simulation available**
  For safety reasons the VAL-UC7 functions shall be tested in simulation before the integration to the SPIDER robot. A 3D simulation is required that includes an environment, the kinematics of the robot, and simulations of the point clouds from the 4 lidar sensors.

Table 11 – VAL-UC7 requirements

4.8 VAL-UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse

4.8.1 Description of the use case

The goal is to improve the warehouse throughput, considering that delays in warehouse operation is critically undesirable, since it has a domino effect on the supply chain. The handling, storage and retrieval of warehouse goods by automated shuttles are optimized using Artificial Intelligence techniques. AI will optimally organize and analyse the masses of generated data, in order to improve the warehouse throughput.

The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could destabilise the system. The shuttles to gain better computational capability to host AI functions shall use the FRACTAL nodes. Real-time Information (e.g. diagnostics, battery health, task) hosted on the shuttle operation are registered in the AI database (Big data). Therefore, the FRACTAL node will be suitable to satisfy the computational requirement at low energy.

The shuttles will be edge-computing nodes that will process real-time information at very high speed through integrated filters. Task handling will be shifted from material flow computer to shuttles with local decision capabilities (e.g. routing and sequencing). The system shall minimize human interruptions resulting from faults.

The warehouse system shall utilize new data flows (via deep learning techniques) to optimize the warehouse throughput.

The following AI features are desirable:

- Establish uninterrupted communication between the shuttles by exploiting machine learning techniques on the aggregated data obtained from signal connectivity monitoring.
- Predictive maintenance: Task that previously led to failure or low performance will be optimized and corrected to improve the warehouse availability.
- Adaptive system: A shuttle system that will adapt independently to new situations within the warehouse.
- Power optimization and improved storage strategy: By optimizing the location of high-velocity goods, while spreading them out in an optimal way to minimize congestion and to improve the retrieval efficiency. Machine learning will be exploited to establish the desired optimal values.
- Route optimization: Aggregated data of route-patterns and delivery efficiency will be exploited through AI application to obtain a higher throughput for the warehouse.
- Pick-up order (Productivity): Using supervised learning techniques with inputs – accumulated pickup list to schedule an optimized system directed picking (Output – result of the best pattern).

- Defined bulk processing of orders. Bulk information is giving to a SWARM including expected timing. The SWARM resolves the solutions to deliver as specified.

4.8.2 Roadmap to achieve use case KPI and objectives

This section summarizes the plan to get a successful use case in WP8 at the end of the project. Further elaborations will occur in WP8.

1. **Requirements and specifications**
   Definition and specification of the use case requirements- and technical specifications for the shuttle based warehouse. Alignment with other use cases.

2. **Shuttle concept**
   Provide a concept for the Shuttle based warehouse system in regards of the FRACTAL nodes key features. Defining a safety concept for the full system in terms of maintenance and other key features.

3. **Shuttle development**
   Developing the Shuttle Hard- and Software to verify the FRACTAL node's capabilities in warehouse automation.

4. **Verification and demonstration**
   Providing a basic warehouse environment to verify and demonstrate the integration of the shuttle node

4.8.3 Requirements

These requirements from the use case are inputs for objectives O2, O3, O4 and O1 (which correspond to WP4, WP5, WP6 and WP3).

Objective 2: **Non-functional (low power, safety, security, high-performance trade-off) requirements**
These requirements are inputs for WP4. • Safety Certification for Black-Channel communication (ASIL 3, ISO 26262) In Order to allow the storage device to be ASIL 3 compliant, the communication channel needs to be certified • Real-Time capabilities The FRACTAL Node should provide real time capabilities in order to correctly control the storage devices with a RT-Patch enabled Linux OS
Objective 3: **Cognitive and autonomous node requirements**
These requirements are inputs for WP5.

• Pathfinding To allow storage and path improvements, the AI of the FRACTAL node should support pathfinding to allow enable the nodes to find the best available shuttle for tasks • Machine learning To allow predictive maintenance features to be developed, machine learning is required in order to predict failures of certain parts and devices
Objective 4: **Mutable and fractal communication requirements**
These requirements are inputs for WP6. • Wireless Communication To allow communication with other FRACTAL nodes and external systems, WiFi Communication is required with a minimum Bandwidth of 300 Mbit/s. Ad-Hoc (mesh) and Access-Point based connections are necessary • Diagnostic protocol to shutdown device on communication loss • 2 1000Mbit/s Ethernet interfaces • Localization features (Localization of device inside of the warehouse)
Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements are inputs for WP3. Target: customizable node (PULP)/commercial node (VERSAL) The FRACTAL node shall provide multi-core technology with at least 2 cores. • The node shall handle with multi-threading applications. • The node shall have at least 800 MHz on each core. • The node shall provide at least 4GB DDR RAM. • The node shall provide at least 32 GB eMMC or similar memory
Other requirements
This section describes requirements that are not related to WP3/4/5/6 but are inputs for the integration of the use case in WP8. • The FRACTAL node shall provide an EtherCAT stack on 1 of 2 Ethernet interfaces • The FRACTAL node shall provide a ProfiNET Master stack • Linux with RT Patch as Operating System • Gyroscope • 1 CANOpen interface (D-SUB9) • Serial TTY interface • 1 USB3 Port

Table 12 – VAL-UC8 requirements

5 Safety, Security & Low Power Techniques

This section relates to objective O2 and pillar 2 and it describes the non-functional techniques addressed in WP4.

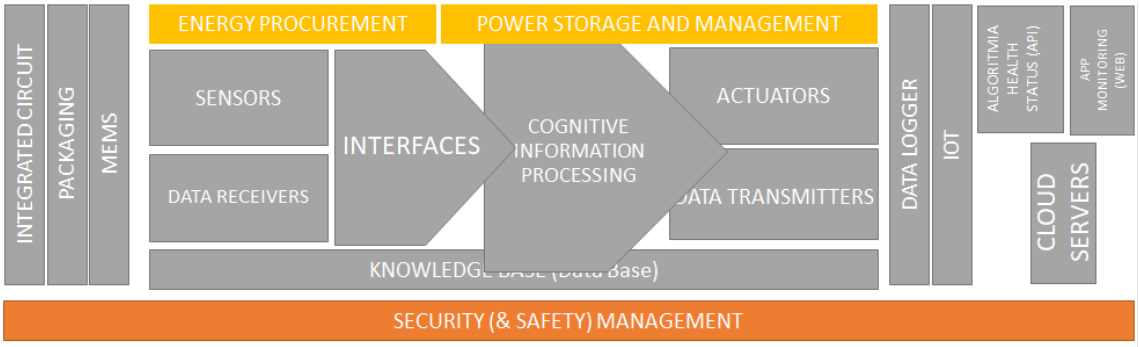

Figure 16 – Blocks from the Cognitive System to adapt for guaranteeing pillar 2

A central idea for the fulfilment of O1 (Design and Implement an Open-Safe-Reliable Platform to Build Cognitive Edge Nodes of Variable Complexity), is to ensure that the FRACTAL system is safe and reliable. In the context of safety, determinism is essential as many sub-services are implemented for the FRACTAL node, as determinism will ensure that the system is predictable at all times. A system has a deterministic behavior if, given an initial state at the instant of time and a set of future timed inputs, the future states and the values and instants of all future outputs are entailed (can be predicted without a doubt). In WP4, the concept of fractality is projected at both the chip level and at a system level. The concept of determinism is used both at the chip and system level to satisfy O1. The time-triggered concept is a known technique used to facilitate the implementation of safety services for a distributed real-time system. The time-triggered concept referred herein provides temporal partitioning of bandwidth to aid applications operate in a deterministic context. Several layer 2 communication protocols such as the time-sensitive networking, and TTEthernet all try to provide determinism on a network level. Determinism at the network level is taking care of by implementing these protocols.

However, as part of the safety requirement proposed for the FRACTAL system, determinism is also proposed at the node level. Therefore, the interconnection of components within a chip should be deterministic. Determinism should be supported at the chip level by networking each functional element of the FRACTAL node in a time-triggered manner, thus as a time-triggered network-on-chip (NoC).

5.1 Interconnection Architecture

Network-on-chip (NoC) technology is a network-based communication system designed for an integrated circuit such as the System-on-Chip. A typical NoC-based MPSoC is shown in Figure 17. It is composed of several components, called nodes, including Processing Elements (PEs), such as CPUs, custom IPs, DSPs, etc., and storage elements (embedded memory blocks). All the Processing Elements are

connected to network adapters through Network Interfaces (NIs). Communication through the NoC is performed by enabling PEs. These PEs send and receive packets through the network composed of switches/routers, and they are connected together through physical links or channels. To ensure determinism, the exchange of messages should include a message class transmitted in a time-triggered fashion. The NoC should support a heterogeneous mix of processing elements, i.e., CPUs, GPUS, and Memory elements.

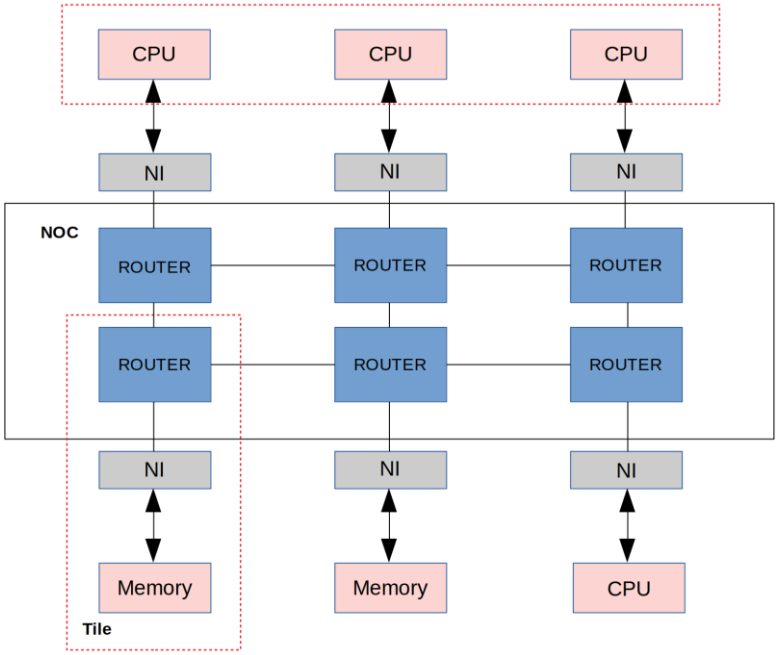

Figure 17 – Typical NoC-based MPSoC

The selection of topology is one of the major important points when designing NoC since the scalability, cost, and power consumption depends on it. The topology describes the structure of the different nodes in SoC. This project shall support at least a mesh topology. The mesh topology specified is projected to provide high scalability and flexibility in comparison with other topologies such as star and ring topology.

There are essentially two different types of switching techniques in NoC, circuit, and packet switching. The majority of NoCs use packet switching since it ensures shorter latencies compared to circuit switching, and the power consumption during the transmission of data is also low.

One of the main objectives of FRACTAL node is to have efficient power consumption, making packet switching a suitable consideration. The FRACTAL project thus specifies wormhole routing for the packet switching, which is one of the primary solutions used to reduce the required buffer size in the router by dividing the packet into a small number of flits. Wormhole switching does not only reduce the power consumption in the whole system but also reduces latency during the transmission of a packet.

Time-triggered NoC supports source-based routing in which the route information is defined in the source while the complete path information is injected into the packet. The need to establish communication between off-chip and on-chip generates the necessity to implement a gateway interface in the NoC. A gateway is responsible for redirecting the messages between NoC and the off-chip communication network. The off-chip and the on-chip system usually operate at different clocks. FRACTAL thus proposes providing support for deterministic communication between two off-chip networks.

The NoC should be scalable and configurable with parameters such as network size, buffer size, number of input and output ports and the number of virtual channels.

In summary, the NoC provides the platform to manage and control the low power and safety services that are planned for WP4. The Time-triggered NoC is proposed in WP4 to provide determinism and ensure predictable timing for safety critical applications such as the VER-UC7. In addition, to ensure the capacity to process time-series data as required by VER-UC2, the major services delivered by WP4 shall be implemented in hardware (the programmable logic of an FPGA) to mitigate processing delays of the WP4 services for the use case application. To correctly control storage devices as required by VAL-UC8, the temporal predictability provided by the time-triggered NoC shall be used to promote the fulfilment of this requirement.

5.2 Low Power Services

5.2.1 Node level

Measurement of the power consumption of nodes and links allows us to know the total power dissipation of the whole system and the particular component for each node. Reducing the power consumption turns into the optimization of the NoC. Different techniques are applied for each node to reduce the power consumption of Fractal nodes.

Clock Gating

Clock Gating is a power management technique used to reduce SoC dynamic power dissipation by removing the clock signal when the circuit is not in use. It is possible to perform this technique by working on the Clock Tree, specifically on a "pruning" of the clock tree, to allow on/off switch for timing distribution in the circuits. The pruning provides (or does not provide) the clock signal in certain areas, that means, e.g. disable the switching of flip-flops with a clear reduction of power consumption. An accurate clock tree design must contain these enabling conditions in order to use and benefit from clock gating technique.

Power gating

Power gating is a technique that provides for each core in the SoC a low power mode, to reduce power consumption by shutting off the electric current to circuit blocks that are not in use, and an active mode, to increase the energy flow to circuit blocks. This

technique works as a runtime adapter to energy requests in the circuit: the low power mode will be activated in case no data needs to be processed, while the active mode will operate in case of elaboration data needs.

The power gating is more delicate to use than clock gating because it increases time delays in the circuit. Furthermore, it affects design architecture more than clock gating, and indeed some trade-offs are necessary between the amount of leakage power saving in low power mode and the energy dissipation to enter and exit the low power mode.

DVFS

Dynamic Voltage and Frequency Scaling is a combination of two power management techniques. The voltage and the frequency can be adjusted in real-time mode depending upon the current needs.

The Dynamic Voltage Scaling allows the reduction of voltage usage in a hardware component, for example, to preserve power in mobile devices, or the increase of voltage usage to support high-performance requests.

The Dynamic Frequency Scaling adapts the frequency of a processor depending on the needs of the current task. For example, the reduction of frequency is used to reduce power consumption in battery-powered devices or helps to reduce the side effects of circuits heating. These two techniques often appear combined since lower frequencies require lower voltage for the digital circuit's proper functioning.

5.2.2 System level

Data aggregation and compression are data processing processes used to reduce the amount of data which could be transferred in a system. In energy-constrained systems, we have to consider minimizing the amount of communication needed for data exchange among nodes. For example, in a wireless sensor network, the energy resources and communication range could be a problem because of the expensive data transmission costs in general, but it can be reduced using suitable data compression and aggregation technique.

Data Compression

Data Compression is a process used to encode information, in order to reduce the amount of data needed for a piece of the given information, used for storage or transmission of data. However, some reasons can motivate data compression, such as saving space, compatibility, gain in processing time, security, and others.

In general, data compression follows two steps: compression and decompression. These operations may be further divided into two different categories: lossless, which can be reversed to the exact original data without loss of details and without errors, and lossy, where the processing causes errors or loss of details from the original data. Lossless compression is useful in applications where every information is relevant, for example, a character in a text, while a lossy compression may be acceptable in applications where the loss of information does not compromise its functioning, for example, a single frame in a video.

Common standards techniques of data compression are Run-length encoding (RLE)¸ Discrete Cosine Transform (DCT), Burrows-Wheeler Transform (BWT)¸ Discrete Wavelet Transform (DWT), Huffman Coding, and others.

Data Aggregation

Data aggregation is the process of collecting and aggregating data from various sources for a certain type of end-use. Data aggregation techniques remove data redundancy and improve energy efficiency in a system; combined with a data aggregation protocol, they can also reduce network traffic.

The compression is based on the repetition of proper data from a node. This is why when neighbouring nodes have identical data; the compression process is called aggregation. For example, considering a complex network composed of several heterogeneous nodes, the neighbouring nodes' data is highly correlated spatially and temporally, and it can lead to the base station receiving redundant information. The aggregation of data, therefore reduces this redundancy and, consequently, the data processing. System or network architecture plays a central role in identifying the right technique or protocol to adopt.

In addition, the Xilinx Versal platform provides advanced features to optimise power through its ACAP (Adaptive Compute Acceleration Platform) architecture. Nevertheless, to mitigate the consumed power by the services implemented in the NoC, Frequency scaling is extended to routers that do not fall temporally on the message transmission path in the NoC. In WP4, a context monitor shall be used to observe both off-chip and local states (e.g. slack time). It is planned to utilise the slack information for power management such as performing voltage and frequency scaling accordingly. This is particularly useful in fulfilling the low power consumption requirement of VER-UC3

5.3 Safety Services

Safety services[8] are implemented to guarantee the safety of applications that run in the FRACTAL nodes. These services should contain fault tolerance capabilities at the network level, including detection, containment, and masking of faults. The services will support both soft and hard real-time applications. WP4 targets its implementation of safety services in two layers. The top layer is the FRACTAL hierarchical system which consists of interconnected FRACTAL nodes. The bottom layer services are implemented within a FRACTAL node.

At the hierarchical layer, WP4 specifies that each FRACTAL node should incorporate seamless redundancy services to verify the correctness of messages sent by each Fractal node. By sending redundant packets on different routes, it is possible to detect a fault when the redundant packet is different at the destination. In WP4, it is

[8] Safety certification will be addressed in deliverable D2.2.

intended to incorporate seamless redundancy services in the communication protocol between FRACTAL nodes. It is also intended that each FRACTAL node should provide support to integrate built-in self-test and monitoring services to maintain safety and reliability. At the bottom layer, WP4 plans to implement seamless redundancy within the NoC for communications between the constituent resources.

The network should include means to monitor and control multicore interference, which will be deployed building on specific Performance Monitoring Units (PMUs). This is a requirement for WP3, which should provide the PMUs. Services to enforce time and space diversity for redundancy should also be deployed, thus enabling some form of light lockstep execution that allows avoiding common cause failures. For that purpose, WP3 should provide interfaces to create redundant processes which execute with some staggering.

5.4 Security Services

Security services are implemented in Fractal nodes to assure confidentiality, integrity, authentication, authorization and non-repudiation.

The design of the cybersecurity measures should follow the well-known design and evaluation patterns (IEC 62443…) to facilitate the cybersecurity certification of the node in the future.

In Fractal, we are assuming a SL2 level of security according to IEC 62443. In this level, we are assuming that the system is not going to be tampered with. In other words, the root-of-trust is the device itself. More complex attacks on the system requiring hardware access of a great number of resources are not contemplated.

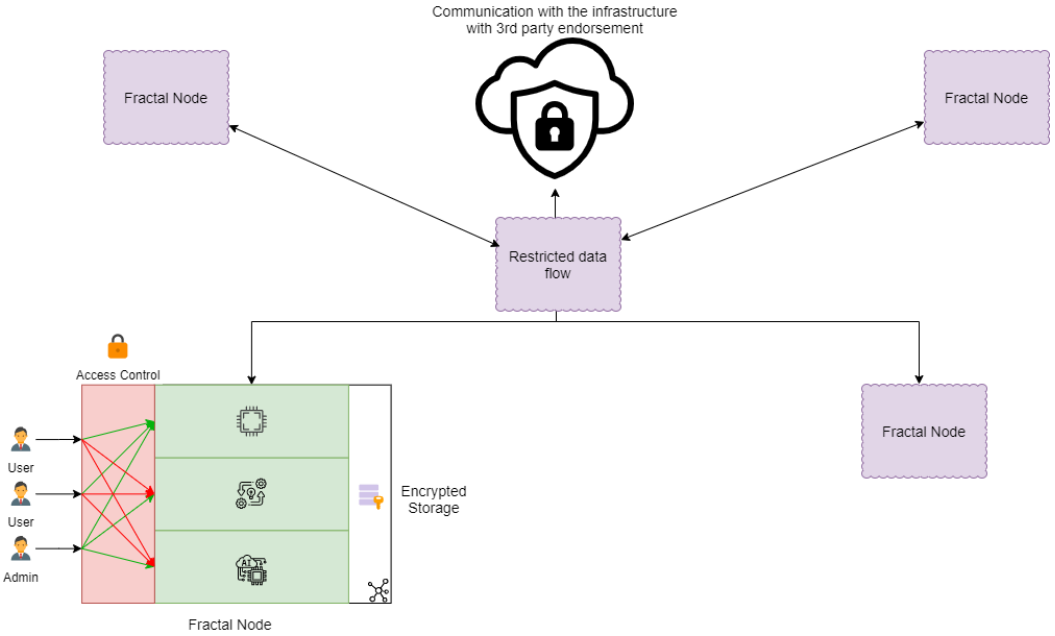

Figure 18 – Fractal security services at node and system level

5.4.1 Node level

Each node should include cybersecurity features to comply with its cybersecurity services.

Access Control

One important feature of these types of systems is the ability to control which users can access the systems and which ones cannot. Moreover, each user may have different privilege access. For example, an admin may be able to access to all the processes but a user only to the process relevant to him. Also, all the login activity will be monitored, to better understand what happened in the node in case of a breach.

Encrypted Storage

For some use cases, the Fractal node is going to obtain or use user's personal information. The node needs to have the capacity to encrypt and decrypt this information. In the case that the node is compromised, the data should not be obtainable by an untrusted 3rd party.

5.4.2 System level

Since PMUs deployed for safety reasons expose execution information, their information could be used to implement side-channel attacks. Therefore, an appropriate security layer should be built on top to limit access to such PMUs. This may pose some process/task privilege management, or information obfuscation means for the PMU on WP3.

Encrypted and MACed communication

In order to assure that the communication with external entities (other Fractal nodes or an external infrastructure) is secure, each node will have the required capabilities to cypher and MAC incoming and outgoing messages.

Key and certificate infrastructure

The secure communication can only take place when each element has valid credentials in the form of public certificates and private keys. Nevertheless, if a certificate is revoked, which entity should provide a new one? How can we assure that the system is trustable? For this, a small PKI architecture will be implemented. This architecture will allow the revocation and renovation of certificates and it will be used a trust point in the system.

5.5 Development methods in time-triggered scheduling

Typically, a scheduler decides which task will be executed, the time of task execution, and which resources will execute that task in cases of multiple resources. Due to the safety nature of some use cases, such as the UC7 (Autonomous robot for

implementing safe movements), temporal predictability is required. WP4 will provide predictability at the fractal node level using the NoC and at the hierarchical fractal layer by integrating a time-triggered communication for the interaction between fractal nodes. As a result of these temporal requirements, the scheduling services planned in WP4 shall support time-triggered paradigms.

Decisions in time-triggered systems are carried out at specific time slots; therefore, a scheduler that supports the scheduling of tasks for a semi-static time-triggered resource is desired. Meta-schedulers are usually invoked after any context event (slacks, faulty events, faulty nodes) within the system at run time, providing the ability to support real-time response capabilities, which are requirements seen in UC6 and UC8. It is planned in WP4 to implement an AI-based meta-scheduling scheme for both the FRACTAL node and hierarchical systems to fulfil these requirements. The search space for scheduling increases considerably as the context events increases. It is planned to use the AI scheduling strategy that incorporates a machine learning model trained at design time to decrease computation resources considerably at runtime. An AI-based scheduler facilitates handling the complexity of adaptive systems via predictable behavioral patterns established by static scheduling algorithms.

Also, the solutions in WP4 shall use the support for the context events such as dynamic slacks to enable frequency and voltage adjustments of the cores and NoC. The voltage and frequency adjustments promote low power consumption. This is significant for UC3 as it is projected to increase the system's battery life.

The large number of context events that could occur usually entails an increase in the time required for re-scheduling during adaptation procedures. The FRACTAL project proposes the use of AI to handle re-scheduling. Therefore, the FRACTAL system will support AI-based schedulers.

5.6 Requirements flowing down to WP3

Objective 1: **Open-Safe-Reliable and low power node architecture**
• PMUs measuring multicore interference [PULP]. • Interfaces to create redundant processes which execute with some staggering [PULP]. • Process/task privilege management or information obfuscation means for the PMU [PULP]. • The system should produce time-bounded decisions (reworded from "The system should produce decisions at least at 10 Hz rate, possibly at a 30" Hz rate.") • Time-triggered communication shall be supported for interaction between FRACTAL nodes

Table 13 – WP4 requirements flowing down to WP3

6 AI and safe autonomous decision

This section relates to objective O3 and pillar 3 and it describes the cognitive and AI techniques addressed in WP5.

The goal for the studies conducted in WP5 is to integrate complex physics knowledge of the environment (including self-organization, brain networks, model-based agents, nature forming processes, emergence, and stochastic algorithms) in combination with AI algorithms to obtain a Predictive, Prescriptive, and Trusted edge.

Currently, edge-computing based implementations basically collect, process, and send upwards data with no predictive or adaptative capabilities or autonomy. FRACTAL will add intelligence to the node in order to create a mutable edge node with context awareness as well as predictive capabilities.

To achieve the goals set by objective O3 and pillar 3, WP5 will perform the following activities:

- Develop AI algorithms for building a synthetic representation of the node's operating environment.
- Develop AI algorithms for evaluating a node's potential actions in the simulated operating environment, helping in AI decision-making at the edge.
- Develop AI algorithms for dependable, fault-tolerant, intelligent decision-making dynamic control feature at FRACTAL node level and system level.
- Develop AI algorithms that ensure mutability, i.e., a node's capability to adapt temporally and spatially to changes in its context by changing its configuration and fractality level, depending on the operating environment and system requirements.

Blocks from the Cognitive System to adapt for guaranteeing pillar 3 are highlighted in Figure 19.

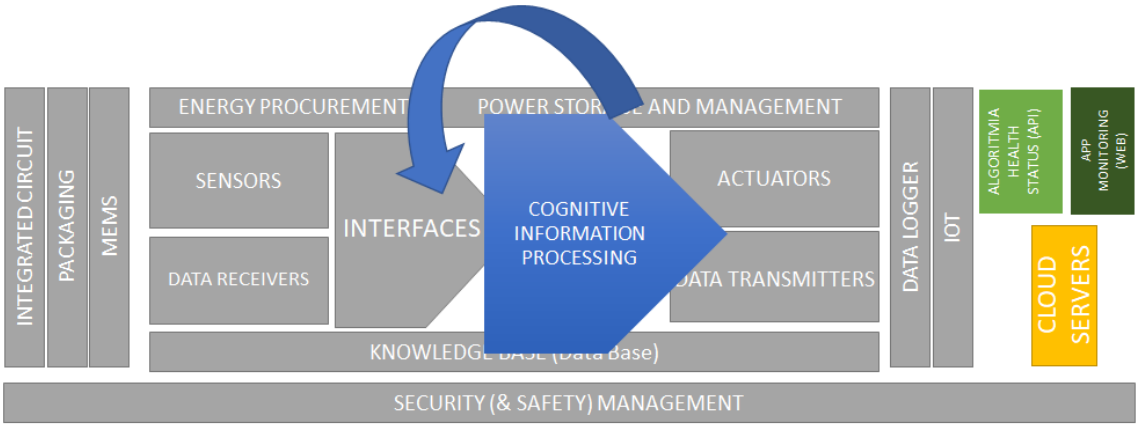

Figure 19 – Blocks from the Cognitive System to adapt for guaranteeing pillar 3

In order to meet the criteria set by objective O3 and pillar 3, the requirements related to AI and safe autonomous decision are further divided into following categories:

- Communication (lead by MODIS, Section 6.1)

- Distribution (lead by IKER, Section 6.2)
- AI performance (lead by RULEX, Section 6.3)
- Data & model lifecycle (lead by HALTIAN, Section 6.4)
- Inference (lead by UOULU, Section 6.5)
- Learning (lead by UOULU, Section 6.6)
- Run & development environment (lead by ZYLK, Section 6.7)

In the above-mentioned sections, the requirements are at first discussed and defined in general followed by specific requirements set by the use cases, when applicable. Finally, Section 6.8 defines requirements for WP3 from WP5 point of view.

6.1 Communication requirements

The communication structures are implemented to guarantee proper communication between the FRACTAL nodes with particular attention to methods of redundancy and protection of communication from external attacks. An approach which could be adapted and deepened to each use case to ensure the secure payload application data between nodes and cloud is depicted in Figure 21.

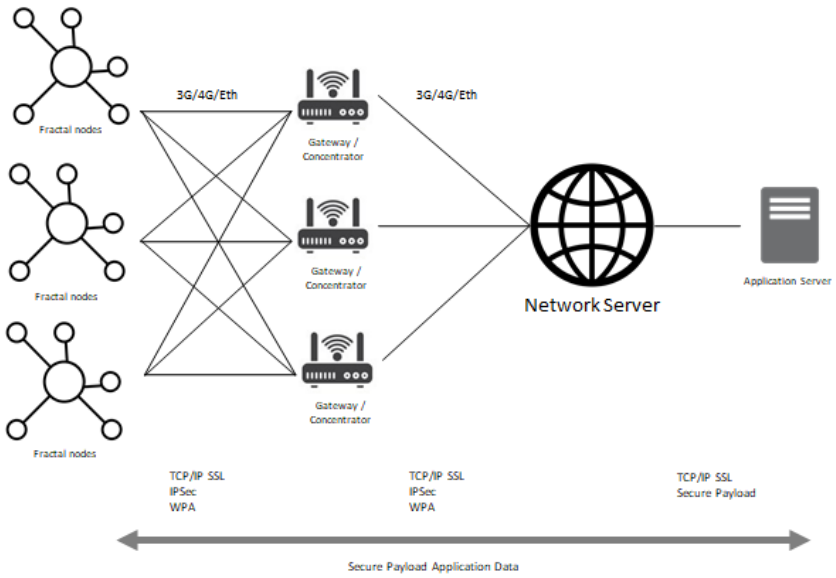

Figure 20 – Secure Payload Application Data

In particular, for each use case we will describe the network architecture focused on high level communication and embedded sensor communication and we will present functional and innovative solutions to safeguard the transmitted data. In order to achieve the goals, we will consider encryption techniques and security protocols at the transport layer, network layer and link layer. A general approach which could be adapted and deepened to each use case is depicted in Figure 21.

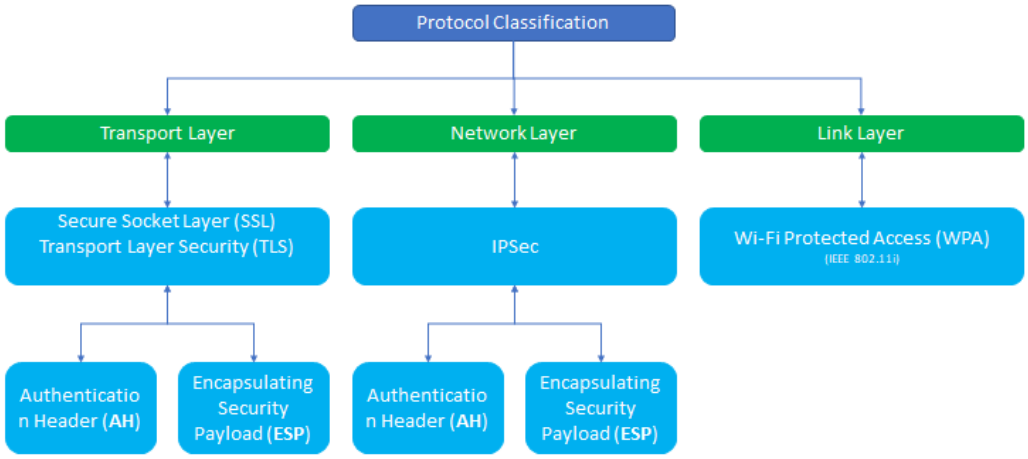

Figure 21 – Encryption techniques and security protocols at the transport layer, network layer, and link layer

All network architecture will be adaptable to the needs of each UC and in particular, network dimensioning should meet the following requirements:

- **Channels**: The data link channels should meet the needs of each use case and, at the same time, maintain compliance with the European regulations relating to the connection type.
- **Protocols**: The design of communication protocols should ensure stable and safe communication. The system should be able to guard the transmitted data from cyber-attacks and to ensure the protection of sensitive data.
- **Bandwidth:** The bandwidth for data communication should be dimensioned in such a way as to ensure the possibility of adding new system features. Another very important thing about bandwidth sizing is that it should avoid the states of saturation since they can affect the system functionality.
- **Latency:** The communication design should meet the need of each components in terms of maximum communication latency to avoid events of link down or loss of connection. The latency parameters are also important for performance monitoring system.

Note: Channels, protocols, bandwidth and latency will be quantified during the UCs specific delivery activities (waiting for partners inputs).

6.2 Distribution needs

Distributed Artificial Intelligence (DAI) refers to an approach for solving complex learning, reasoning, and decision-making problems; on which the problem is divided into smaller subproblems and distributed to autonomous (or semi-autonomous) intelligent processing nodes (usually called agents). These agents handle the

subproblems individually and communicate and interact with each other to integrate and combine their partial solutions to solve the overarching problem and achieve the established goals. This approach is well suited to face and solve large and complex problems, characterized by physically distributed knowledge and large-scale data managing. In FRACTAL, the distribution of the AI and machine learning models and algorithms will be addressed under the perspective of three different main dimensions:

- Centralization vs. decentralization
- Hierarchy
- Opportunism / dynamicity

These main dimensions are discussed in general in sections 6.2.1, 6.2.2, and 6.2.3. Section 6.2.4 defines the needs set by the use cases for Distributed Artificial Intelligence.

6.2.1 Centralization vs. decentralization

Standard machine learning approaches require centralizing the training data on one machine or in a datacenter. However, recently various approaches are advocating for the decentralization of this approach by locally training AI models on various decentralized devices holding their own local data.

In FRACTAL, both approaches should be considered to meet the requirements of different AI models for the different use cases. On the one hand, the decentralization of AI models leads to *smarter* models (which better fits to the specific requirements of a particular device), lower latency, and less power consumption. On the other hand, for more complex models requiring large-scale data from different sources and high computing capabilities, it may be necessary to train the models on the cloud and then deploy them on the edge devices.

Another emerging and interesting approach will be federated learning on which edge devices learn collaboratively from a shared model while keeping their own training data locally. The shared model is first trained in a centralized fashion on a server using a large-scale centralized dataset and then, the distributed devices download the model and improve it by using their own local data (federated data).

6.2.2 Hierarchy

Hierarchical learning is inspired by the human's ability to conceptualize the world with different abstraction levels. In a similar manner, in hierarchical learning a complex task is divided in simpler tasks, whose output is used for the accomplishment of the complex task.

In FRACTAL, depending on the use case or the functionality, complex AI models may be deployed by using different layers in a hierarchical organization on which high-level algorithms are fed with the outputs of low level-algorithms focused on simpler tasks, for the accomplishment of more complex tasks (divide and conquer). For example, for the use case 5, a complex task such as *automatic accurate stop at door equipped platforms aligning the vehicle and platform doors* may be divided on

different simpler tasks (e.g., automatic platform detection, door position detection, stopping distance prediction), which predicted outputs will be used for the accomplishment of the complex task.

6.2.3 Opportunism / dynamicity

In dynamic application domains, such as those considered in the use cases for the FRACTAL system, the environment could change during problem solving. In this kind of environments, intelligent agents must be opportunistic in order to take most of the available resources in each moment. Therefore, it is necessary that the developed AI models and algorithms could be dynamically deployed on the different layers of the FRACTAL system in an opportunistic fashion, in order to achieve an efficient execution in terms of power consumption, latency, connectivity, etc. For example, at a given moment, a device may have enough connectivity to connect to a centralized AI model in a cloud platform trained with large-scale data from different nodes, whereas when the connectivity is lost it may use its own model (decentralized) trained with local data.

6.2.4 Use case needs for Distributed Artificial Intelligence

UC1: Improving the quality of engineering and maintenance works through drones	
AI application	1) "*Supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface.*" 2) "*Monitoring of both workforce and machinery within a construction area, by deploying a WSN that provide information about the status and location of the workers in real time*".
Centralization vs. decentralization	No distribution of learning required. However, federated learning or other decentralized learning architectures could help enhancing model's performance. For application 1), training a model with a huge dataset of images could be heavy in terms of data processing and model training. Thus, initially, models could be built on the cloud computing platform, with more powerful computing resources/capabilities, over a common (and bigger) dataset of structural status images. Once the initial model has been built and trained it could be deployed on each node (federated learning) where it could be improved by training it with its own local data (i.e., the images captured by a particular drone).

Hierarchy	Hierarchical learning is not required; however, it could be used to divide a complex learning task into several simpler learning tasks, which will allow to solve the complex learning problem. For example, in application 2), different models for different prediction subtasks could be built. Over those models, a higher-level model could be built for predicting more complex tasks. For example, different models could be used for predicting workers and machinery trajectories in real time. Over the predictions made by these models, a higher complexity level model could predict whether, considering the predicted trajectories, a collision would occur or not.
Opportunism / dynamicity	Not identified yet.

Table 14 – UC1 DAI needs

UC2: Improving the quality of automotive air control	
AI application	Predictive maintenance of the components in an automobile engine air-path.
Centralization vs. decentralization	No distribution of learning required. However, decentralized learning architectures could help enhancing model's performance. For example, the predictions made by decentralized models could be built using in-use data of physical products within a powertrain (in-vehicle data capturing the behavior of components such as turbochargers, oil pipes, valves or coolers), and combined with the predictions made by centralized models built using in-use data from testbeds for powertrains (endurance runs, such as contactors and fuses used in testbenches and batteries.
Hierarchy	Hierarchical learning is not required; however, it could be used to divide a complex learning task into several simpler learning tasks, which will allow to solve the complex learning problem. The overarching goal of anticipating engine air-path problems before they occur, could be divided in different prediction tasks, each of them focused on a particular air-path control strategy. For example, different models focused on classifying the different parts composing the final product as healthy or erroneous could be combined with anomaly detection models and models predicting the remaining

	lifetime of the product. After building the different models, a high-level model would take as input the predictions of these models, in order to anticipate problems.
Opportunism / dynamicity	Not identified yet.

Table 15 – UC2 DAI needs

UC3: Smart meters for everyone	
AI application	A low-cost machine-vision based application to read conventional meters.
Centralization vs. decentralization	Centralized learning is required. In order to minimize the power consumption and computing resources of the nodes (e.g., memory), the data and the models for these use case will be centralized on the cloud computing platform. Thus, the image recognition algorithms will be built and trained on the cloud platform and nodes will just send the required data to the cloud platform, where the meter stands will be extracted.
Hierarchy	Not identified yet.
Opportunism / dynamicity	Not identified yet.

Table 16 – UC3 DAI needs

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
AI application	A machine vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block.
Centralization vs. decentralization	No distribution of learning required. However, federated learning or other decentralized learning architectures could help enhancing model's performance. This use case uses pre-trained object recognition models (e.g., tiny YOLO (You Only Look Once)). However, these models could be trained also with local data with the aim of fine tuning them for the use case data.
Hierarchy	Not identified yet.

Opportunism / dynamicity	Not identified yet.

Table 17 – UC4 DAI needs

UC5: Increasing the safety of an autonomous train through AI techniques	
AI application	On-board AI-enabled computing platform for autonomous train operations.
Centralization vs. decentralization	No distribution of learning required. However, federated learning or other decentralized learning architectures could help enhancing model's performance. Models will be built and trained in a centralized fashion in an office-lab environment and then will be distributed in different nodes deployed on the vehicles. With the objective of giving vehicles autonomy and decision-making capabilities so they can observe and interpret the environment in an independent manner, these models could be improved with local data of the vehicle, training them in a decentralized fashion (federated learning).
Hierarchy	Hierarchical learning is not required; however, it could be used to divide a complex learning task into several simpler learning tasks, which will allow to solve the complex learning problem. In this use case, a complex task such as automatic accurate stop at door equipped platforms aligning the vehicle and platform doors may be divided on different simpler tasks (e.g., automatic platform detection, door position detection, stopping distance prediction), whose predicted outputs will be used for the accomplishment of the complex task.
Opportunism / dynamicity	Not identified yet.

Table 18 – UC5 DAI needs

UC6: Elaborate data collected using heterogeneous techniques	
AI application	A smart totem is equipped with smart sensors and actuators like, for example cameras, that collect data and implement AI based content analysis providing output and actuations.
Centralization vs. decentralization	The node should/could support distributed learning approaches (e.g., federated learning).

	Initially AI models could be deployed in a centralized fashion in the cloud platform where more data and computing resources are available. However, in order to build smarter totems, which better fit to the specific requirements of a particular scenario, models could be decentralized and trained with local data from distributed nodes (federated learning).
Hierarchy	Hierarchical learning is not required; however, it could be used to divide a complex learning task into several simpler learning tasks, which will allow to solve the complex learning problem. For example, interacting with customers and promoting products according to the individual's preferences is a complex task that may be divided in smaller and simpler subtasks. For example, products recommendations could be made upon the audio and video analysis performed by other AI models, products preferences predicted from customers input, etc.
Opportunism / dynamicity	Not identified yet.

Table 19 – UC6 DAI needs

UC7: Autonomous robot for implementing safe movements	
AI application	The "Smart Physical Demonstration and Evaluation Robot" (SPIDER) is an autonomous robot prototype. Within this use case, the Cognitive Edge Node developed in FRACTAL will be integrated in the autonomous robot SPIDER and evaluated against its applicability for performing computationally intensive relevant vehicle functions of variable complexity at the edge of the network (near the source of the data) while still being able to guarantee extra-functional properties (dependability, timeliness) for preserving safety- and security operational behaviors.
Centralization vs. decentralization	Models build in a centralized manner will be deployed on the fractal nodes.
Hierarchy	Hierarchical learning is not required; however, it could be used to divide a complex learning task into several simpler learning tasks, which accomplishment will allow to solve the complex learning problem. Complex tasks, such as AI based decision making techniques may require to be built upon the predictions

	made by specific models focused on simpler tasks (e.g., enhanced AI-based computer vision, collision detection functions, etc.).
Opportunism / dynamicity	Not identified yet.

Table 20 – UC7 DAI needs

UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	
AI application	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyze the masses of generated data, in order to improve the warehouse throughput. The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could de-stabilize the system. Real-time Information (e.g., diagnostics, battery health) hosted on the shuttle operation are registered in the AI database (Big data).
Centralization vs. decentralization	Fully distributed decision making (swarm intelligence) by shuttle agents.
Hierarchy	Not identified yet.
Opportunism / dynamicity	Not identified yet.

Table 21 – UC8 DAI needs

Proposal:	
AI application	N/A
Centralization vs. decentralization	Models could be built and trained with centralized data on one machine or in a datacenter or in a decentralized approach on which AI models are trained on different nodes holding their own local data. Another approach will be federated learning on which edge nodes learn collaboratively from a shared model while keeping their own training data locally.

Hierarchy	Learning tasks could be solved by single algorithms or hierarchical learning algorithms on which a complex task is divided in simpler tasks, whose output is used for the accomplishment of the complex task.
Opportunism / dynamicity	N/A

Table 22 – Proposed requirements for Distributed Artificial Intelligence

6.3 AI Performance requirements

AI algorithms should enable autonomous decisions in the FRACTAL node. While the objective is to create a single framework for managing different AI applications in different domains, each use case is employing different algorithms, has different objectives and therefore the expected performance of the AI node may differ significantly. For this reason, the different use case scenarios have been considered trying to highlight which are the expected performances for each of them. The combination of all these requirements allows the implementation of proper algorithms and approach to fulfill the different expectations in the different application domain. The FRACTAL architecture should also allow a scalable approach where resources can be added according to the requests of the application. These dimensions have been considered in evaluating the performance requirements:

- Efficiency, i.e. the time needed by AI algorithms to build a model and to apply a model in real time scenarios.
- Effectiveness, i.e. the accuracy of the AI inference. This dimension requires the definition of proper KPI that of course depends on the specific use cases. Moreover, the effectiveness of AI results depends not only on the AI method itself but also on the quality of the used data.
- Reliability and availability: how much the AI system should be reliable, i.e. providing its outputs in any situation.

These main dimensions are discussed in general in Sections 6.3.1, 6.3.2, and 6.3.3. Section 6.3.4 defines the needs set by the use cases for AI performance.

6.3.1 Efficiency

Concerning efficiency, we should keep in consideration two different aspects: the efficiency in making inferences, i.e. for applying an already existing model to streaming data, and the efficiency in building models, starting from new data.

In most of the applications the efficiency in building a model is not an issue since the model used for inference is supposed to be quite static. Therefore, a refresh of the model is needed if: (a) there is some major change in the dataset, or (b) the performances are decreasing significantly. In both cases, the re-training of the model is done off-line manually or in some cloud environments. In any case, this is an operation that does not require high efficiency since the old model can keep working while the new model is trained.

Regarding the efficiency in making inferences, this of course depends on the specific application. Most of the applications involve video streams that should be analyzed quickly to allow the inference to be continuously updated with the stream. Some less critical applications can wait for some seconds to make a decision, while others need a continuous generation of inferences to allow real time autonomous decisions. In these cases, the inference rate ranges from 10 frames per second (fps) to 30 fps. This means that the whole decision cycle should be completed in 0.03-0.1 s. Notice that, as it will be explained later, the goal of AI modules is to allow autonomous decisions also in safety-critical systems, so delays in generating an inference and therefore a decision may result in potentially bad behaviors. For the goal of the project, a human supervision is always established to avoid catastrophic events if AI fails to produce a good decision in the right times.

6.3.2 Effectiveness

Defining the requirements of the FRACTAL system in terms of effectiveness is quite hard. The effectiveness of an AI method does not depend only on the methods itself, but above all on the quality of the data used for generating the model. Moreover, evaluating if the quality of a decision system is good or bad depends on the context: a level of accuracy could be acceptable in some fields while it is not sufficient in another contexts. Even the measure used to evaluate an AI system depends on the type of problem and on the goal of the analysis. Consider for example the case of image recognition in video stream, which is transversal in different use cases. In general, if some objects, people, or situations should be recognized in a video frame, it is important that such inference is done with high accuracy. For example, an application could consist in detecting whether a given objects is present in the frame or which is the age of a person looking at the camera. For all these cases, it is hard to define a target on this before starting the planning and experimentation of the different components, but we could assume that an accuracy of about 90% is a good target for many applications. This means that 90 out of 100 predictions (e.g. predicting whether an object is present in the frame) are correct. But, according to the requirements and the implementation of the use cases, it also possible that different KPIs are used. For example, the number of false negatives could be reduced as much as possible. In this case, it is accepted that the system has a higher number of false positives (i.e. of scenes where the object is detected even if it is not present actually), but we should avoid the opposite situation, where the object is present but is not detected. KPIs and target for those KPIs will be defined during the project by the single use case owners.

6.3.3 Reliability and availability

The system is designed for 24 hours a day operation (H24) and to work in several situations, including different light and weather conditions. Moreover, all the situations that can somehow affect the quality of the AI inference, such as occlusions, graffiti, and stains, should be taken into account. To enable good decisions also in these situations, it is important that the datasets used for generating the AI model contains also examples of “irregular” conditions.

Some use cases are explicitly excluding the industrial application and therefore reliability issues are less relevant. On the other hand, other applications are more safety critical and need to have a higher level of reliability. For example, a system for detecting people or obstacles in autonomous driving vehicles needs to perform correct inferences in any situations. It is worth noting that for the purposes of the project, a human supervision is always present, so that catastrophic consequences of wrong or delayed decisions are avoided.

6.3.4 Use case needs for AI performance

UC1: Improving the quality of engineering and maintenance works through drones	
AI application	An UAV based platform will be used to detect and monitor incipient cracks on concrete structures. The detection will be performed by an autonomous AI application that will use the imagery acquired by the drones to determine the presence of cracks.
Efficiency	To be defined.
Effectiveness	To be defined – Without knowing the actual structures, conditions and dataset it is impossible to define a target. Nevertheless, according to the preliminary test, it is expected to obtain a >90% precision
Reliability and availability	To be defined – Without knowing the actual structures, conditions and dataset, it is impossible to define an expectation for the system's reliability. However, it is important to include in the dataset all the potential difficulties (i.e. textures, graffiti, stains, etc.) expected to be found during the operation, in order to maximize the robustness of the system.

Table 23 – UC1 AI performance needs

UC2: Improving the quality of automotive air control	
AI application	This UC is aiming at developing analytical solutions that allow predictions about future incidents in an automotive air-path.
Efficiency	The system should be able to process data at a 10 Hz frequency.
Effectiveness	To be defined.
Reliability and availability	The system should be able to produce a prediction quickly in any situation.

Table 24 – UC2 AI performance needs

UC3: Smart meters for everyone	
AI application	Identifying numbers in an image via Convolutional Neural Networks upon request.
Efficiency	The system should detect about 16 digits and produce an output in less than 1 s, preferably in 10-100 ms. Training is done offline sporadically.
Effectiveness	To be defined.
Reliability and availability	The system is run upon request, so there is no need of continuous availability. Nevertheless, the system should be able to provide an output whenever it is required.

Table 25 – UC3 AI performance needs

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
AI application	The UC4 will used the inference of Tiny-YOLO neural network consisted of convolutional layers, pooling layers, fully connected layers, activation functions and batch normalization. The goal of the algorithm is to take as input a video stream and show on the output (display) the location of the detected objects within the stream as well as their labels based on the class that they belong.
Efficiency	• The system should be capable to process the frames from video stream at a rate higher than 30 fps. • The system implements inference of neural network that is previously trained, and no additional training will be performed.
Effectiveness	The level of correctness on detection and classification of the objects in the video stream should be at least 85%.
Reliability and availability	Even though dependability is highly relevant for industrial application by intention we exclude this for UC4. The primary focus for the use case is real-time property.

Table 26 – UC4 AI performance needs

UC5: Increasing the safety of an autonomous train through AI techniques	
AI application	People detection in the platform when passenger transfer (people getting on/out)
Efficiency	• Expected analysis frequency (inference): 10fps • Model training frequency: Months (it will be done offline, not in FRACTAL node)
Effectiveness	Precision in detecting persons should be at least 90%.
Reliability and availability	The system should work in any condition (different visibility conditions; light, weather, occlusions). The system cannot stop working although driver will be in charge of safety to avoid severe consequences.

Table 27 – UC5 AI performance needs

UC6: Elaborate data collected using heterogeneous technologies	
AI application	This use case includes three specific AI based blocks: • to process images collected by cameras to detect heterogeneous data like user age/gender, detect and count people at totem proximity, etc. (more details available in Section 4.6) • to process audio signal collected by microphones to detect speaker age, gender, and language • to process data generated from the aforementioned AI blocks and from other data sources to select content and information to be provided, output channels among those available and other eventual actions.
Efficiency	Efficiency is still to be defined. As reported in Section 4.6, 10 seconds is the limit for keeping user's attention. Therefore, this is the limit for the whole process of elaborating video and audio signals collected, and consequently to select and display the content.
Effectiveness	KPI are currently under definition
Reliability and availability	The system is supposed to continuously work 24/7 to provide personalized advertisement and information support. Its AI blocks have different expected reliabilities (they are currently under definition). In case of failure, the totem should commute to a traditional behavior displaying some predefined generic contents

Table 28 – UC6 AI performance needs

UC7: Autonomous robot for implementing safe movements	
AI application	Path Tracking Function which allows to steer a vehicle from a start point to a target point following a predefined path and evading obstacles (static and dynamic) along this path. This function takes as input a cost map (or occupancy grid) representing the vehicle's environment and vehicle state data such as velocity and orientation. The output of the function are control values affecting the linear velocity and the angular velocity of the vehicle. The function is derived by means of a Reinforcement Learning approach using a reward strategy rewarding proximity to the path and penalizing crashes with obstacles.
Efficiency	It is planned to use the AI systems in real-time like scenarios, meaning that the path tracking function should be used to steer a vehicle in a real-world scenario. Thus, the AI system should give a prediction with a frequency of 10 Hz. It is planned that the AI system should be re-trained only if additional features (such as smooth driving behavior, lane keeping...) are required. A re-training would take about one to several days and will not be performed on the FRACTAL board.
Effectiveness	As performance indicators we introduce metrics measuring how good the tasks of path following and obstacle avoidance in test scenarios can be solved. Especially we consider metrics measuring • proximity to the given path, • awareness of obstacles. Our aim is to be able to follow a given path without calculating control commands that would lead to a crash into an obstacle in far more than 90%.
Reliability and availability	For normal operation the function has to deliver results with minimum 10 Hz. Absence of updates leads to an emergency brake situation.

Table 29 – UC7 AI performance needs

UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	
AI application	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyze the masses of generated data, in order to improve the warehouse throughput. The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could de-stabilize the system. Real-time Information (e.g., diagnostics, battery health, task) hosted on the shuttle operation are registered in the AI database (Big data).
Efficiency	To be defined.
Effectiveness	To be defined.
Reliability and availability	To be defined.

Table 30 – UC8 AI performance needs

6.4 Data & model lifecycle concept

The concept consists of a distributed machine learning platform that manages:

- the distribution of the inference work between the edge and cloud
- dedicated embedded and edge machine learning algorithms
- data management, that e.g., takes care of the GDPR related issues
- ethical issues related to artificial intelligence
- security and privacy.

In the FRACTAL approach the technologies are developed together with the business models for them in a co-design fashion and demonstrated in use cases of radio base stations and elevator group optimization, among others. Relations to a wider community and ecosystem are also taken care of by dissemination but also with direct interactions. All these aspects are elaborated in the subsections below.

6.4.1 Embedded and edge machine learning algorithms

Deep learning has been the most important advance in machine learning for the last decades. The power of the deep learning systems has incited the industry to apply deep learning algorithms to many applications under various constraints. However, deep learning systems demand excessive power consumption, computational capacities, and memory volumes, require large training data sets, and are poor in quantifying uncertainty. This prevents deep learning systems from being deployed in resource-limited devices and environments, such as hand-held and mobile devices. The costs of the components that can perform deep inferences and the power

demands prohibit the broad deployment of these systems into mass markets. One of the solutions to this challenge is to transfer all data to edge computing facilities or cloud services to perform the heavy inference work and transfer the results back to the frontal devices. However, this system architecture wastes the computing capacities of the frontal devices and has no clear balance between the communication cost and computation cost, thus degrading the performance and increasing the burden of the communication between the entities. In addition, it increases the load of the edge facilities and the cloud services rendering them and thus being unable to provide services for real-time applications.

In FRACTAL, we aim to develop state-of-the-art machine learning algorithms that are suitable for complex distributed systems with flexible hardware and communication configuration. The solutions build on improved deep learning techniques specifically designed to tackle the above challenges, and on probabilistic modelling techniques that are ideally suited for resource-limited environments. Based on our recent advances in various techniques, we will apply multidisciplinary research to achieve our goals together with other stakeholders. Our approaches include, but are not limited to:

- Apply the next generation deep learning technology called Operational Neural Networks (ONNs) to achieve a superior learning and generalization performance with minimal network complexity, which in turn will improve both the classification and anomaly detection accuracies as well as the computational efficiency for real-time applications
- Using Compressive Sensing (CS) and Sparse Approximation (SA) techniques to find more sparse representations of a given signal, which are easier to learn and require less memory and computations in the analysis, hence more efficient for real-time applications on edge devices
- Combining graph analytics, graph learning, and deep learning architecture to process data in a graph structure and explore effective techniques to perform hierarchical inferences in a multi-layer manner. The proposed approach can greatly improve the accuracy and efficiency of a massively distributed system where data can be modelled in a graph structure.
- Probabilistic programming for cost-efficient development of probabilistic machine learning solutions for resource-limited devices and environments, enabling development of machine learning solutions that are aware of their own uncertainty, can be trained on limited data while relying on strong prior knowledge, and can protect the privacy of the data providers.

6.4.2 Distributed machine learning platform

Embarrassingly Parallelization of sequential machine learning algorithms can yield both reduced time and resource utilization. Also, data locality can reduce fetch times and eliminate extra network and I/O costs, saving both time and resource. Therefore, modern schedulers may take advantage of these and automatically employ caching and avoid shuffles where data locality is already achieved. However, in a cluster at the edge or cloud, there can be a mix of different resource types, and therefore, the jobs must be analysed before any resource scheduling policies can be applied.

Platforms are the key components of the FRACTAL solutions. They offer the means to perform real-time analytics in the Cloud or at the edge depending on the requirements of FRACTAL applications.

However, the heterogeneity of FRACTAL platforms introduce challenges in terms of deployment and management.

- State-of-the-art distributed computing and machine learning frameworks and dedicated hardware need to be evaluated with respect to the environment defined by FRACTAL applications (i.e., real-time constraints, heterogeneity, and large distributed networks).
- In FRACTAL, analytic tasks are performed in real-time, which poses additional requirements to the deployment framework of machine learning (ML) microservices in a secure environment. We will evaluate the ML model deployment strategies for the FRACTAL analytics platform based on the ML algorithms and the targeted Service Level Agreements (SLAs). This will lead to the design and implementation of the ML model deployment framework for edge analytics.
- Realtime analysis at the edge strives interfacing between the platforms and network, QoS management, network management, and the resilience of the platforms in terms of resource and security.
- Currently, ML applications communicate with the top-level scheduler through a resource request/release system in the state-of-the-arts platforms. There is no existing protocol for allowing schedulers or applications to optimize or scale down resource utilization.

In FRACTAL, we aim to leverage the existing solutions pertaining to the edge and distributed computing and extend their capabilities towards the realization of real-time FRACTAL applications. This requires the development of a new type of scheduler that considers the heterogeneity of FRACTAL platforms and the needs/constraints of the business verticals. The performance of the scheduler needs to be monitored by a novel QoS framework and supported by the underlying network via innovative solutions. Furthermore, the FRACTAL framework must ensure the resiliency of FRACTAL platforms and applications. We aim to demonstrate the applicability and usefulness of the FRACTAL distributed machine learning platforms in the field.

6.4.3 Data management

The data management approach defines a systematic perspective on critical components that are needed to establish businesses breaking organizational borders between public entities, private companies, and consumers. Data strategy incorporates views for (but not limited to) data owner consent, technical foundation, culture, business, and metrics.

The data architecture relies primarily on the data flow from sources to consumers: data mediation (collection), data refining and storing, and data offering. Secondarily, the architecture focuses on making data usable: data governance as well as security and privacy. Governance covers a wide range of topics for cataloging the data, access control, ownership, traceability, and performance to name some. The FRACTAL

project defines the components to fulfil the data strategy to ensure data can be shared and used by the stakeholders.

In FRACTAL we aim to enable real-time applications which require low latency communication for efficient performance. Addressing this challenge requires a fresh view of the design and management of data on the network level. Particularly, we need to ensure smooth data flow for FRACTAL applications. The novel design and management solutions should rely on adequate network traffic model and system level simulator. These two components combined will enable tracing bottlenecks and vulnerabilities in network and system architecture, including evaluation and reliability prediction (e.g., network level delays, blocking probability, forced termination probability) of real-time FRACTAL applications.

6.4.4 Security and privacy

The analysis of large-scale data in AI and ML on distributed platforms (edge computing) raises important questions from the information security standpoint. These questions gravitate around matters such as ownership, storage, privacy, and protection, among others. Such matters echo moral and legal concerns as security breaches could lead to compromise of private data and sensitive information, consequently, harming individuals, businesses, and governments in the process. Considering the high stakes, development and deployment of AI and ML in applications such as Industrial IoT and Cyber-Physical Systems (CPS) should be met with information security measures from the outset of a system's lifecycle to the later stages of use and maintenance.

In terms of privacy, different forms of differential privacy are quickly becoming established as the standard approach for privacy-preserving machine learning. Most current deployments are based on local differential privacy, which allows strong individual privacy guarantees but can seriously compromise the utility of the data. There are emerging research results in combining differential privacy with secure multi-party computation that allow much higher utility, but further research is needed to make these practical at scale and in the edge computing context.

In FRACTAL, we aim at addressing the need for information security measures in secure system design, development, and application by considering different aspects of security in edge computing context. Such aspects include security and privacy challenges in edge computing, emerging security regulations and legislations (e.g., GDPR) and their influence on edge computing, current and emerging security threats and risks in edge computing, and available techniques and practices for building security into edge computing. By studying and considering these aspects, we shall propose new security approaches that account for seamless integration of security within edge computing context.

6.4.5 AI ethics

IoT (Internet-of-Things) devices, CPSs and AI are becoming increasingly ubiquitous. Autonomous vehicles are starting to be active in the traffic and nearly every large website uses varying degrees of AI to e.g., to provide recommendations to its users. Large tech corporations are currently developing their own multi-purpose AIs (IBM Watson etc.). Though these systems affect nearly every individual in developed

countries and even in the developing world, they are largely developed by private organizations with little outside influence.

Most of these systems are black boxes. Outside observers seldom know what data goes into the box, and sometimes not even the developers know exactly what happens with the data. The only clear thing is what the system ultimately does with the data in terms of actions. However, when designing systems with large societal impacts, it is important to know exactly why they act the way they do. When an autonomous vehicle gets into an accident, those involved will want to know why, if only to determine who is to be held accountable.

In designing systems that affect virtually everyone, regardless of whether they personally use the system or not, ethical design principles are needed. Transparency is needed for accountability and responsibility to be possible.

The approach taken by FRACTAL is to enable Ethically Aligned Design into practice within the FRACTAL framework context. The aim is to enhance trust in the system analysis, design, construction, deployment, and evaluation phases. Tasks are defined to deal with ethics-related concerns such as accountability, responsibility, transparency, and privacy. As the result, potential conflicts for enabling real-time AI/ML processing products & environments are identified and strategies are defined to overcome the raised concerns. Empirical evaluation of the proposed strategies is performed in use cases.

Additional key approach adopted by FRACTAL project refers to ambition of studying aspects that can be automated in the process of ethical analysis. Any action requiring human control and oversight is a potential bottleneck and thus a risk factor from a performance viewpoint. It is also a quality issue since humans are prone to errors when systematic decision making is expected. This is required by an ethical concept called fairness. Automation will be implemented by means of a prototype of a recommender system.

6.5 Inference requirements

Inference refers to the runtime functionality where the AI application predicts, makes decisions, or provides recommendations in its operating environment. We describe the inference requirements in relation to the following aspects:

- Level of autonomy (e.g., AI application provides recommendations, or it acts independently to fulfil goals)
- Distribution of inference: are there a number of nodes contributing to the inference results? (e.g., model splitting, model pruning, model ensembling)
- Co-operativity of distributed agents (i.e., are the agents able to seek a global maximum in learning effectiveness and efficiency, or do they look for some equilibrium?)
- Decision/action environment of the FRACTAL agent (i.e., what is the agent specifically making decisions/predictions/recommendations about? On what data?)
- Goals and subgoals (i.e., what is the agent aiming for?)

Section 6.5.1 defines the needs set by the use cases for inference in the context of the above-mentioned aspects.

6.5.1 Use case needs for inference

UC1: Improving the quality of engineering and maintenance works through drones	
AI application	1) *"Supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface."* 2) *"Monitoring of both workforce and machinery within a construction area, by deploying a WSN that provide information about the status and location of the workers in real time"*.
Autonomy	No autonomy required.
Distribution	Local inference only.
Co-operativity	N/A
Environment	1) Agent will provide predictions based on video images in a machine vision setting. 2) Agent will provide proximity alerts in a wireless mobile sensor network setting.
Goals	1) Detection of structural faults. 2) Safety of workers.

Table 31 – UC1 inference needs

UC2: Improving the quality of automotive air control	
AI application	Predictive maintenance of the components in an automobile engine air-path.
Autonomy	No autonomy required.
Distribution	Local inference only.
Co-operativity	N/A
Environment	Agent will provide predictions of future engine faults using runtime sensor data of physical products within a powertrain (in-vehicle data capturing the behavior of components such as turbochargers, oil pipes, valves or coolers) or runtime sensor data from testbeds for powertrains (endurance runs,

	such as contactors and fuses used in testbenches and batteries).
Goals	Anticipation of engine air-path problems before they occur.

Table 32 – UC2 inference needs

UC3: Smart meters for everyone	
AI application	A low-cost machine-vision based application to read conventional meters.
Autonomy	No autonomy required.
Distribution	Local inference only.
Co-operativity	N/A
Environment	The fractal RISC-V platform will be interfaced with a low power camera that can take pictures of the meter. In a second step, the platform must analyze the picture and extract the meter stand. The main challenges of this task will be to reliably detect digits in an image with a pattern recognition algorithm, on a platform with only a few 100 kB of memory and in a power envelope of a few milliwatts such that the device can remain active for multiple years.
Goals	Read conventional meters remotely with machine vision.

Table 33 – UC3 inference needs

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
AI application	A machine vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block.
Autonomy	No autonomy required.
Distribution	Local inference only.
Co-operativity	N/A
Environment	*"The LLOD building block takes as an input a video stream generated from the camera. The stream is handed to a device that runs algorithm for computer vision on top of it. Once the frame processing is finished the device publishes the results on the display. The output is localization of the objects in the image and their classification based on the*

	group that they belong. All this will be performed in real-time as the input video stream flows."
Goals	Enhance automation processes with the intelligent capability to detect and recognize objects visually: • The LLOD shall be able to detect and locate the objects in an input image. • The LLOD shall be able to recognize the detected objects. • The LLOD shall perform detection and recognition of all objects in the image through a single observation. • The inference of LLOD shall be able to process the input images in real-time. • The operation of the inference of LLOD shall be isolated within the edge node.

Table 34 – UC4 inference needs

UC5: Increasing the safety of an autonomous train through AI techniques	
AI application	On-board AI-enabled computing platform for autonomous train operations.
Autonomy	Autonomous operation of train velocity, doors.
Distribution	Local inference only. Some interaction with platform-based sensors or systems can be introduced?
Co-operativity	N/A
Environment	• The agent will detect platform area based on train localization information (odometry sensors, balise information…) and different visual pattern (visual sensors) detection/identification (characteristic patterns which identifies train platforms). Platform detection functionality will enable CV&AI based automatic train approximation to accurate train stop. • The agent will perform precise localization inside platform area using visual patterns detection, identification and tracking in order to reach accurate stopping point and managing automatic train operation (traction and brake commands, ATO functionality). The visual patterns will be designed and chosen to maximize the detection and identification processes in any possible lightness and meteorological conditions. On the other hand, these patterns will be installed according predefined precise distances to obtain physical accurate measurement from correctly calibrated visual sensors.

	• The agent will manage automatic safe door enabling (ERMTS functionality) making sure the train is completely stopped in the platform area (using visual sensors) avoiding a) door opening operation if the train and platform doors are not precisely aligned and b) door closing operation if any passenger is getting in/out the train.
Goals	• Correct automatic platform detection. • Accurate automatic stop at door equipped platforms, aligning the vehicle and platform for correct passenger transfer. • Safe passenger transfer, with a correct detection of the passengers who are getting in/out the train (in platform area) avoiding any door closing operation before all train's doors are free of crossing-passengers.

Table 35 – UC5 inference needs

UC6: Elaborate data collected using heterogeneous techniques	
AI application	A smart totem is equipped with smart sensors and actuators like, for example cameras that collect data and implement AI based content analysis providing output and actuations.
Autonomy	Autonomous interaction with users.
Distribution	Distributed decision-making (i.e. activity) among AI agents in each totem.
Co-operativity	Agents are fully co-operative.

Environment	• Smart totems, each with:
	  ○ A number of sensors (video, audio, proximity) and actuators (audio setup & video screen for selected content).
	  ○ An AI agent, capable of detecting user age, gender, proximity, other user context based on sensor data, and interacting with the user by selecting content.
	• Clients interacting with the totems by way of the totem's actuators.
	• AI agents in each totem co-operating for inference and learning.

Goals	Overall goal is to maximize the impact of personalized advertisements and product recommendations, driving customers to buy products.

	Each node should support AI solutions to process images collected by cameras to: • detect user age • detect user gender • detect people at totem proximity • count people in totem proximity • compute heatmap • detect crowd intensity and variation • detect (nice to have) level of attention The node should support AI solutions to process audio signal collected by microphones to: • detect speaker age • detect speaker gender • detect speaker language The node should support AI solutions to process heterogeneous data to: • select content/info to be provided • select the output channel among those available (e.g., video, audio, etc.) • select eventual further output/actuations

Table 36 – UC6 inference needs

UC7: Autonomous robot for implementing safe movements	
AI application	*"The "Smart Physical Demonstration and Evaluation Robot" (SPIDER) is an autonomous robot prototype. Within this use case, the Cognitive Edge Node developed in FRACTAL will be integrated in the autonomous robot SPIDER and evaluated against its applicability for performing computationally intensive relevant vehicle functions of variable complexity at the edge of the network (near the source of the data) while still being able to guarantee extra-functional properties (dependability, timeliness) for preserving safety- and security operational behaviors."*
Autonomy	The agent is an autonomous, wheeled robot, capable of independent movement.
Distribution	Local inference only.
Co-operativity	N/A
Environment	Sensors:

	• VAL-UC7 four lidar sensors with at least 16 lines each and a range of minimum 50 meters. Actuators: • hardware interface and safety controller, motion controller, sensor interfaces, user interfaces. 3D simulated environment. Proving ground for real hardware tests.
Goals	Co-execution of safety- and security relevant functions with AI functions on a single hardware platform.

Table 37 – UC7 inference needs

UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	
AI application	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyze the masses of generated data, in order to improve the warehouse throughput. The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could de-stabilize the system. Real-time Information (e.g., diagnostics, battery health, task) hosted on the shuttle operation are registered in the AI database (Big data).
Autonomy	Shuttle agents make autonomous decision on, e.g., routing and sequencing. No central controller.
Distribution	Fully distributed decision making (swarm intelligence) by shuttle agents.
Co-operativity	Agents are fully co-operative.
Environment	Sensors: • To be defined Actuators: • To be defined

Goals	• Improve smart warehouse throughput. Delays in warehouse operation are critically undesirable since they have a domino effect on the supply chain. • Minimize human interruptions resulting from faults. • Establish uninterrupted communication between the shuttles by exploiting machine learning techniques on the aggregated data obtained from signal connectivity monitoring. • Predictive maintenance: Task that previously led to failure or low performance will be optimized and corrected to improve the warehouse availability. • Adaptive system: A shuttle system that will adapt independently to new situations within the warehouse. • Power optimization and improved storage strategy: By optimizing the location of high-velocity goods, while spreading them out in an optimal way to minimize congestion and to improve the retrieval efficiency. Machine learning will be exploited to establish the desired optimal values. • Route optimization: Aggregated data of route-patterns and delivery efficiency will be exploited through AI application to obtain a higher throughput for the warehouse. • Pick-up order (Productivity): Using supervised learning techniques with inputs – accumulated pickup list to schedule an optimized system directed picking (Output – result of the best pattern). • Defined bulk processing of orders. Bulk information is given to a SWARM including expected timing. The SWARM resolves the solutions to deliver as specified.

Table 38 – UC8 inference needs

Proposal:	
AI application	N/A
Autonomy	Ranging from fully autonomous agents to a hierarchical model of arbitrary depth where a higher-level agent may exert some control over the lower level agents associated with it.
Distribution	Ranging from local inference to distributed inference over the FRACTAL hierarchy.
Co-operativity	Ranging from co-operative agents to multi-agent systems where at least some of the agents are non-co-operative.

Environment	N/A
Goals	N/A

Table 39 – Proposed requirements for inference

6.6 Learning requirements

Learning refers to the functionality where the AI application is provided data on its operating environment and the application learns how to fulfil its goals. Learning may be conducted in a separate, offline phase, before production usage (runtime). Alternatively, learning and inference may be intertwined such that they alternate or even happen simultaneously. We describe the learning requirements in relation to the following aspects:

- Distribution of learning (i.e., centralized, federated, decentralized?)
- Co-operativity of learning (i.e., are the agents able to seek a global maximum in learning effectiveness and efficiency, or do they look for some equilibrium?)
- Model selection / model architecture

Section 6.6.1 defines the needs set by the use cases for learning in the context of the above-mentioned aspects.

6.6.1 Use case needs for learning

UC1: Improving the quality of engineering and maintenance works through drones	
AI application	1) *"Supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface."* 2) *"Monitoring of both workforce and machinery within a construction area, by deploying a WSN that provide information about the status and location of the workers in real time"*.
Distribution	No distribution of learning required.
Co-operativity	N/A
Model	No requirements on model or model architecture.

Table 40 – UC1 learning needs

UC2: Improving the quality of automotive air control	
AI application	Predictive maintenance of the components in an automobile engine air-path.

Distribution	No distribution of learning required. Federated learning or other distributed learning architecture could enhance both efficiency and effectiveness of learning.
Co-operativity	N/A
Model	No requirements on model or model architecture.

Table 41 – UC2 learning needs

UC3: Smart meters for everyone	
AI application	A low-cost machine-vision based application to read conventional meters.
Distribution	No distribution of learning required. Federated learning or other distributed learning architecture could enhance both efficiency and effectiveness of learning.
Co-operativity	N/A
Model	No requirements on model or model architecture.

Table 42 – UC3 learning needs

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
AI application	A machine vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block.
Distribution	No distribution of learning required. Pre-trained model.
Co-operativity	N/A
Model	Pre-trained CNN (YOLO).

Table 43 – UC4 learning needs

UC5: Increasing the safety of an autonomous train through AI techniques	
AI application	On-board AI-enabled computing platform for autonomous train operations.
Distribution	No distribution of learning required. Federated learning or other distributed learning architecture could enhance both efficiency and effectiveness of learning.
Co-operativity	N/A

Model	Must support ONNX.

Table 44 – UC5 learning needs

UC6: Elaborate data collected using heterogeneous techniques	
AI application	A smart totem is equipped with smart sensors and actuators like, for example cameras, that collect data and implement AI based content analysis providing output and actuations.
Distribution	The node should/could support distributed learning approaches (e.g., federated learning).
Co-operativity	All agents are assumed fully co-operative.
Model	• Age & Gender classifier: FPGA-based CNN implementations for edge devices. • Possibly other models for context awareness.

Table 45 – UC6 learning needs

UC7: Autonomous robot for implementing safe movements	
AI application	*The "Smart Physical Demonstration and Evaluation Robot" (SPIDER) is an autonomous robot prototype. Within this use case, the Cognitive Edge Node developed in FRACTAL will be integrated in the autonomous robot SPIDER and evaluated against its applicability for performing computationally intensive relevant vehicle functions of variable complexity at the edge of the network (near the source of the data) while still being able to guarantee extra-functional properties (dependability, timeliness) for preserving safety- and security operational behaviors."*
Distribution	No distribution of learning required. Federated learning or other distributed learning architecture could enhance both efficiency and effectiveness of learning.
Co-operativity	N/A
Model	Unknown. • Reinforcement learning based 3D simulated environment available.

Table 46 – UC7 learning needs

UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	
AI application	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyze the masses of generated data, in order to improve the warehouse throughput. The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could de-stabilize the system. Real-time Information (e.g. diagnostics, battery health, task) hosted on the shuttle operation are registered in the AI database (Big data).
Distribution	Distributed learning between agents. Independent and identically distributed (IID) data may be assumed.
Co-operativity	Agents are fully co-operative.
Model	DNN? To be defined.

Table 47 – UC8 learning needs

Proposal:	
AI application	N/A
Autonomy	Ranging from local learning to distributed (e.g. federated) or fully decentralized in the FRACTAL hierarchy.
Co-operativity	Ranging from co-operative multi-agent learning to non-co-operative multi-agent learning.
Model	N/A

Table 48 – Proposed requirements for learning

6.7 Run & development environment requirements

In this section, an overview of the environment requirements is offered for developing and running AI artifacts. It is important to notice here that these two phases are separated, which means that the development environment (researcher computer) and the run environment (FRACTAL node) are actually different systems. They both will be closely related for technological alignment, but many development tools may be included that the run environment may not need.

Apart from this, the environment design must accomplish some basic features to obtain a good user experience. Ideally, it should provide a wide range of possibilities in terms of tools and technologies, in order to offer a flexible work methodology, and

also taking into an account the development process phases to provide an optimal workflow. The environment development must facilitate the complete process of computation.

In order to make the first approximation to the development environment, we assume that the target FRACTAL devices will be Linux based, providing a so flavored OS capable of providing a basis for the installation of the major part of the technologies mentioned on this section. The definition of a complete Run & Development environment for AI techniques on the Edge is a complex task, and for this reason this design will be enriched during the project, providing a solution to the necessities it may arise.

6.7.1 Available tools

On one hand, the FRACTAL node should be able to support different **programming languages** to permit the usage of some of the predominant tools on the AI field.

C, C++ runtime should be included. Also, other managed languages such as Python or Java are also strongly recommended.

This will ensure compatibility with many of the current ML/DL (deep learning) technologies (Python), and advanced analytics technologies (Java JVM), which include strategies to distributed computation management.

Moreover, from the development environment perspective, it could be useful to identify some tools (such as IDEs, learning frameworks, etc.) for defining a common development methodology for the FRACTAL project.

Once this is defined, the development environment should include tools for the generation of **ML, DL models**. For this, some of the most used tools at the present could be included, such as TensorFlow, PyTorch, Spark ML, etc. The generated models should then be able to run on the FRACTAL nodes. At the present, there exist several tools and projects that provide different functionalities to manage the **model deployment**. The principal phases may include model exposition for obtaining predictions, re-training, etc. In the analysis and selection of these tools, the model lifecycle management design (previously defined in this WP) should be considered. To give some examples, MLFlow, Apache Submarine, etc.

Moreover, the support of **advanced analytics** tools is proposed for the FRACTAL node. These could suppose great benefits from the interoperability perspective of the nodes, as these tools are already designed for distributed execution paradigm. Some of them could be ETLs (Extract, Transform, Load) (such as Apache NiFi/MiNiFi), data storages (SQL flavored), message queueing systems (such as Apache Kafa, Rabbit MQ), real-time event processing systems/CEP motors (such as Apache Flink), notebook tools for data science (Zeppelin, Jupyter).

Finally, focusing on neural networks the FRACTAL node should be able to support the Open Neural Network Exchange (ONNX) standard for machine learning interoperability. This standard will ease the process of using a variety of frameworks, tools, runtimes and compilers and will facilitate the use of the Low Energy Deep Learning Library (LEDEL) (see 8.2.2.2).

6.7.2 Usable technologies / technology stacks

Technology stack is a set of tools for implementing some intended IT idea. As a rule, a tech stack consists of programming languages, frameworks, libraries, various development tools and programming approaches themselves. The synthesis of all of the above determines the viability and competitiveness of the application, its functionality, scalability, and further maintenance. FRACTAL is a large project that will need a large technical stack to maintain the integrity and performance of the application. Multiple layers of programming languages and frameworks will be necessary to perform for different data conditions. Scalability for further developments (vertical) or the addition of new users (horizontal).

Tech stacks and data ecosystems are some of the many technological services available for technological application development and maintenance. These systems allow the development to build and implement an application or service in a centralized environment where all of the tools (development, data ingestion and treatment, and AI processes) can work cooperatively to each other.

In this respect, a use case could involve all of the steps related to data treatment, starting from edge data collection, data processing, data analysis and input for later in situ solution making. This highlights the necessity of a robust tool infrastructure capable of addressing each of the processes.

For the FRACTAL scenario, it seems important to keep the architecture as modular as possible and, for this reason, **containers and microservice** technologies could be studied as possible approaches. Here, several technologies can be found, such as Docker and Kubernetes.

Also, apart from the tools mentioned on the previous section, further tool stacks could be considered for the purpose of designing the FRACTAL technological approach. An example of this could be the Apache Foundation **Hadoop Ecosystem** for including advanced analytics capabilities in the node. Some relevant tools could include **resource managers** (such as Apache YARN), that permit to control computational resources over different nodes in a cluster. This kind of technologies are able to allocate resources for executing containers over the nodes of a cluster, and some of them are exploring the approach using FPGAs instead.

These topics should contribute to the definition of the definitive tech stack. However, many research and decision-making steps should be carried out prior to achieving this. Also, the decision making in the platform (WP3 PULP, VERSAL) design will be fundamental, as the included technologies (such as programming languages) will define the ability to use these approaches. The advancements on the WPs and UCs will help to identify further necessities and naturally evolve the technological approach for the Run & Development environment.

6.7.3 Interoperability & integrations with other systems

From the software perspective, a **microservice** approach could contribute to maintain the interoperability between the modules of the solution. Ideally, some common **API** format could be defined in order to capture "FRACTAL microservice"

communication requirements. Besides, APIs could be documented in a unified manner (using tools such as doc-swagger). For building microservices, technologies such as API REST, Web Services, etc. could be used.

Also, **predefining tools** for some of the data lifecycle phases could assure integrations between systems and services. Here, interesting approaching could be found, such as using message queuing systems for sharing and enriching data between nodes and the outer world.

It is also important to consider **AI model interoperability** between systems. For example, adopting approaches for exporting ML models to PMML to assure interoperability between programming languages.

6.7.4 Continuous integration / DevOps platform

A DevOps platform (development and operations platform) is a very helpful tool for project management and milestones. DevOps platforms give developers tools and automation capabilities to perform and manage continuous delivery. They are very useful to speed up the development and to use the available tools.

Using a DevOps platform will provide an agile, flexible and easy way of releasing software and management of the project development. Different DevOps phases and platforms may be useful here, to give some examples:

- **Version control**, using a technology as Gitlab.
- **Automated build and deployments**, using for example Jenkins that provides support for building, deploying and automating a project and working on continuous integration.
- Functional and non-functional testing, such as Junit.
- **Monitoring**, such as Naggios.
- **Provisioning and change management**, such as Docker.

6.7.5 Use case needs for run & development environment

UC1: Improving the quality of engineering and maintenance works through drones	
AI application	1) "Supervision of critical structures as bridges or viaducts, where images of the structural status will be collected through the use of UAVs, systematizing the visual inspection in near-real-time to detect failures and cracks in the concrete surface." 2) "Monitoring of both workforce and machinery within a construction area, by deploying a WSN that provide information about the status and location of the workers in real time".

Development environment	Supposing the nodes are built on a Linux OS.
Programming languages	To be defined once the platform definition is clear. Supported languages must be known, being Java and Python probably necessary for data management and AI model development.
Data management	Hadoop Ecosystem tools support is recommended (Java support required) to obtain distribute execution and management of data.
AI model deployment	TensorFlow.
Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous Integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker, ...)

Table 49 – UC1 run & development environment needs

UC2: Improving the quality of automotive air control	
AI application	Predictive maintenance of the components in an automobile engine air-path.
Development environment	LinuxOS required for UC2.
Programming languages	To be defined once the platform definition is clear. C++ compiler is a must, and Python required for TensorFlow development.
Data management	N/A
AI model deployment	TensorFlow framework models required.
Interoperability & integration	Although more information is required, the edge node is expected to have container support.

Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 50 – UC2 run & development environment needs

UC3: Smart meters for everyone	
AI application	A low-cost machine-vision based application to read conventional meters.
Development environment	Linux is feasible, but lighter OS like littleKernel, Zircon, Nuttx or FreeRTOS are preferable.
Programming languages	To be defined once the platform definition is clear.
Data management	N/A
AI model deployment	No specific requirement for AI operation.
Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker, ...)

Table 51 – UC3 run & development environment needs

UC4: Low-latency Object Detection as a generic building block for perception in the edge for Industry 4.0 applications	
AI application	A machine vision-based object detection and recognition algorithm in form of a Low-Latency Object Detection (LLOD) building block.
Development environment	The LLOD shall be able to run application on top of the system software that will control its operation. • The RISC-V processor shall run 64-bit Linux operating system.

Programming languages	To be defined.
Data management	N/A
AI model deployment	TensorFlow/Caffe/Darknet frameworks shall be supported by the edge node for generation of the inference.
Interoperability & Integration	Although more information is required, the edge node is expected to have container support.
Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 52 – UC4 run & development environment needs

UC5: Increasing the safety of an autonomous train through AI techniques	
AI application	On-board AI-enabled computing platform for autonomous train operations.
Development environment	The board shall have Linux OS.
Programming languages	The board shall have C++ compiler. The HW accelerators should be programmed with OpenCL.
Data management	N/A
AI model deployment	The HW accelerators shall be compatible with TensorFlow's framework outputs (nice to have).
Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 53 – UC5 run & development environment needs

UC6: Elaborate data collected using heterogeneous techniques	
AI application	A smart totem is equipped with smart sensors and actuators like, for example cameras that collect data and implement AI based content analysis providing output and actuations.
Development environment	The Node shall have Linux OS such as Ubuntu or Petalinux.
Programming languages	The Node shall have a C++ compiler and related standard libraries.
Data management	Not specified
AI model deployment	The Node shall be able to execute TensorFlow-Keras framework models, or other standard Machine Learning APIs. • The HW accelerators shall be compatible with TensorFlow-Keras framework outputs. • OpenCV Library.
Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 54 – UC6 run & development environment needs

UC7: Autonomous robot for implementing safe movements	
AI application	The "Smart Physical Demonstration and Evaluation Robot" (SPIDER) is an autonomous robot prototype. Within this use case, the Cognitive Edge Node developed in FRACTAL will be integrated in the autonomous robot SPIDER and evaluated against its applicability for performing computationally intensive relevant vehicle functions of variable complexity at the edge of the network (near the source of the data) while still being able to guarantee extra-functional properties (dependability, timeliness) for preserving safety- and security operational behaviors.”
Development environment	**Linux** - Main functions of the VAL-UC7 shall be implemented on a Linux platform. Most suitable distributions would be Ubuntu 16.04, Ubuntu 18.04 or derivates.

Programming languages	C++ or Python API OpenCV Library (not mandatory)
Data management	Not specified.
AI model deployment	**C++Suite** - Main functions of the VAL-UC7 shall be implemented with C++ on Linux. Thus, a compiler, and a debugger are necessary and not mandatory a profiler. The compiler needs to support at least C++11 functions.
Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 55 – UC7 run & development environment needs

UC8: Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse	
AI application	Handling, storage, and retrieval of warehouse goods by automated shuttles are optimized using Artificial intelligence techniques. AI will also optimally organize and analyze the masses of generated data, in order to improve the warehouse throughput. The automated shuttle systems shall operate as agents of swarm intelligent system to improve its reliability. To eliminate the need for a central coordinator in which communication failures could de-stabilize the system. Real-time Information (e.g. diagnostics, battery health, task) hosted on the shuttle operation are registered in the AI database (Big data).
Development environment	Linux with RT Patch as Operating System
Programming languages	No languages specified.
Data Management	Not specified
AI model deployment	To allow predictive maintenance features to be developed, machine learning is required in order to predict failures of certain parts and devices.

Interoperability & integration	Although more information is required, the edge node is expected to have container support.
Continuous Integration (DevOps)	Note that this is a proposal, other platforms or technologies may be chosen. But compatibility with DevOps approach tools should be contemplated (Git, Jenkins, docker...)

Table 56 – UC8 run & development environment needs

Proposal:	
Development environment	The platform should support Linux OS, although other choices may be made by any UC if required.
Programming languages	C/C++ should be included, and also including Java and Python is recommended to operate ML/DL/advanced analytics tools.
Data management	Whenever data management is required, distributed computing tools (such those from Hadoop Ecosystem) may be available/supported.
AI model deployment	Most common libraries and tools support, TensorFlow, Pytorch, etc. Also support/availability for a model deployment technology such as MLFlow is recommended.
Interoperability & integration	Container support is recommended.
Continuous Integration (DevOps)	Compatibility with DevOps approach tools should be contempled (Git, jenkins, docker...)

Table 57 – Proposed requirements for run & development environment

6.8 Requirements flowing down to WP3

Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements flow down to the FRACTAL node building blocks by WP3 (hereafter referred to as the *"WP3 platform"*). • The WP3 platform shall support a POSIX operating system [VERSAL/PULP]   ○ The WP3 platform should support multithreading. • The WP3 platform shall support a C/C++ runtime (versions TBD). [VERSAL/PULP]

• The WP3 platform shall support Tensorflow and Tensorflow Federated runtime libraries (versions TBD). [VERSAL/PULP] • The WP3 platform shall support Python libraries (versions TBD). [VERSAL] • The WP3 platform SHOULD support ONNX standard (version TBD). [VERSAL/PULP] • The WP3 platform shall support Java Runtime (to allow distributed execution management technologies) (version TBD) [VERSAL/PULP] • The WP3 platform may support MLFlow or other ML lifecycle management platform [VERSAL/PULP]

Table 58 – WP5 requirements flowing down to WP3

7 Mutable and fractal communications

This section relates to objective O4 and pillar 4 and it describes the communication techniques addressed in WP6.

The goal of Mutable and fractal communications in FRACTAL is to design, develop and deploy the FRACTAL system engineering framework considering a microservices and containers-based software implementation. It consists of: (i) a processing platform at the edge with connection to different IoT devices and cloud platforms; (ii) and an edge controller infrastructure (in the cloud) to manage and control the edge nodes update and operation. This solution will follow a fractal configuration improving the scalability from Low Computing to High Computing edge node.

In the context of communication techniques, the evaluation of the ML model deployment strategies is essential for the FRACTAL node analytics platform based on the ML algorithms and the targeted future Service Level Agreements (SLAs). This will lead to the design and implementation ML model deployment framework for edge analytics. In FRACTAL node analytic tasks are performed in real-time, which poses additional requirements to the deployment framework of ML microservices in a secure environment. Realtime analysis at the edge strives interfacing between the platforms and network, QoS management, network management with SDN, and the resilience of the platforms in terms of resource and security. Currently, ML applications communicate with the top-level scheduler through a resource request/release system in the state-of-the-arts platforms. The challenge is that there is no well-defined existing protocol for allowing schedulers or applications to optimize or scale down the resource utilization.

Requirements flowing down to WP3 are described in the section 7.5.

CPS Communication Framework

CPS Communication Framework for the connection of a wide range of IoT devices and well-known cloud platforms.

FRACTAL will consider novel communication (i.e., 5G) and storage techniques enabling network scalability following a fractal configuration. Besides OT and IT-oriented communication middleware, this pillar also encloses the distributed resource management and orchestration techniques that are necessary to drive the distributed FRACTAL node infrastructure.

FRACTAL edge computing for data processing

Optimized data (video and audio) processing based on AI approaches. Such processing will be executed on the edge in the heterogeneous applicative domain. Beside the edge processing, the wireless communication capabilities will be evaluated through delay and processing point of view as heavy data transmission will cause additional delay also to the cellular network side.

Communication protocol implementation

Surveillance of critical industrial and logistic environments. Usage of the low power communication protocol for further localization system integration.

Test platform for 5G M2M communication

A cellular radio modem is integrated into a small set of FRACTAL IoT nodes and their operation is tested and verified at the OULU 5G Test Network. Depending on the use case, the most suitable radio is selected for a node and several measurements will be executed, i.e., delays, throughput and power consumption. The variety of wireless technologies is available in 5GTN portfolio from 5G to NB-IoT and LoRa technologies, on top of the different measurements tools that are in the use of experiments.

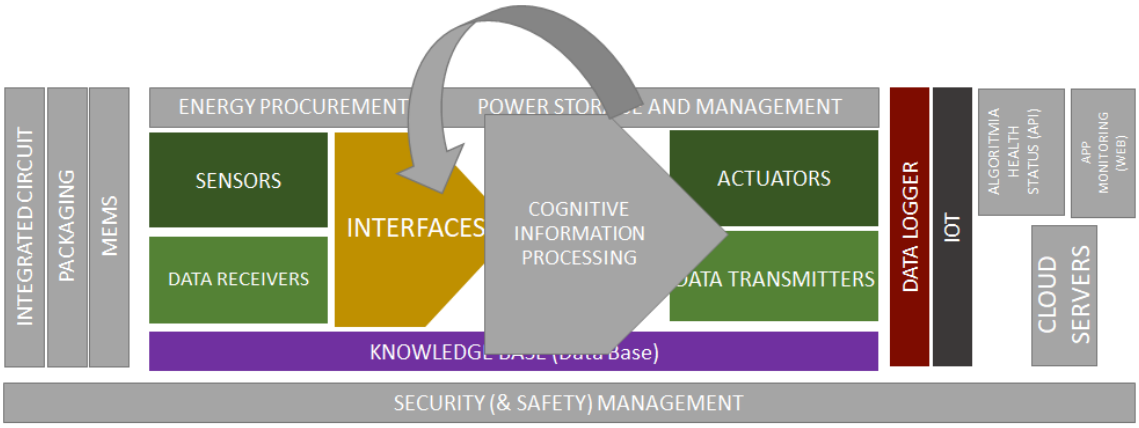

Figure 22 – Blocks from the Cognitive System to adapt for guaranteeing pillar 4

7.1 Edge node design and implementation

This section is focused on providing a requirement description of the edge node design and implementation.

This requirement set has a purpose to develop and deploy the necessary edge computing infrastructure. For this purpose, a preliminary analysis of existing open-source edge computing-based software platforms, such as e.g., Apache Kura, EdgeX Foundry, StarlingX, OpenEdge KubeEdge or MS IoT Edge etc., must be done in order to select a reference implementation. The required set of software components and tools useful for the FRACTAL engineering framework should be selected and updated to enable core functionalities such as e.g., application provisioning and scheduling, application containerization and deployment or dynamic balancing of workload to the available infrastructure resources. In addition, the architecture should be based on to core microservices for common operation and the scheduler; components developed in WP5 for including specific processing capabilities depending on the

domain will also be updated to be deployed within the FRACTAL engineering framework.

As a set of requirements, three channels should be defined:

- The interconnection between the edge infrastructure and IoT devices (e.g., sensors, actuators)
- The interaction with the cloud; and
- Internal communication between the required software components.

New components for enabling the communication with different IoT devices using standard protocols (e.g., KNX, Modbus, BACnet, OPC UA, etc.) or proprietary protocols from the use cases within the FRACTAL project should be developed and deployed inside the engineering framework. Special attentions should be paid to the design and development of wireless communication interfaces taking into account reliability and real-time requirements. To make it easier to create intelligent components, as a requirement a device abstraction layer could offer a unified interface for devices, regardless of the device type or connectivity protocol and without the need-to-know details of a specific protocol. A generic data model, as well as a common interface to communicate with the cloud, should be defined as well as implemented and deployed in order to guarantee interoperability among different data sources, the edge and the cloud. These two channels should consider wired connectivity (based on Ethernet) and wireless connectivity (based on 5G cellular protocol if technically possible or, at least, 4G protocol). Furthermore, analysis and evaluation of IT and OT convergence in real-time requirements will be performed. As a requirement, different communication alternatives in terms of architecture (orchestration or choreography) and communication protocols (e.g., HTTPS, Kafka, AMQP, etc.) should be analyzed and chosen for microservices' internal interaction.

Requirements for appropriate mechanisms to support remote monitoring, resource management and dynamic reconfiguration of the edge nodes should be set. These key capabilities will enable the FRACTAL paradigm in a heterogeneous distributed infrastructure such as the one considered in this project. In this sense, functionalities for remote monitoring and management of the FRACTAL nodes should be provided, allowing nodes to remotely start and stop applications, as well as deploying and uninstalling software components. Moreover, automatic mechanisms to dynamically allocate the available distributed resources to the demanding applications will be provided. This feature is key to supporting the desired levels of availability and fault tolerance, whilst ensuring the scalability of the overall FRACTAL system.

7.2 Edge center controller infrastructure

This section is focused on providing a description of the edge controller infrastructure required for the use case’s correct performance.

The design of the edge control infrastructure is closely related to the technological election of the components for the FRACTAL nodes, in terms of the actual tools that will be used. As this is a work in progress, we will describe in this section a **theoretical approximation** to the edge controller infrastructure, specifying the

functionalities needed to be accomplished to ensure the FRACTAL ecosystem correct behavior. The actual implementation will be completely defined as the WPs advance. However, and to provide some examples of the technologies that could be used here, we will make the assumption that the FRACTAL nodes will have a Linux based OS.

Similarly, to the separation of environments previously proposed on section 0, which proposed to distinguish between the *Run Environment* and the *Development Environment*, the edge infrastructure will be divided into two main **environments**. This will encompass a central platform (i.e., machine/cloud for the administration of the nodes/services) from which the actual nodes will be governed. This aspect highlights the necessity for a set of interactions between the FRACTAL nodes and the edge data centers which are specified below.

The main objectives of the edge center controller are closely related to the **DevOps** methodological approach, which for instance includes real time monitoring, artifact deployment and version control. Also, control over the edge is related to data governance, management and alarm tracking. Real time monitoring occurs through the local control of the different FRACTAL nodes, which are also interconnected. For this reason, robust and secure connections must be established in order to ensure distributed services performance.

It is important to keep in mind one of the objectives of the project, which is the **interoperability** of the nodes, and the ability they will have to join strengths in order to build a larger entity. From this perspective, one could think about different layers of nodes, where we will find nodes closer to the data origins dealing with data ingestion services, and distributed applications running over several nodes as we get further away from the source. This association of nodes to build larger distributed services is closely related to the phase of the data processing (ingestion, storage, etc.) and the demands (in terms of volumetry) of data origins. From this point of view, it is important to consider modular architectural designs, based on containers and microservices.

- **Edge data lifecycle: ingestion functionalities**

Primary services over the data lifecycle should have the ability to operate distributed, allowing both horizontal and vertical scalability.

The data ingestion process should take into account the varying data origin and types. At this point, the node should provide several functionalities, such as real-time analysis, model prediction, storage, and intercommunication with other services in general.

These features could align with an ETL (Extract, Load, Transform) following the data-in-motion approach. Several tech stacks are already available to accomplish this task. An example being the Apache Foundation **Hadoop Ecosystem** (Cloudera stacks) for including advanced analytics capabilities in the node, which also allows easy scalability and high-availability. More concretely, for instance the tool Apache Nifi assures these functionalities, and aligns with the edge computing paradigm through the MiNiFi agents, where edge control mechanisms are included.

- **Characteristics of the communication network and protocols**

There exist multiple approaches to capture the communication requirements of an edge computing scenario. The FRACTAL nodes as defined in the use case would be IoT devices interconnected to each other and to the FRACTAL edge center, which would ultimately be connected to the cloud. The aim of the FRACTAL approach should be focusing on light reliance on the cloud, so most part of data treatment must be done in the nodes themselves, assuring fast analysis and response capability.

However, the edge controller should be **aware of the status** of the different nodes under its govern. This ranges from the simple knowledge that the device is there and up (heartbeat exchange), to more complex tasks such as deployments, version control, etc.

IoT communication protocols are required, some examples could be C2, MQTT and CoAP. Further research will help to elucidate more precisely the characteristics and requirements of the IoT devices, so a protocol choice can be made.

- **Basic functionalities and core services**

One of the innovative aspects the FRACTAL project revolves around is modularity (keeping the architecture divided into independent modules which are able to work separately). Microservices emerge as a valid approach for this problem, through the modularization of the classical monolithic server-side application into independent services[9]. These services are therefore loosely-coupled as do not depend on each other for their performance, providing flexibility to the whole architecture.

Although further research on the use case's scenario is still required before specifying which functionalities will be required, an edge scenario should presumably provide: metadata storage and treatment, alerts and notifications, logging and auditing mechanisms, automatic updating tools, failure or error monitoring and prediction, self-maintenance, or a registry & authentication system, to mention some.

The integration and interoperability of such microservices, however, must be addressed. As microservices can be developed following different programming strategies, they need APIs to communicate, which must also be specified.

It is important to keep in mind approaches such as containers, which provide an implementation technology for microservices. These are represented by lightweight images and work as a self-contained and independent application which, after deployed, is fully isolated[10]. This container configuration requires a management tool for the automation and coordination during the container lifecycle. Technologies such as **Kubernetes** are widely used, and also with resource management tools such as YARN or Mesos.

[9] Leppanen, T., Savaglio, C., Loven, L., Jarvenpaa, T., Ehsani, R., Peltonen, E., … Riekki, J. Edge-Based Microservices Architecture for Internet of Things: Mobility Analysis Case Study. 2019 IEEE Global Communications Conference (GLOBECOM). (2019).

[10] Pahl, C., & Lee, B. Containers and Clusters for Edge Cloud Architectures -- A Technology Review. 2015 3rd International Conference on Future Internet of Things and Cloud. (2015).

7.3 Validation of the edge computing architecture

Validation of the edge computing software engineering framework require functional testing and non-functional testing at different levels in the systems:

- At a **component level**, validating the functionality and data flow of the different software components (i.e., microservices) that comprise the edge computing framework.
- At a **system level**, validating the overall system end-to-end operation, in terms of functionality and performance, along with blocks of components interaction and system integration with the final ecosystem of each use case.

Considering the basic functionalities of the FRACTAL edge node, at a component level, the validation of the edge computing framework needs at least to address the following aspects:

Functional tests:

- Validation of sensor compatibility and use case-specific testing, depending on the final purpose of the sensors. Functionality and data flow of components in charge of handling data flows from sensors or external devices must be validated.
- Validation of the integration with different types of devices and platforms through existing interfaces, by means of different IoT data protocols. Adaptation to framework internal data model must be also validated.
- Validation and testing of data filtering, processing, and analytic components to ensure that the system can operate as required.

Non-functional tests:

- Validation of the performance of the device and sensor communication with the edge computing node, validating the latency and quality of service levels as the number of devices connected to the system increases.
- Validation of the scalability of the edge node, ensuring that it can handle and process requests without affecting latency as the system scales up in the number of connections with IoT data platforms or interfaces being used to communicate.
- Load and stress testing to ensure that the data filtering and processing performance is not affected by the number of devices and connections.

In addition to the validation of the main components of the edge computing framework, the validation at a system level is also needed:

Interoperability:

- Validation of the interoperability across all the integrated components to ensure the compatibility and data interaction with other devices.

Functional tests:

- Validating the basic system functionalities as software component management and update, dynamic reconfiguration and remote monitoring, as well as validating data flows between the system and its connected ecosystem.

Non-functional tests:

- Performance, reliability, and scalability testing to validate data flow and ensure that performance remains unaffected by variations in scale.
- Validation of system latency between main blocks of components, ensuring that this speed remains unaffected by the addition of new blocks into the edge computing framework.

Methodology for validation of microservice systems

Microservices use well-known technologies, such as RESTful APIs or message queues, for which the software industry already has well-established testing tools and best practices. The main challenges with microservice architectures are, on one hand, the large number of services that can compose an application, along with the dependencies between them, and on the other hand, that each microservice needs to function properly despite other microservices that they depend on are unavailable or not responding properly.

In this context, continuous validation becomes essential for building, testing, and deploying containerized microservices to ensure that the overall system operates in accordance with its objectives, to automate the deployment process of new versions of the artifacts and to reduce the risks of releasing updates and unexpected downtime due to failures and irregular operating conditions.

In order to enable an automated validation of the system operation, it will be required that every single software component (i.e., microservice) provides a definition of its external interface, along with an automated unit-testing of its source code.

Integrated in CD/CI tool, such as Jenkins, where the unit-testing will be executed, and if it is successful, the code will be packed into a container and a set of container self-test will be provided to ensure that all the components of the container are working correctly and the interface will be also validated to be as the one previously defined.

If all tests executed on a microservice development machine, then the microservice will be deployed on a test environment, based on the final target architecture, where functional and load integration testing will be performed, checking also possible issues caused by orchestration if used, life cycle management and communications.

As long as the microservice passes all tests on the target test environment, then it is safe to deploy it to the production environment.

7.4 Integration, testing and validation of standalone communication sub-systems

This task will focus on communications to and from the Fractal nodes. The activities include integration and testing the FRACTAL nodes' wireless communications capabilities with a stand-alone communications subsystem.

The assumption in the start of the designs is that communications modules will be connected to the FRACTAL node by USB-C, USB3.0 or Ethernet, thus, both wired and

wireless communications are utilized, depending on use cases and requirements, such as power consumption and data rates. According to the initial requirements in the use cases, there are several nearly opposite demands for communication to be fulfilled, i.e., from 300 Mb/s to only a few bytes per day and from node-to-node direct communications to over-the-air provided updates.

Due to the different requirements, several wireless means will be utilized. This will include both unlicensed and licensed wireless technologies. The most popular unlicensed ones are Wi-Fi, Bluetooth and LoRa technologies as the licensed include 3GPP specified technologies from 4/5G to NB-IoT and LTE-M.

Wi-Fi can offer hundreds of Mb/s over a fairly short distance while LoRa support low power transmission over a long, up to tens of kilometers distance, range. Bluetooth can be used for mesh type of communications between the nodes.

The counter part in the 3GPP defined technologies are 4/5G which can offer up to Gb/s of data rates, while both NB-IoT and LTE-M variants support long distances and low power consumption for the use of FRACTAL nodes.

The FRACTAL nodes with its' wireless connectivity options will be tested in the 5GTN network of OULU to validate the performance in conjunction with a real-life wireless state-of-the-art network. The 5GTN as a live test network supports all the above-mentioned different technologies and it has already a wide range of devices integrated into the network. There will be pre-tests done with even wireless CAN adapters that will be used, for example, in UC 2.

The 5GTN is currently built mainly on Rel. 15 supporting HW and SW, but the network itself is constantly being updated with recently completed Rel 16. features. Different parameters that will be tested during the validation include e.g., data rate, latency, block error rate and end-to-end delays. For Rel. 16 communications jitter testing it may become important in industrial use cases with Time Sensitive Network (Industrial Ethernet over wireless) are available.

7.5 Requirements flowing down to WP3

Objective 1: **Open-Safe-Reliable and low power node architecture**
These requirements flow down to WP3. • GAIA-X:Technical Architecture compability • Dynamic adaption capability (Dynamic adaptation of cloud and edge service is of key importance to enable the seamless integration of cloud infrastructure and edge equipment) • Efficient rerouting of the information from IoT devices to the containers and the inter-container communication • Flexible online reconfigurability • Integration and interoperability of microservices • API management

• MLOps (related to DevOps) - Automated orchestration and certification (to certify the state of the edge node from firmware state to software layer, e.g., ML models in use are accurate) on the edge including Commission/decommission of the edge and deployment framework of ML microservices in a secure environment • GPU enablement • Support for controllability and observability • Multi-user massive MIMO (multiple-input and multiple-output) enablement • Targeted future Service Level Agreements (SLAs) definitions • Unlicensed and licensed wireless technologies enablement • The WP3 platform shall support implementations for Command and Control (C2) protocols

Table 59 – WP6 requirements flowing down to WP3

8 Node architecture and building blocks

This section relates to objective O1 and pillar 1 and it describes the architecture and building blocks addressed in WP3. Therefore, it provides the requirements of those hardware and software components upon which the remaining layers of the stack and the UCs build. Hence, technologies in this chapter lay at the bottom of the FRACTAL stack, as depicted in Figure 23.

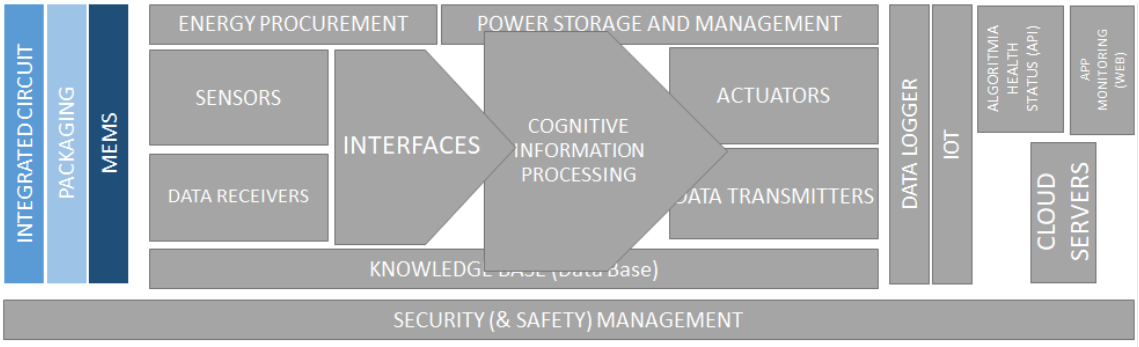

Figure 23 – Blocks from the Cognitive System for pillar 1

This chapter details, for both baseline platforms, i.e., Xilinx VERSAL and RISC-V based ones, how requirements from the UCs, as well as from WP4/5/6 are captured to some good extent, together with the node and building block requirements for each platform, classifying them across hardware and software parts.

8.1 VERSAL-based node

The Xilinx VERSAL System on Chip (SoC) is expected to be deployed as part of the VCK190 Evaluation Kit board, which provides support for several I/O interfaces and memory devices. The requirements from the UCs on the VERSAL hardware platform often related to the components in the SoC or the board, which, based on our analysis, should be supported. Those requirements are detailed in the following table for the different UCs building on the VERSAL platform identifying what components should capture them.

UC	REQUIREMENT	MEANS
UC1	VERSAL node	SoC
UC2	Non-volatile memory	Board
UC2	≥ 4 cores (parallel processing)	SoC (2+2 Arm cores)
UC2	≥ 16 GB RAM	Board
UC2	10 MOP/s to 1 GOP/s processing performance	SoC
UC4	Low Latency Object Detection (LLOD): CPU and hardware accelerator	SoC

UC4	LLOD: Powerful enough to run the inference without any stall as the video streams flows.	SoC
UC4	LLOD: Re-configuration of the hardware for different inference models	SoC
UC5	≥ 4 cores	SoC (2+2 Arm cores)
UC5	Handle with multi-threading applications	SoC
UC5	≥ 60 GFLOPS	SoC
UC5	≥ 16 GB DDR RAM	Board
UC5	HW acceleration	SoC
UC5	HW acceleration based on Graphics Processing Unit (GPU)	SoC
UC6	≥ 1 High Definition (HD) camera	Board
UC6	≥ 1 microphone	Board
UC6	HW acceleration	SoC
UC6	HW acceleration based on GPU	SoC
UC6	Modular and scalable architecture	SoC
UC6	Store data locally in a secure manner	Board
UC6	Control an interactive touchscreen display	Board
UC6	Control an audio speaker	Board
UC6	Hardware for convolutional neural networks applications	SoC
UC8	≥ 2 cores	SoC (2+2 Arm cores)
UC8	Handle with multi-threading applications	SoC
UC8	≥ 800 MHz	SoC
UC8	≥ 32 GB eMMC or similar memory	Board
UC8	≥ 4 GB DDR RAM	Board

Table 60 – UC requirements for the VERSAL hardware platform

Similarly, the following table reports a number of requirements from the UCs related to the software support of the VERSAL platform. Those generally relate to whether the platform supports Linux, math libraries and Open Computing Language (OpenCL).

UC	REQUIREMENT	MEANS
UC1	Linux	SoC+board
UC2	Linux	SoC+board
UC2	Math libraries	SoC+board
UC4	Linux 64-bits	SoC+board
UC5	Linux	SoC+board

UC5	HW accelerators should be programmed with OpenCL	Vitis
UC6	Linux	SoC+board
UC6	Monitoring system able to measure response time of tasks	Linux and Vitis
UC8	Linux	SoC+board

Table 61 – UC requirements for the VERSAL software platform

All the requirements captured from WP4/5/6 flow down for VERSAL software/hardware platform as listed below:

WP	REQUIREMENT	LOCATION/FEATURE
WP4	The system should produce decisions at least at 10 Hz rate, possibly at a 30Hz rate.	SoC (RPU)
WP4	Time-triggered communication shall be supported for interaction between Fractal nodes	SoC
WP5	The WP3 platform SHALL support a Portable Operating System Interface (POSIX)	Linux, Real Time Operating System (RTOS)
WP5	The WP3 platform SHALL support a C/C++ runtime (versions to be defined).	Linux
WP5	The WP3 platform SHOULD support multithreading.	
WP5	The WP3 platform SHALL support Tensorflow and Tensorflow Federated runtime libraries (versions to be defined)	Linux
WP5	The WP3 platform SHALL support Python libraries (versions to be defined)	Linux (Python with binding for Xilinx RunTime)
WP5	The WP3 platform SHOULD support Open Neural Network Exchange (ONNX).	Linux + Vitis
WP6	Flexible online reconfigurability	SoC
WP6	Support for controllability and observability	SoC
WP6	Unlicensed and licensed wireless technologies enablement	SoC + board extensions

Table 62 – WP4/5/6 requirements for the VERSAL software /hardware platform

8.1.1 Hardware requirements

The VERSAL architecture combines different engine types with a wealth of connectivity and communication capability and a network on chip (NoC) to enable seamless memory-mapped access to the full height and width of the device. Intelligent Engines are artificial intelligence Engines for adaptive inference and advanced signal processing compute, and DSP Engines for fixed point, floating point,

and complex MAC operations. Adaptable Engines are a combination of programmable logic blocks and memory, architected for high-compute density. Scalar Engines, including Arm Cortex-A72 and Cortex-R5F processors, allow for intensive compute tasks.

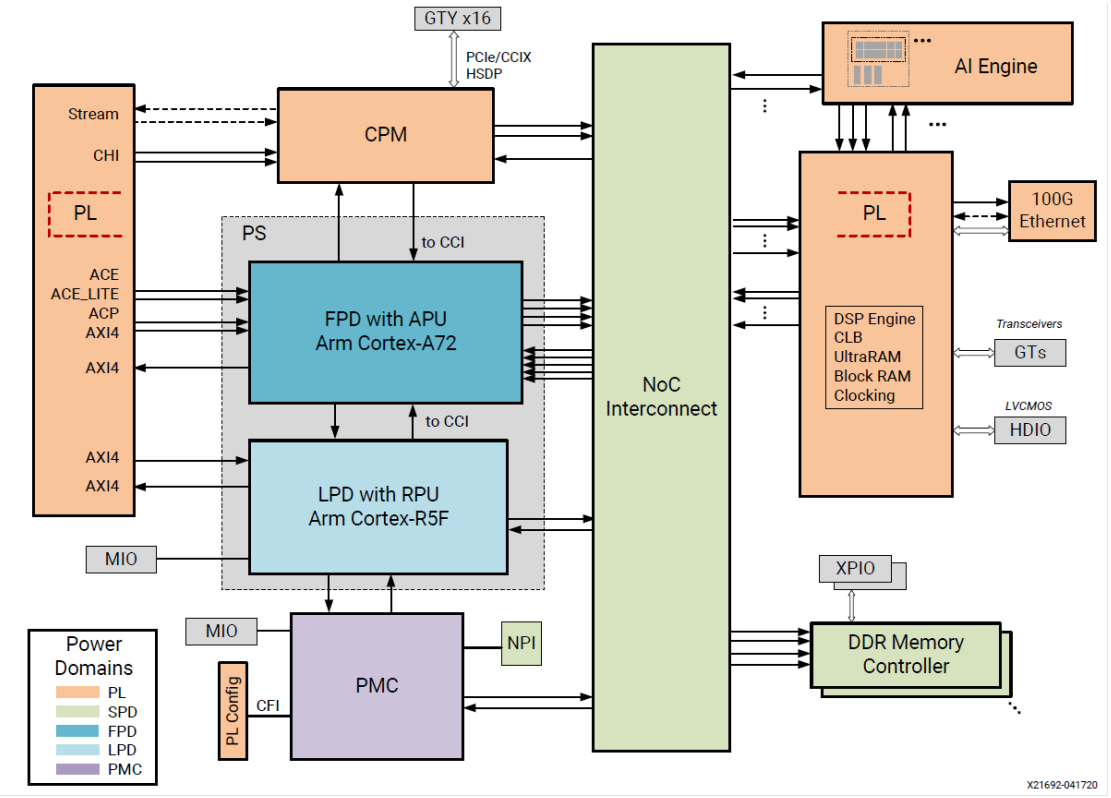

Figure 24 – VERSAL SoC schematic

Following subsections will detail the characteristics of the hardware components available in VERSAL Adaptive Compute Acceleration Platform (ACAP), especially those related to safety, security and low power and cognitive awareness, whose particularities will be described separately.

8.1.1.1 Safety, security and low power management

As shown in the Figure 24, VERSAL has a centralized Platform Management Controller (PMC) that boots the device after power on reset. However, this subsystem includes other functionalities related to power management, safety or reliability. The PMC handles device management control functions such as device reset sequencing, initialization, boot, configuration, security, power management, dynamic function eXchange (DFX), health-monitoring, and error management. Further details are provided next.

Firmware running on the Platform Management Controller (PMC) is available with code examples for user run-time power management, but basic services are provided already. Processor sleep/wake and clock frequency-reduction features (section 5.2, 8.1.2.1) would be developed along the requirements of the UCs. Solutions of essential

concepts for power analysis and run-time management on VERSAL might be supplemented with implementation in WP4.

Safety requirements can be mapped on the VERSAL platform by hardware support for concepts of partitioning and isolation. Redundancy management is provided using the Platform controllers and RPUs and Hypervisors are readily available to isolate guest OS using the built-in coherency features of the interconnect and MMU features.

Along with these features the modular firmware services (section 5.3, 8.1.2.1) that are targeting the new VERSAL platform still need review for fit for particular designs. Insights should become available as reference designs for these isolation concepts. This would include restrictive/coherent access for shared memories and shared peripherals and using self-test libraries (LBIST) at runtime.

Security management on the VERSAL platform leverages from Secure Booting seen in earlier Xilinx platforms, where the firmware partitions to be loaded can be controlled with secure aspects (authenticity, confidentiality). The root of booting is a hardware state machine that validates the hardware against golden hash values and evaluates the eFuses. After this, only the ROM Code Unit (the RCU) in the PMC is starting with further system checks and configuration (CDO based). It finally securely loads the PLM, which is then in charge for all further images under partial user control, like ARM TrustZone, platform firmware, AIE firmware and bitstreams.

The same security features apply to multi-partition loading of OSs, exchanging bitstreams at runtime (DFX) and AI engine firmware. Along with these principles the fallback boot handling and field update features need to be defined. One-time programmable nodes like eFuses are available but are not easily demonstrated with the development kits as this permanently restricts debug capabilities.

8.1.1.2 Cognitive awareness

Since FRACTAL involves several use cases, each of them having different computing and hardware accelerating requirements, available hardware accelerator implementation options need to be considered. The characteristics and architecture of the different compute engines available in VERSAL heterogeneous platform are described in this subsection.

In general, the choice of acceleration hardware, whether PL or AI Engines, depends on the type of algorithm and data ingress and egress paths. Scalar Engines provide complex software support. Adaptable Engines provide flexible custom compute and data movement. Given their high compute density, AI Engines are well suited for vector-based algorithms. Based on this, it must be considered that:

- Scalar processing elements like CPUs are very efficient at complex algorithms with diverse decision trees and a broad set of libraries. However, these elements are limited in performance scaling. Application control code is well suited to run on the scalar processing elements.

- Programmable logic can be precisely customized to a particular compute function, which makes them best at latency-critical real-time applications (e.g., automotive driver assist) and irregular data structures (e.g., genomic sequencing). However, algorithmic changes have traditionally taken hours to compile versus minutes.
- The AI Engine processors deliver more compute capacity per silicon area versus PL implementation of compute-intensive applications. AI Engines also reduce compute-intensive power consumption by 50% versus the same functions implemented in programmable logic (PL) and also provide deterministic, high-performance, real-time digital signal processor (DSP) capabilities. Because the AI Engine kernels can be written in C/C++, this approach also delivers greater designer productivity. Signal processing and compute-intensive algorithms are well suited to run on the AI Engines.

VERSAL specific AI Engines are an array of very-long instruction word (VLIW) processors with single instruction multiple data (SIMD) vector units that are highly optimized for compute-intensive applications, specifically DSP and AI technology, such as machine learning. These AI engines provide multiple levels of parallelism including instruction-level and data-level parallelism.

FPGA accelerator introduction

Hardware accelerator is a specialized component of SoC to perform computer vision tasks. The component is a field programmable gate array (FPGA) solution consisting of multiple Processing Elements (PE) for supporting computation parallelism at fine-grain granularity. A pre-trained Convolutional Neural Networks (CNN) for image processing is usually deployed on top of the accelerator. The key advantage of the accelerator is the utilization of the wide concurrency that CNN inference exhibits.

The architecture of the accelerator is devised in two directions:

- The architecture of PEs and
- The architecture of the memory hierarchy

PE is a small, specialized component that performs CNN operations. It consists of a multiply accumulate unit, an adder unit, a unit for non-linear Leaky ReLU operation and a pooling unit. In addition, the PE has a control unit that drives the data through operational units. The PEs are interconnected between to pass the data from one to the other.

The memory is devised in form of a hierarchy to improve timing performance and energy consumption by exploiting temporal reusability of CNN's parameters. This is achieved through small and fast buffer memories located near the PEs. While one buffer supplies the PEs with data the other one prefetches the anticipated data from DRAM and vice versa. PEs also have a local memory in the form of registers to keep the current input and output data. Each operation on PE requires at least two memory read and one memory write. If all these accesses are performed directly on the off-

chip DRAM memory the generated latency and the amount of consumed energy would make the accelerator not adequate to deal with high computation workload of CNN.

The hardware accelerator is configurable and flexible since it is an FPGA solution, thus making configurable the number of PEs as well as the size of the deployed local memories. Such an approach gives us the opportunity to have customized accelerator topology for different CNN types, while the architectural design of the PEs and the memory hierarchy remain the same.

Based on these appreciations, in the beginning of the project the main approach is not only to analyze, understand and determine the requirements coming from every UC, but also the proposed roadmap in order to achieve use case objectives. These inputs will potentially determine the required hardware development needed for providing cognitive awareness to FRACTAL node based on VERSAL platform. Reference architecture of a cognitive edge computing node with FRACTAL properties will be defined and a common repository of generic qualified components will be set up. Particular attention will be paid on providing flexible computing nodes, that are reusable by others and that efficiently support the software on providing acceleration for the learning part.

Several acceleration approaches (e.g., approximate computing on general-purpose CPUs, GPUs, custom AI/Machine Learning (ML)-oriented accelerators on Field-Programmable Gate Array (FPGA), Component Off The Shelf (COTS) AI/ML-oriented accelerators, etc.) will be considered, evaluated and compared in order to identify the best ones for the different FRACTAL nodes also with respect to extra-functional properties (e.g., timing performance, power consumption, etc.).

8.1.1.3 Integration

The VERSAL platform provides support for integration of heterogeneous compute elements with emulation and co-simulation in the Xilinx Vitis development environment. To use these features, the proper and shareable addition of FRACTAL elements like acceleration kernels in PL or the AI Engines requires packaging for this tooling. Adding such elements in the context of the tools further sets requirements for OS layer in an edge device. In the VERSAL ecosystem, services of the Xilinx runtime (XRT) are commonly used to set up and operate these accelerator components. These services restrict and define the form of the accelerators and need to be followed in the design. Also, this must be supported with insight into the application partitioning between the heterogeneous compute elements in the VERSAL devices, i.e., the type of kernel and topology, e.g., for memory resources.

8.1.2 Software requirements

8.1.2.1 Safety, security and low power management

As shown in the Figure 24 (section 8.1.1) and described in section 8.1.1.1, VERSAL has a centralized Platform Management Controller (PMC) that handles device management control functions.

Power efficient designs require usage of complex system architectures with several hardware options to reduce power consumption and usage of a specialized CPU to handle all power management requests coming from multiple masters to power on, power off resources, and handle power state transitions. In addition, there are other resources like clock, reset, and pins that need to be similarly managed.

The platform management is available to support a flexible management control through the PMC. This platform management handles several scenarios and allow the user to execute power management decisions through its framework (equivalent to what it is done in Linux, which provides basic power management capabilities like CPU frequency scaling). However, some limitations apply. Because of the heterogeneous multi-core architecture of VERSAL, individual processors can't make autonomous decisions about power states of individual components or subsystems. Instead, a collaborative approach is taken, where a power management API delegates all power management control to the platform management controller. This PMC is the key component in coordinating the power management requests received from the other processing units, and the coordination and execution from other processing units through the power management API. This framework manages resources such as power domains, power islands, clocks, resets, pins and their relationship to CPU cores, memory, and peripheral devices.

So, while it is not yet defined, the provided power management API could be used, since this platform management framework abstracts the complexity associated to administrate the power-management of a multiprocessor heterogeneous system. However, in necessary case, adaptation of the PMU unit proposed in sections 8.2.1.1 and 8.2.2.1 could be considered.

Related to security issues, those applications that require device-level security could implement boot image encryption and authentication, functionalities that are supported by VERSAL.

Complimentary safety or security features not provided natively by VERSAL platform, could make use of the PL and processing units to implement required functionalities (e.g., software module to provide redundancy or spatial and temporal partitioning management). Related sections for RISC-V platform can be read as reference.

8.1.2.2 Cognitive awareness

Regarding the software services for cognitive awareness, there are no concrete platform requirements.

8.1.2.3 Integration

Complex heterogeneous accelerators based on programmable logic potentially have hundreds to thousands of computing cores, either hard-cores (ARM complex), soft-cores (e.g., Microblaze or RISC-V in programmable logic), ASIC (e.g., the AI engine) or soft IPs (e.g., a neural accelerator in programmable logic). This might cause serious issues when all of them require accessing the shared resources, such as the DRAM memory banks via the on-chip AXI interconnect and ports. This aspect might not be captured at design time, but show up when the system is integrated, and the (benchmark or realistic) workloads are deployed onto the platform, representing a showstopper for performance in predictability. This will be analyzed in detail for the domain of, and it will be proposed a mixed hardware/software solution to mitigate contention over the shared memory and more generally in shared resources. These methodologies might involve both ad-hoc application-specific circuits, and software level artifacts, such as OS driver extensions and/or novel hypervisor architectural design.

8.2 RISC-V (PULP) based node

The FRACTAL project from the beginning decided to provide two different platforms addressing different needs. Applications that need a more mature technology and SW support and need higher performance would target the Xilinx VERSAL platform. For UCs that have lower performance requirements (closer to IoT applications) the RISC-V based open-source PULP (Parallel Ultra Low Power) platform provides a second and flexible architecture that can be tailored to applications. As part of the PULP platform, there are several different single, multi-core and multi-cluster systems. As a basic platform, FRACTAL will use the single core PULPissimo system, but UC owners will be free to use any other implementation that fits their requirements. PULPissimo and other PULP based systems have already been implemented on a variety of FPGA based platforms, any of which can be used by the UCs.

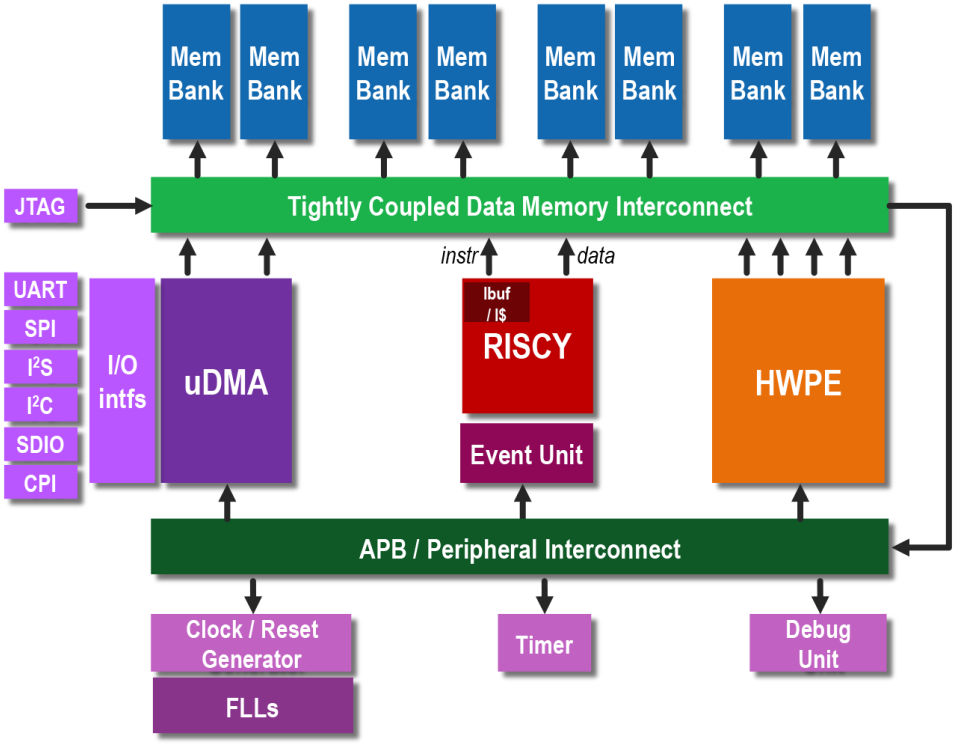

Figure 25 – PULPissimo SoC schematic

The requirements from the UCs on the RISC-V hardware platform as well as the means to satisfy them are as reported in the following table.

UC	REQUIREMENT	MEANS
UC1	PULP node	SoC
UC3	512 kB on-chip memory	SoC
UC3	2 MB off-chip memory	Board
UC3	4 MB Flash memory for weights	Board
UC3	Encrypt the program	SoC (HWcrypt)
UC3	Small form factor	SoC
UC3	Real time clock	SoC+board
UC3	UART Interface	SoC+board
UC3	Camera interface	Board
UC3	SPI, I2C, I2S, USB interfaces	SoC+board
UC3	Deep sleep currents <10uW	SoC
UC3	PULP node	SoC
UC4	LLOD: CPU and hardware accelerator	SoC

UC4	LLOD: Powerful enough to run the inference without any stall as the video streams flows.	SoC
UC4	LLOD: Re-configuration of the hardware for different inference models	SoC
UC4	Shall support M extension for multiplication	SoC
UC4	Should support "V" extension for data-level parallelism	SoC (not PULP)
UC4	Should support "P" extension for data-level parallelism	SoC (not PULP)
UC7	A FPU for double precision operations	SoC
UC7	≥ 2 cores (nice to have)	SoC
UC7	≥ 2 GB RAM	Board
UC8	≥ 2 cores	SoC
UC8	Handle with multi-threading applications	SoC
UC8	≥ 800 MHz	SoC
UC8	≥ 32 GB eMMC or similar memory	Board
UC8	≥ 4 GB DDR RAM	Board

Table 63 – UC requirements for the RISC-V hardware platform

Similarly, several requirements from the UCs relate to the software support of the RISC-V platform. Those generally relate to whether the platform supports Linux. While several systems from the PULP platform family support Linux, the support is not available for all systems at the moment. Especially symmetric multiprocessing (SMP) Linux support is not really an option for smaller scale implementations. UC owners are in discussions with ETH to discuss different options.

UC	REQUIREMENT	MEANS
UC1	Linux	SoC+board
UC3	OS littleKernel (lk), Zircon, freeRTOS or NuttX	SoC+board
UC4	Linux 64-bits	SoC+board
UC7	Linux	SoC+board
UC8	Linux	SoC+board

Table 64 – UC requirements for the RISC-V software platform

All the requirements captured from WP4/5/6 flow down for RISC-V software/hardware platform are listed below:

WP	REQUIREMENT	LOCATION/FEATURE
WP4	PMUs measuring multicore interference.	BSC performance monitoring unit for safety [*Hardware Requirements*]

		BSC safety and security software support for the performance monitoring unit (PMU) [*Software Requirements*]
WP4	Interfaces to create redundant processes which execute with some staggering.	BSC software-only diverse redundancy support for safety [*Software Requirements*]
WP4	Process/task privilege management or information obfuscation means for the PMU.	BSC safety and security software support for the performance monitoring unit (PMU) [*Software Requirements*] Support for Machine, User & Supervisor RISC-V privilege modes if a rich OS is implemented [*Hardware Requirements*]
WP4	The system should produce time-bounded decisions (reworded from "The system should produce decisions at least at 10 Hz rate, possibly at a 30Hz rate.")	Use cases can use platform features to implement this requirement
WP5	The WP3 platform SHALL support a POSIX operating system	PULP systems support POSIX based operating systems. However, applications that require heavy support from an operating system are probably not best suited for the PULP systems used in this project.
WP5	The WP3 platform SHOULD support multithreading.	There are PULP systems that support multithreading. Although this is not the main systems that are intended for UCs, UC providers are welcome to use them.
WP5	The WP3 platform SHALL support a C/C++ runtime (versions TBD).	PULP platform supports all common C/C++ development environment runtimes.

WP5	The WP3 platform SHALL support Tensorflow and Tensorflow Federated runtime libraries (versions TBD).	Tensorflow generated graphs can be efficiently mapped to PULP based systems. However, the PULP based systems are not meant or designed to replace common Linux running computing nodes. They are highly efficient specialized architectures for data centric processing.
WP5	The WP3 platform SHOULD support ONNX.	For using the LEDEL library and facilitate the interoperability of the models.
WP5	The WP3 platform shall support Java Runtime (to allow distributed execution management technologies) (version TBD)	TBD in D2.3
WP5	The WP3 platform may support MLFlow or other ML lifecycle management platform	TBD in D2.3
WP6	The WP3 platform shall support implementations for Command and Control (C2) protocols	TBD in D2.3

Table 65 – WP4/5/6 requirements for the RISC-V software/hardware platform

8.2.1 Hardware requirements

FRACTAL involves several UCs each with different computing requirements. In the first part of the project, the goal is to understand and determine the requirements from every UC that potentially could use the experimental FRACTAL node based on RISC-V cores.

The hardware requirements for the system can be divided into two categories.

a) the processor core
b) the system that includes the core, the memory and all the peripherals.

For the processor core we identify the following parameters:

- Word size (64/32 bit): RISC-V is a flexible implementation that allows cores with different bit widths. PULP platform currently has three different 32bit cores and one 64bit core.
- ISA extensions (yes/no): An attractive part of RISC-V is the ability to add custom instructions to the processors without interfering with the standard instructions. There is great freedom (with some limitations) to add such instructions, however the SW environment has to be adapted to take advantage of such extensions.

For the system we further identify the following parameters:

- Architecture (single core/many-core/multi-cluster): PULP platform is organized around systems that have different compute capabilities. For smaller applications, systems with a single processor core are available. ETH has had great success with cluster-based many-core architectures for IoT applications and for more demanding applications, multi-cluster architectures are also available.
- Heterogeneous acceleration (yes/no): For certain applications, the system can be enhanced by domain-specific accelerators. The PULP architecture has been designed to be flexible to add a series of accelerators with relative ease.
- Memory size (kB): All PULP based systems use a hierarchy of local memories, not necessarily caches. On-chip memory is a crucial parameter that determines the area (cost) and performance (speed, power) of the system. Several applications require larger on-chip memories to be efficient, and the memory requirement of the system is an important parameter to determine the suitability of the chosen platform.

Common performance parameters such as circuit area, maximum operating frequency, power consumption, energy per operation are dependent on the implementation technology and as explained in section 8.2.1.3, such parameters will be estimated depending on the chosen implementation technology for each use case.

The common platform will furthermore be enhanced as part of the FRACTAL project for safety, security and low power (section 8.2.1.1 below) and cognitive awareness (section 8.2.1.2) whose parameters will be described separately.

8.2.1.1 Safety, security and low power management

Performance monitoring unit for safety

A multicore interference-aware Performance Monitor Unit (PMU) will provide safety support for verification, validation and deploying safety measures during operation. Additionally, it will allow secure access to PMU information. The PMU is an advanced statistical unit including controllability and observability channels that will be used to deal with timing interference concerns in safety-critical real-time applications on top of the PULP-related SoCs.

The PMU can be divided in the following components:

- Maximum-Contention Control Unit (MCCU): Provides control and quality of service to the shared resource, holding one quota counter per each core, which is set to the maximum number of cycles that such core can use that shared resource. It allows implementing safety measures on top.
- Request Duration Counter (RDC): Are a series of registers which monitor and measure each event duration when trying to access a shared resource, recording the maximum latency observed. It helps timing verification (e.g., Worst-Case Execution Time estimation).
- Cycle Contention Stack (CCS): Provides interference information about when a contender tries to access a shared resource that is being used by another

contender. With this information, a contention stack is generated to measure the amount of interference created by each contender. It provides diagnosis capabilities for timing validation and during operation.

The PMU is Advanced Microcontroller Bus Architecture compliant (AMBA-compliant), all registers can be accessed to be read or written through Advanced eXtensible Interface (AXI) or Advanced High-performance Bus (AHB) interfaces. The PMU is fully customizable and can be tailored to a wide variety of multicore architectures, including those based on RISC-V architecture. The PMU's control and parametrization will be done by software with an appropriate library.

Safety and security support for the Interconnect

Implementation of hardware support for safety and security for the interconnection architecture or network-on-chip (NoC) in the RISC-V platform requires including support to improve isolation capabilities of the RISC-V multicore platform to achieve predictability, fault containment and security. Developments will be provided on top of an AXI compliant interconnect. This interconnect will be used to interconnect RISC-V cores and accelerators with memory devices. The NoC will also incorporate debugging information to allow the PMU to obtain accurate timing information for the SoC components that are connected to the NoC.

Suggested safety features

Use cases that plan to integrate on PULP platform have not expressed strong requirements related to safety features, either because they are in a prototype phase or because they will address safety at the system-level.

Therefore, this section includes suggestions for additional features, in a "bottom-up" fashion, that could be integrated in the processor cores or the SoC for future needs.

These features relate to the following generic safety requirements:

- Availability: readiness of the system when needed and ability to recover from a failure
- Predictability: ability to predict how a system will behave with given inputs
- Real-time and controlled timing execution: ability to meet hard deadlines
- Reliability: ability to resist to wear-out and harsh environmental conditions
- Integrity: ability to retain data uncorrupted
- Explainability: capability to explain why a system is behaving as it is (pre-requisite for predictability)
- Observability: ability to observe the detailed system state
- Simplicity (limited complexity): ability of a system to be understood end-to-end in a concise manner

Processor core-level safety features (suggestions)[11]:

- Ability to invalidate/flush the cache and Translation Lookaside Buffer (TLB) to return to a known state (predictability)
- Ability to define non-cacheable address chunks for peripherals (predictability)
- Ability to disable caches (higher predictability at the expense of performance)
- Ability to disable predictions, such as branch predictions (higher predictability at the expense of performance)
- Availability of local memory, such as scratchpad or locked lines/set in the L1 cache (strong real-time and higher predictability for critical processes)
- Performance counters:
    - Cache usage (hit, miss…)
    - TLB usage (hit, miss…)
    - Exceptions
    - Executed instructions
    - Branch predictions
    - Timestamp (number of clock cycles)
- MMU and privilege levels (to implement a Linux-like OS or a hypervisor for safety-critical applications that require one)

Some of the above core-level features will be developed for the OpenHW Group's open-source CVA6 core (32/64-bit Linux-capable RISC-V core derived from PULP 64-bit ARIANE core developed by ETH Zürich).

We can have several processor cores in the SoC. Some applications may choose to host safety-critical processes on a dedicated core or several cores. Therefore, the following features are suggested:

SoC-level safety features (suggestions):

- Ability to dedicate core(s) for safety-critical processes (explainability and simplicity, increased protection against interferences)
- Primitives for spatial partitioning, e.g., cores that do not include a Memory Management Unit (MMU) shall not be able to read from/write to critical address spaces. This can be implemented through a memory protection unit, also called PMP in the RISC-V context (integrity)
- Configurable QoS with fixed or bounded latency between safety-dedicated core(s) and main memory / interface controllers through the interconnect (predictability).
- The QoS control can be:
    - static (defined at power up)
    - pseudo-static (defined when launching a new process or application)
    - dynamic (run-time adaptive)

11 The text between brackets ( ) identifies the generic safety requirement that is addressed by the feature.

- Ability to route interrupt sources to different cores (Predictability: non-critical interrupts should not be handled by cores processing critical tasks.)
- Periodic timers (primitive for the scheduler)

Processor core and SoC levels (suggestions):

- The controls of the above features shall only be accessed in the relevant privilege level. (integrity)
- A debug infrastructure is needed to deploy applications using the above features.
- Ability to detect, report and correct safety-related errors (ability to recover from a failure). This includes a hardware diagnostics layer for core and SoC health monitoring at boot time and periodically during operation.

8.2.1.2 Cognitive awareness

Support for acceleration software technologies

Necessary hardware support will be provided to efficiently execute the LEDEL library in the RISC-V platform. This includes the efficient interconnection of the LEDEL supported accelerators to memory and the CPU to accelerator communication. Within the framework of this task, the possibilities of High-level Synthesis CNN accelerators for SoC with embedded FPGAs.

LEDEL library will be provided in the RISC-V platform. This software component of FRACTAL will benefit from the development of different hardware accelerators in order to efficiently execute several Convolutional Neural Networks architectures on top.

Accelerator definition

The FPGA accelerator, described in section 8.1.1.2, will be part of the PULP platform as well.

As stated in 8.1.1.2, an analysis needs to be carried out, in order to determine the required hardware development needed for providing cognitive awareness. Last two paragraphs of that section apply to PULP platform as well.

Infrared camera support

In order to integrate the infrared camera module in the RISC-V platform for the iris diagnosis experimentation, s, so that they meet different requirements in term of data analysis, image acquisition, pre-processing of data. At same time the scope is to meet the requirements for the FRACTAL nodes for the expected cognitive awareness.

8.2.1.3 Integration

The PULP Platform has been primarily optimized for efficient ASIC implementations. FPGA based implementations are used for hardware emulation and to design demonstrators. This is why, basic mapping (memories to BRAMs, reset logic and a few others) will be performed. As a result, PULP based FPGA implementations target mostly modest operating frequencies (50-100MHz) on entry level FPGA boards.

Hardware emulation on an FPGA platform not only enables prototypes that could be used in in-field demonstrations at a higher Technical Readiness Level (TRL), but also has the benefit of adding a large variation of interfaces and supporting circuitry as part of a standard FPGA development platform that readily includes DDR memory, Flash, SD cards, USB, I2C, HDMI, Ethernet interfaces and many more. Therefore, FPGA implementations (when compared to actual ASIC implementations) are more flexible and can more readily be adapted to existing infrastructure.

8.2.2 Software requirements

8.2.2.1 Safety, security and low power management

Safety and security software support for the performance monitoring unit (PMU)

The PMU component will be accompanied with a developed library to read/write all PMU's registers, which includes configuring it and obtaining statistics.

The library will include the functionality of controlling the Maximum-Contention Control Unit (MCCU) quota counters values and alerting the user when allocated quotas have been exceeded, as well the capability of doing a close monitoring of the Request Duration Counters (RDC) and the Cycle Contention Stack (CCS) registers.

As part of these features, the library will include basic capabilities such as:

- Reset and/or initialize all counters or individual counters.
- Read statistics registers.
- Enable and disable PMU's components: MCCU, RDC or CSS.
- Self-test of PMU's components.

While not yet defined, the PMU will also include support to manage access rights to the PMU information and configuration for security purposes.

Software-only diverse redundancy support for safety

A safety element will be provided to reach some degree of diverse redundancy against common cause faults (CCF): a software-based lightweight Dual-Core Lockstep (DCLS) will be developed.

This safety feature builds on redundant processes created by the user, as well as a routine to compare the outcomes of those processes. The solution delivered builds on the following elements:

- A monitor process to orchestrate the execution of redundant applications on different cores guaranteeing the same dynamic instruction in both redundant applications never execute simultaneously, thus providing time diversity.
    - The monitor issues both redundant threads monitoring their staggering by building on their local performance monitoring counters.
    - Upon the detection of a risk of losing diversity during the next monitoring period, the monitor stalls the trailing thread during the next monitoring period.
    - If the trailing thread cannot catch up with the head thread during the next monitoring period, it is allowed to execute normally.
- Outcome comparison at the end of the execution.

Note that this solution has limitations related to the need of both redundant threads executing exactly the same control path and the same number of instructions, which is a constraint propagated to the user. Alternatively, we consider providing hardware support for true DCLS as a best effort option.

Software support for suggested safety features:

Along with the suggested hardware safety features identified above in a "bottom-up" fashion, the following software support is recommended:

- Mixed-criticality support: The platform shall accommodate safety processing along with less or not critical processes.
- Temporal partitioning and scheduling: predictability for critical processes sharing processor resources with other processes.
- Spatial partitioning:
    - To protect the integrity and confidentiality of the memory space and peripherals used by safety-critical processes.
    - Other non-memory mapped resources can be partitioned (dedicated buses, process-core affinity…)
    - Can also be used to make access time more predictable by managing the location of given data (e.g., scratchpad)
- Explicit communication between partitions: primitives to explicitly share data and send events between partitions that can have different criticality levels.
- QoS and WCET monitoring and control: to leverage related HW controls for the sake of critical processes (in a mixed-critical context)
- Data mapping tool:
    - To correctly exploit spatial partitioning, the SW developer needs to correctly map data to memory spaces.
    - This can be done thanks to the linker and/or a higher-level tool.
- Related documentation and API

Low-end OS-support

As an experimental implementation the NuttX RTOS will be integrated to the FRACTAL PULP node (https://nuttx.apache.org/docs/latest/index.html). NuttX offers a POSIX compatibility, a sort of limited version of Linux.

The motivation for this work is to be able to use simple processor cores. The energy consumption and component cost will potentially go down. Services Platforms and Cognitive Agents layers will have limitations depending how much memory they require and how much actual processing is required. After all FRACTAL architecture is designed to delegate the processing intensive tasks to the hardware, efficient implementation of this interface with NuttX is crucial.

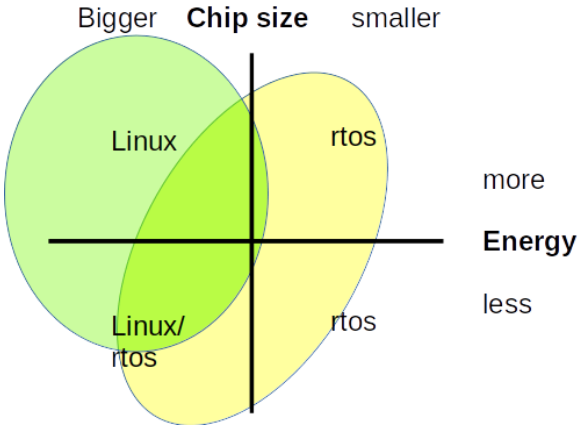

Figure 26 – Comparing Linux and RTOS requirements

The figure above illustrates this. The green circle presents the Linux platforms and the yellow circle this NuttX approach. Also, on the green area, the FRACTAL libraries are fully available. Due the NuttX POSIX compatibility the libraries are also available in the yellow area but with limitations. However, the idea will be that libraries themselves are not ported; they are just used as a limited set.

If we look at the use cases, the UC3 could utilize this solution best, but also other use cases where sub nodes need less features and chip price and/or energy consumption matters.

Yet another experimental option is to use OpenAMP[12] together with NuttX and Linux for the asymmetric multi-core processing. One core is running NuttX and the others Linux. Linux will be woken up for processing intensive tasks, e.g., for training AI. Asymmetric operation could be beneficial for various use cases, by potentially lowering energy need and requiring less resources from the chip.

In the drawing above this means that both the green and yellow circles can be used in the same application. While the PULP SMP is limited, it may be implemented to some other target.

12 https://www.openampproject.org/

8.2.2.2 Cognitive awareness

Acceleration support

The cognitive awareness needs, defined by the use cases requirements, will be fulfilled through the implementation of Artificial Intelligence (AI) algorithms in the software layer of the FRACTAL node. These AI algorithms will be focused on different neural network topologies according to the requirements of each use case and will be specially designed to be able to run on low energy resources. For this task, it will be provided the Low Energy Deep Learning Library (LEDEL), which will have these requirements:

- Focused on inference. The LEDEL available in the FRACTAL node will run the models, but these neural networks will probably need to be trained outside the LEDEL due to high computational requirements.
- ONNX format. To ensure the compatibility of the models, the LEDEL will use the Open Neural Network Exchange (ONNX) format. ONNX improves interoperability using different frameworks in the training step, and also eases the use of different hardware accelerators (check section 8.2.1.2).
- C++. The LEDEL will be provided in C++ and, therefore, the software platform will need a C++11 compiler.
- Linux. The LEDEL will work on Linux platforms.
- Other requirements. The LEDEL will need other standard requirements to be compiled. For example: cmake, eigen, protobuf, etc. These dependencies will be easily satisfied using the Anaconda environment (Python).
- Support both Fully Connected Neural Networks (FCNNs) and Convolutional Neural Networks (CNNs). Specific architectures (like Tiny-YOLO for UC4) will be defined later during the WP5 work.

Finally, some requirements related to the hardware accelerators will be defined according to 8.2.1.2. In order to use an accelerator, the LEDEL might incorporate other requirements related to that accelerator (for instance, CUDA support for NVIDIA GPU).

Infrared camera support

Hardware support for infrared camera module in the RISC-V platform will be granted in order to handle the iris diagnosis. The sensor should be connected to the RISC-V platform via the MIPI interface using the drivers already presents in the Linux kernel of the platform. The RISC-V interconnection will be managed with API and libraries selected for the purpose of expanding the functionality of the platform. The platform for data collection, pre-processing of data, data analysis, fault management and sensor setting will be implemented at software level.

8.2.2.3 Integration

POSIX compatible NuttX operating system will be integrated to the selected PULP platform(s). A necessary development environment and tools will be produced with

C/C++ support. Integration of service and agent layers are based on UC needs and NuttX resources.

Additionally, study the integration of OpenAMP asymmetric interface between Linux and Nuttx will be carried.

Regarding the integration of the LEDEL in the FRACTAL platform, the library will be provided in C++ code. In order to use the LEDEL, the code will be compiled to generate a library that can be included in any project. A detailed documentation will be generated in order to ease the compilation step according to the defined software and hardware stacks.

Software integration for infrared camera module in the RISC-V platform will be granted in order to handle the experimentation of the iris diagnosis. As mentioned above, the integration between the RISC-V Platform and infrared camera module will be managed with API and libraries selected for the purpose of expanding the functionality of the platform. This includes the support for the system integration process, the sensor configuration and calibration process, the sensor fault monitoring and the data collection process the whole it should be provided in C++ code.

9 Conclusions

This deliverable describes all the use cases of the FRACTAL project, it gives their context and highlights their objectives and contribution to the FRACTAL goals.

Each demonstrator also provides a set of business and technical Key Performance Indicators. These KPIs shall be further decomposed, they are at use-case level and shall be refined first at system level KPIs and then reach technical KPIs at component level.

The Use Cases are one core element driving the project with its needs and requirements highlighted in this document. These needs are then appointed to the "Node Architecture & Building Blocks" (WP3), "Safety, Security & Low Power Techniques" (WP4), "AI & Safe Autonomous Techniques" (WP5) and "Mutable and Fractal Communications" (WP6).

Even if this document is just a beginning and still need to be completed, tools will be used to gather them and track requirements described here. This tracking will allow to determine, as the project progresses, the coverage (number of requirements covered by the design) and compliance (number of fulfilled requirements).

10 List of Figures

Figure 1 – Top-down approach of the FRACTAL specification 10
Figure 2 – UC1 demonstrator 1 scheme.....14
Figure 3 – UC1 demonstrator 2 relations description 15
Figure 4 – UC1 demonstrator 2 scheme.....15
Figure 5 – Engine control diagram 19
Figure 6 – Physical Air Path diagram 20
Figure 7 – Predictive maintenance models link the in-use phase to the development and the workshop/maintenance phase 22
Figure 8 – Overview of UC2 24
Figure 9 – Smart meter diagram.....27
Figure 10 – VER-UC4 Object detection and recognition in industry 34
Figure 11 – CAF Istanbul's fully automated metro.....40
Figure 12 – Smart Totem illustration.....45
Figure 13 – Schematic representation of the VAL-UC6.....46
Figure 14 – Smart Physical Demonstration and Evaluation Robot (SPIDER).....51
Figure 15 – Sensor setup for collision avoidance function of VAL-UC7.....52
Figure 16 – Blocks from the Cognitive System to adapt for guaranteeing pillar 2 ..59
Figure 17 – Typical NoC-based MPSoC.....60
Figure 18 – Fractal security services at node and system level 64
Figure 19 – Blocks from the Cognitive System to adapt for guaranteeing pillar 3 ..67
Figure 20 – Secure Payload Application Data.....68
Figure 21 – Encryption techniques and security protocols at the transport layer, network layer, and link layer.....69
Figure 22 – Blocks from the Cognitive System to adapt for guaranteeing pillar 4 110
Figure 23 – Blocks from the Cognitive System for pillar 1 118
Figure 24 – VERSAL SoC schematic.....121
Figure 25 – PULPissimo SoC schematic.....127
Figure 26 – Comparing Linux and RTOS requirements 137

11 List of Tables

Table 1 – Document history 7
Table 2 – List of FRACTAL use cases 12
Table 3 – FRACTAL objectives 12
Table 4 – VER-UC1 requirements 18
Table 5 – VER-UC2 requirements 26
Table 6 – VER-UC3 requirements 32
Table 7 – VER-UC4 requirements 39
Table 8 – VAL-UC5 requirements 44
Table 9 – Target TRL for VAL-UC6 47
Table 10 – VER-UC6 requirements 50
Table 11 – VAL-UC7 requirements 55
Table 12 – VAL-UC8 requirements 58
Table 13 – WP4 requirements flowing down to WP3 66
Table 14 – UC1 DAI needs 72
Table 15 – UC2 DAI needs 73
Table 16 – UC3 DAI needs 73
Table 17 – UC4 DAI needs 74
Table 18 – UC5 DAI needs 74
Table 19 – UC6 DAI needs 75
Table 20 – UC7 DAI needs 76
Table 21 – UC8 DAI needs 76
Table 22 – Proposed requirements for Distributed Artificial Intelligence 77
Table 23 – UC1 AI performance needs 79
Table 24 – UC2 AI performance needs 79
Table 25 – UC3 AI performance needs 80
Table 26 – UC4 AI performance needs 80
Table 27 – UC5 AI performance needs 81
Table 28 – UC6 AI performance needs 81
Table 29 – UC7 AI performance needs 82
Table 30 – UC8 AI performance needs 83
Table 31 – UC1 inference needs 88
Table 32 – UC2 inference needs 89
Table 33 – UC3 inference needs 89
Table 34 – UC4 inference needs 90
Table 35 – UC5 inference needs 91
Table 36 – UC6 inference needs 92
Table 37 – UC7 inference needs 93
Table 38 – UC8 inference needs 94
Table 39 – Proposed requirements for inference 95
Table 40 – UC1 learning needs 95

Table 41 – UC2 learning needs 96
Table 42 – UC3 learning needs 96
Table 43 – UC4 learning needs 96
Table 44 – UC5 learning needs 97
Table 45 – UC6 learning needs 97
Table 46 – UC7 learning needs 97
Table 47 – UC8 learning needs 98
Table 48 – Proposed requirements for learning 98
Table 49 – UC1 run & development environment needs 102
Table 50 – UC2 run & development environment needs 103
Table 51 – UC3 run & development environment needs 103
Table 52 – UC4 run & development environment needs 104
Table 53 – UC5 run & development environment needs 104
Table 54 – UC6 run & development environment needs 105
Table 55 – UC7 run & development environment needs 106
Table 56 – UC8 run & development environment needs 107
Table 57 – Proposed requirements for run & development environment 107
Table 58 – WP5 requirements flowing down to WP3 108
Table 59 – WP6 requirements flowing down to WP3 117
Table 60 – UC requirements for the VERSAL hardware platform 119
Table 61 – UC requirements for the VERSAL software platform 120
Table 62 – WP4/5/6 requirements for the VERSAL software /hardware platform 120
Table 63 – UC requirements for the RISC-V hardware platform 128
Table 64 – UC requirements for the RISC-V software platform 128
Table 65 – WP4/5/6 requirements for the RISC-V software/hardware platform 130

12 List of Abbreviations

ACAP	Adaptive Compute Acceleration Platform (relates to VERSAL)
AHB	Advanced High-performance Bus
AI	Artificial Intelligence
AMBA	Advanced Microcontroller Bus Architecture
AMP	Asymmetric Multi-Processing
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
ARM	Advanced RISC Machines
ASIC	Application-Specific Integrated Circuit
ASIL	Automotive Safety Integrity Level
ATO	Automatic Train Operation
AXI	Advanced eXtensible Interface
BWT	Burrows-Wheeler Transform
CAN	Controller Area Network
CCF	Common Cause Faults
CCS	Cycle Contention Stack
CD	Continuous Delivery
CDO	Common Data Object
CEP	Complex Event Processing
CI	Continuous Integration
CNN	Convolutional Neural Network
CoA	Collision Avoidance
COTS	Component Off The Shelf
CPS	Cyber Physical Systems
CPSoS	Cyber-Physical Systems of Systems
CPU	Central Processing Unit
CS	Compressive Sensing
CSS	Cycle Contention Stack
CUDA	Compute Unified Device Architecture
CV	Computer Vision
DAI	Distributed Artificial Intelligence
DAI	Distributed Artificial Intelligence
DCLS	Dual-Core Lockstep
DCT	Discrete Cosine Transform
DDR	Double Data Rate memory
DevOps	Development and Operations
DFX	Dynamic Function eXchange
DL	Deep Learning
DNN	Deep Neural Network
DoA	Description of Action

DPU	Deep-learning Processing Unit
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
DVFS	Dynamic Voltage and Frequency Scaling
DWT	Discrete Wavelength Transform
ECU	Engine Control Unit
EDP	Energy-Delay Product
EGR	Exhaust Gas Recirculation
ELT	Extract, Load, Transform
EN	European Norm
ERTMS	European Rail Traffic Management System
EtherCAT	Ethernet for Control Automation Technology
ETL	Extract, Transform, Load
EU	European Union
FCNN	Fully Connected Neural Network
FLOPS	Floating Point Operation Per Second (and multiples: MFLOPS, GFLOPS)
FPGA	Field-Programmable Gate Array
FPU	Floating-Point Unit
FPS	Frames Per Second
GA	Grant Agreement
GB	Gigabyte
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
GPGPU	General Purpose GPU
HD	High Definition
HDMI	High-Definition Multimedia Interface
HESoC	Heterogeneous Embedded System on Chip
HLS	High-Level Synthesis
HW	Hardware
Hz	Hertz
I/O	Input/Output
I2C	Inter-Integrated Circuit interface
I2S	Inter-IC Sound
IBM	International Business Machines Corporation
IC	Integrated Circuit
ICT	Information and Communications Technologies
ICT	International Electrotechnical Commission
IDE	Integrated Development Environment
IID	Independent and Identically Distributed
IoT	Internet of Things
IOU	Intersection Over Union
IP	Internet Protocol
ISA	Instruction Set Architecture

ISO	International Organization for Standardization
IT	Information Technology
JBDC	Java Database Connectivity
JVM	Java Virtual Machine
KPI	Key Performance Index
LBIST	Logic Built-In Self-Test
LEDEL	Low Energy DEep Learning Library (LEDEL)
LLOD	Low Latency Object Detection
LTE	Long-Term Evolution
MAC	Multiply Accumulate
MB	Megabyte
MCCU	Maximum-Contention Control Unit
MIMO	Multiple-Input and Multiple-Output
MIPI	Mobile Industry Processor Interface
ML	Machine Learning
MLOps	Machine Learning and Operations
MMU	Memory Management Unit
MQ	Message Queueing
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrow Band IoT
NoC	Network-on-Chip
OBDC	Open Database Connectivity
ONN	Operational Neural Network
ONNX	Open Neural Network Exchange
OP	Operation (and multiples: MOP, GOP...)
OpenCL	Open Computing Language
OpenCV	Open Source Computer Vision Library
OPC UA	Open Connectivity Unified Architecture
OS	Operating system
OT	Operational Technology
OTA	Over The Air
PE	Processing Element
PL	Programmable Logic
PMC	Platform Management Controller
PMCA	Programmable Many Core Accelerators
PMML	Predictive Model Markup Language
PMP	Physical Memory Protection
PMU	Performance Monitor Unit
POSIX	Portable Operating System Interface
ProfiNET	Process Field Net
PTE	Power Train Engineering
PTF	Power Train Function
PULP	Parallel Ultra Low Power

QoS	Quality of Service
RAM	Random Access Memory
RCU	ROM Code Unit
RDC	Request Duration Counter
ReLU	Rectified Linear Unit
REST	Representational State Transfer
RISC	Reduced Instruction Set Computer
RLE	Run-Length Encoding
ROM	Read-Only Memory
RPU	Real-Time Processing Unit
RT	Real Time
RTOS	Real Time Operating System
SA	Sparse Approximation
SD	Secure Digital
SDK	Software Development Kit
SDN	System Defined Network
SIMD	Single Instruction Multiple Data
SLA	Service Level Agreement
SMP	Symmetric Multi-Processing
SO	Shared Object
SoC	System on Chip
SPI	Serial Peripheral Interconnect
SPIDER	Smart Physical Demonstration and Evaluation Robot
SQL	Structured Query Language
SW	Software
SWARM	System Wide Adaptive Ramp Metering
TBD	To Be Defined
TCP	Transmission Control Protocol
TFLite	TensorFlow Lite
TLB	Translation Lookaside Buffer
TPU	Tensor Processing Unit
TRL	Technical Readiness Level
TTY	Terminal Type
UART	Universal Asynchronous Receiver Transmitter
UAV	Unmanned Aerial Vehicle
UC	Use case
UDP	User Datagram Protocol
UNE	Asociación Española de Normalización
USB	Universal Serial Bus
VAL	Validation
VER	Verification
VLIW	Very Long Instruction Word
WCET	Worst Case Execution Time

WiFi	Wireless Fidelity
WP	Work Package
WS	Web Service
WSN	Wireless Sensor Network
WSN	Wireless Sensor Network
XRT	Xilinx RunTime
YARN	Yet Another Resource Negotiator
YOLO	You Only Look Once

The short names of FRACTAL partners are not considered as abbreviations: ACP, AITEK, AVL, BEE, BSC, CAF, ETH, HALTIAN, IKER, LKS, MODIS, OFFC, PLC2, PROINTEC, QUA, ROT, RULEX, SIEG, SIEM, SML, THA, UNIGE, UNIMORE, UNIVAQ, UOULU, UPV, VIF, ZYLK.