D8.3 Evaluation Result

Deliverable Id:	**D8.3**
Deliverable Name:	**Evaluation Result**
Status:	**Final**
Dissemination Level:	**Public**
Due date of deliverable:	**2023/08/31 (M36)**
Actual submission date:	**2023/08/31**
Work Package:	**WP8 "Case Studies, Specification, Benchmarking & Justification File"**
Organization name of lead contractor for this deliverable:	Virtual Vehicle Research (VIF)
Authors:	Markus Postl, Virtual Vehicle (VIF) Mario De Biase, Akkodis (former MODIS) Leticia Pascual Miret, Solver ML (SML) Raúl García Crespo, Solver ML (SML) Mikel Fernandez, BSC Juan Carlos Rodriguez, BSC Artur Kaufmann, BEEA Ilya Tuzov, UPV Stefano Delucchi, AITEK Jon Mikel Olmos, CAF Tania Di Mascio, UNIVAQ
Reviewers:	Luigi Pomante, UNIVAQ Gianluca Brilli, UNIMORE

Abstract:
This deliverable consists of a report on the results of the executed Justification. The report gives a statement whether the objectives of the FRACTAL project for each industrial use case are reached, discusses the implementation results, and provides plans for certification and commercial purposes.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056

Co-funded by the Horizon 2020 Programme of the European Union under grant agreement No 877056.

Contents

1 History

Version	Date	Modification reason	Modified by
0.0	27/10/2022	Template	VIF
0.1	06/07/2023	First version ready for internal review	Authors
0.2	01/08/2023	Internal review	Reviewers
0.3	28/08/2023	Minor changes after review and ending implementation phase	UC Leaders
1.0	31/08/2023	Version ready for submission	VIF

2 Summary

This document is the outcome of task T8.4, Case Study Justification File. Four of the FRACTAL use cases take part in WP8 (Case Studies, Benchmarking and Quality) for the industrial validation of FRACTAL developments:

- UC5 Increasing the safety of an autonomous train through AI techniques;
- UC6 Elaborate data collected using heterogeneous technologies (intelligent totem);
- UC7 Autonomous robot for implementing safe movements;
- UC8 Improve the performance of autonomous warehouse shuttles for moving goods in a warehouse.

This document presents the results of the implementation activities, which were collected in the *Justification File*. The Justification File lists Key Performance Indicators (KPI) from D8.1, Specification of Industrial validation Use Cases, defines suitable validation methods and test cases for each KPI, and tracks the validation status. In addition, each use provides a benchmark for comparison of the use case system to a comparable state-of-the-art system.

All results are discussed, including plans for improvements, consideration of safety and security, certification, and exploitation to prepare the realization of commercial products.

2.1 Achievements

Highlights, lowlights, results, and novelties are discussed for each use case in the subsections "X.3.2 Discussion of the results".

3 Introduction

This document is output of the task T8.4 (Case Study Justification File). It displays the results of the KPIs defined in D8.1 (Specification of Industrial validation Use Cases).

This deliverable is organized per use case with a specific chapter dedicated to each use case. The chapters are divided into five sections:

- Results of the executed justification plan
- Results of the executed benchmark
- Evaluation of the results
- Consideration of safety and security
- Preparation for realization of commercial products

In the section *'Results of the executed justification plan'* each use case reports the achievement of the defined KPIs. The list of KPIs was defined in D8.1 and contains: KPIs for Implementation Plan Tasks, KPIs for FRACTAL Objectives related to FRACTAL Pillars, and KPIs for UC Features. Within task T8.4 the list of KPIs was transferred to the Justification Plan, an Excel sheet hosted in FRACTAL SharePoint. For each KPI a validation method is defined (e.g., Integration Test, Unit Test, Simulation) and a validation status, see Figure 1, is assigned. Depending on the validation method, test cases are defined. The test cases are attached to the deliverable in Appendix A: Test Cases.

Validation Status	
Fullfilled (Tested)	KPI met target as defined, verification with defined test
Fullfilled (No test)	KPI met target as defined, no test required (add validation comment)
Partially fulfilled	KPI met under certain conditions (add validation comment)
Not fulfilled	KPI could be not fulfilled
Not validated	KPI was not validated (e.g. desired functions)
Deleted	KPI was deleted (justify with comment)

Figure 1 - Validation status of Justification File

The section *'Results of the executed benchmark'* focuses on comparison of the use cases to a state-of-the-art system based on the KPIs defined in D8.1.

A discussion of the results from the sections above is given in the section *'Evaluation of the results'*. Further, highlights of the implementation and perspectives on future improvements are mentioned.

The section 'Consideration of safety and security' addresses safety and security issues in the application and gives argumentation on the needs for those by the use case implementation.

Finally, the section '*Preparation for realization of commercial products*' gives a statement on how the use case supports the realization of commercial products and its requirements on certification.

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

UC5 has as target the improvement of autonomous technologies in railway through the search for suitable platforms that can execute AI based functions with safety capabilities. Autonomous driving in railway is a high complexity challenge from the integration point of view. Basic train operation is extended with systems that can generate driving profiles based on static track information in first automation step. Further steps require the introduction of environment perception (PER) as dynamic information supply for automatic driving systems. Within this context there are several information sources to be taken into account to allow safe and seamless automatic train operation. UC5 implements two functionalities chosen from the full set of functionalities that need to be automated for releasing train driver from supervision.

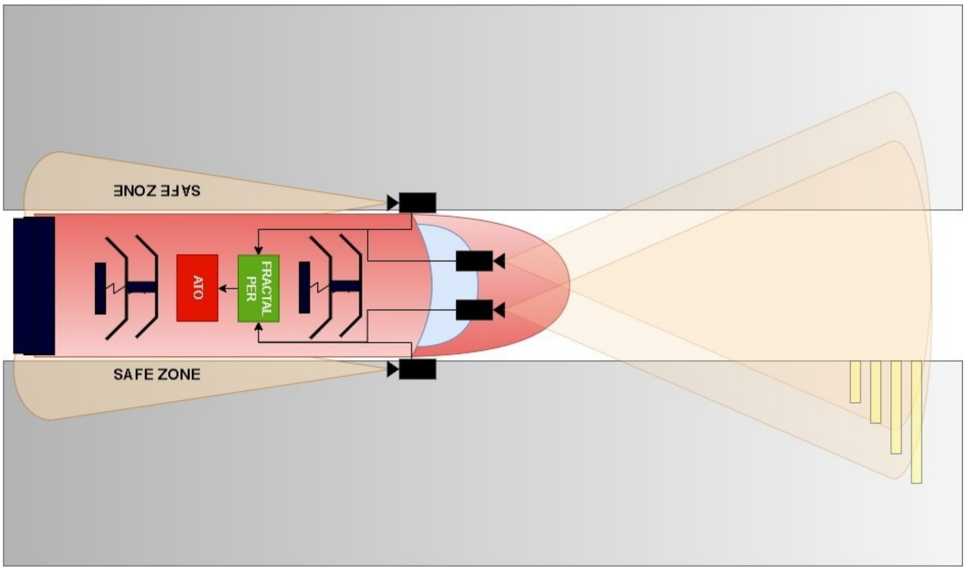


Figure 2 – Sensor setup for UC5

Automatic accurate stop

Stopping the train at the right location is a challenge when the positioning introduces greater error than the stopping precision required. In certain environments such as underground, the required stopping accuracy is as low as ±10cm if platform doors are installed. Dynamic correction based on visual references is the approach presented in UC5 for this situation. Stopping landmarks are detected using AI and distance to that location is calculated using Stereo Vision in order to correct train positioning.

Safe Passenger transfer

Automatic driving can be applied between stations but the operation in station based on passenger boarding is currently manual. Opening/Closing the doors and departing needs environment supervision to check that there are no passengers that can get injured if train departs. Withing this context UC5 implements person detection on the rear mirror cameras of the train for checking door and train surroundings.

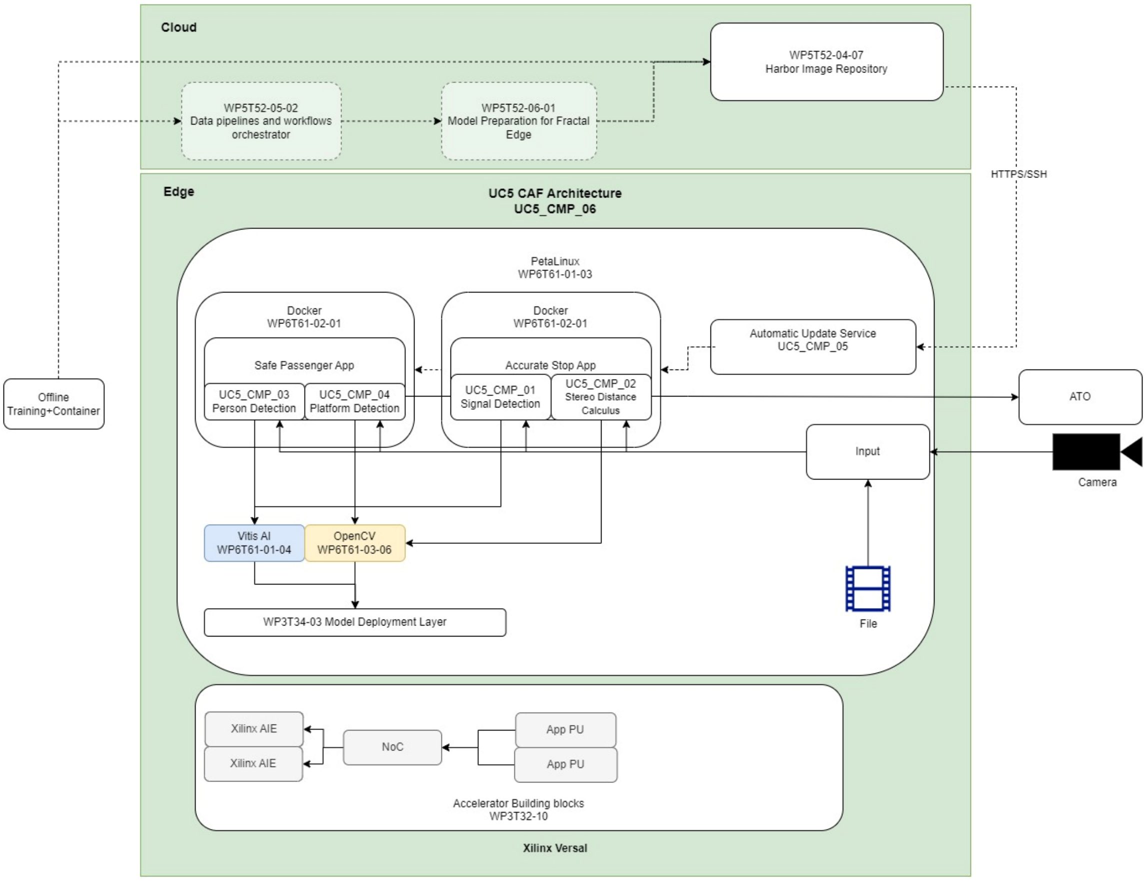
Automatic Software Update

Software updates for On-Board systems are usually performed manually on-site which requires large expenses on maintenance costs. As an extension to Automatic Accurate Stop and Safe Passenger Transfer, introducing a way to distribute software and AI models in a centralized way based on new cloud-edge concepts introduced in fractal can lead to an additional improvement in cost expenses reduction. This UC extension presents the SW and the AI model as a containerized item that is stored in the FRACTAL cloud as the official release. On system startup, edge to cloud connection is stablished and new SW is fetched to keep On-Board system updated.

4.1 Results of the executed justification plan

4.1.1 Implementation

UC5 Implementation is based on integration of VERSAL-dedicated components for base UC functionalities (Automatic Accurate Stop and Safe Passenger Transfer). Inference is achieved through integration of components WP3T32-10(Accelerator building blocks) and WP3T34-03(Versal model deployment layer). Those components allow the use of DPU accelerator in Versal FPGA at HW level providing a SW API to load the model and execute inference on given data processing unit (DPU).



Edge DPU requires a quantized model, for translating the trained model to Edge compatible format a processing pipeline is defined. Initially the model is translated from ONNX original format to h5 format. Model in h5 format is then quantized to integer precision using a subset extracted from training dataset to evaluate the performance variation reducing the precision from original 32-bit float. Component WP5T52-06-01 allows this model translation either processed offline and in cloud. For cloud model translation deployment component WP5T52-05-02(Data pipelines and workflow orchestrator) for orchestrating the translation scripts at manual model input.

Automatic updates allow to extend the use case simulating a Control Center (Cloud) that centralizes SW release management and train fleet (Edge) that executes the SW release with the UC5 base functionalities. This application requires further elements to be integrated. For compact SW distribution, the model and the test binaries are introduced in a Docker container. Docker container processing needs slight changes on the Linux image generated in component WP3T34-03 and WP3T34-03 integration to enable docker container execution in Edge. This paradigm change also requires cloud infrastructure to host the official Docker image containing SW releases which is provided by component WP5T52-04-07 (Images Repository).

The integration result has been measured oriented to UC5 defined KPIs with several tests planned to verify each integration step related to UC5 Key metrics. The integration tests are described in Annex A.

4.1.2 Justification

Justification in UC5 is oriented to covering all the base UC requirements. All KPIs related to main features and requirements are fulfilled except for the non-functional qualification of the platform which requires higher TRL.

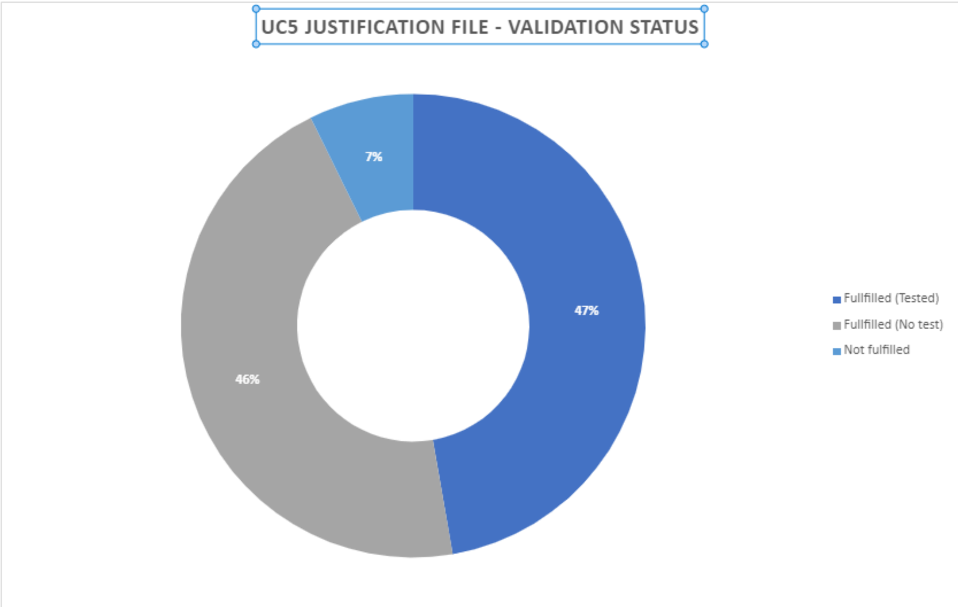


Figure 3 - Validation Status of UC5

Justification of KPI Results (UC5)					
KPI ID	Description	Validation Method	Evidence	Validation Status	Validation Comments
UC5_KPI_IP_01	All subtask success	-	-	Fullfilled (No test)	
UC5_KPI_IP_02	Inference time	Unit Test	UC5_T5	Fullfilled (Tested)	12fps > 10fps requirement
UC5_KPI_IP_03	Build OpenCV on for Target (Versal ARM64) success	Unit Test	UC5_T2	Fullfilled (Tested)	SGBM & BM algorithms working properly
UC5_KPI_IP_04	Accuracy % with respect to X86 platform	Unit Test	UC5_T6	Fullfilled (Tested)	Dealing with quantization
UC5_KPI_IP_05	Build CAF Demonstration Software on Target success	-	-	Fullfilled (No test)	Demo SW working
UC5_KPI_IP_06	Build Safe Passenger Transfer application success	-	-	Fullfilled (No test)	Demo SW working
UC5_KPI_IP_07	Build Accurate Stop application success	-	-	Fullfilled (No test)	Demo SW working
UC5_KPI_IP_08	All subtask success	-	-	Fullfilled (No test)	Implementation not finished yet
UC5_KPI_IP_09	Hours of video recorded	-	Database Stats	Fullfilled (No test)	
UC5_KPI_IP_10	Image Database size	-	Database Stats	Fullfilled (No test)	
UC5_KPI_IP_11	Model accuracy over test database	Unit Test	UC5_T6	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_IP_12	All subtask success	-	-	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_IP_14	Working under secure connection success	Unit Test	UC5_T3	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_IP_15	Model and docker image cloud hosting success	-	-	Fullfilled (No test)	Implementation not finished yet
UC5_KPI_IP_16	Cloud repositories version handling success	-	-	Fullfilled (No test)	Implementation not finished yet
UC5_KPI_IP_17	Metrics obtained for defined model both in X86 and Versal	Unit Test	UC5_T6	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_FO_00	Fractal Technology helps improving State-of-Art in Railways Sector	-	Any KPIs achieved	Fullfilled (No test)	12fps in VERSAL vs 8fps AGX Xavier
UC5_KPI_FO_01	Real-time Inference Time and Accurate high perfomance Cognitive AI based node implemented and running.	Unit Test	UC5_T5	Fullfilled (No test)	12fps> 10fps requirement
UC5_KPI_FO_02	Edge Node application with Secure connection to the Cloud implemented and running.	Unit Test	UC5_T3	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_FO_03	Edge Node Software and Model update Max Time, guaranteeing data is not corrupted, implemented and running.	Unit Test	UC5_T7	Fullfilled (Tested)	Implementation not finished yet
UC5_KPI_FO_04	Software and Model version handling on cloud impleme	-	-	Fullfilled (No test)	Harbor
UC5_KPI_FT_01	Edge Node has USB-C port	-	-	Fullfilled (No test)	Fulfilled by HW properties
UC5_KPI_FT_02	Edge Node has Ethernet connector RJ45	-	-	Fullfilled (No test)	Fulfilled by HW properties
UC5_KPI_FT_03	Edge Node Vitis-AI allows importing and executing ONNX models	Unit Test	UC5_T4	Fullfilled (Tested)	ONNX needs to be transformed, but supportec
UC5_KPI_FT_04	Edge Node Vitis-AI allows importing and executing Yolo	Unit Test	UC5_T5	Fullfilled (Tested)	Execution tested
UC5_KPI_FT_05	Edge Node inference time allows real-time processing of frames	Unit Test	UC5_T5	Fullfilled (Tested)	12fps > 10fps requirement --> Real time in UC5
UC5_KPI_FT_06	Build & System Integration	-	-	Fullfilled (No test)	Demo SW working
UC5_KPI_FT_07	AI Inference Accuracy on Model	Unit Test	UC5_T6	Fullfilled (Tested)	Dealing with quantization
UC5_KPI_FT_08	Edge Node has the ability to perform inference	Unit Test	UC5_T4	Fullfilled (Tested)	
UC5_KPI_FT_09	Edge Node has OpenCV	Unit Test	UC5_T2	Fullfilled (Tested)	
UC5_KPI_FT_10	Cloud Data Set Version Control	-	-	Fullfilled (No test)	Given by SW Version control
UC5_KPI_FT_11	Edge Node frame processing rate > 10fps	Unit Test	UC5_T5	Fullfilled (Tested)	
UC5_KPI_FT_12	Edge Node allows video processing	Unit Test	UC5_T5	Fullfilled (Tested)	-
UC5_KPI_FT_13	Safety Regulation ISO 26262 Automotion	-	-	Not fulfilled	Higher TRLs
UC5_KPI_FT_14	Safety Regulation ISO 61508 Generic	-	-	Not fulfilled	Higher TRLs
UC5_KPI_FT_15	Safety Regulation CENELEC EN50126/8/9: Railway Industry	-	-	Not fulfilled	Higher TRLs
UC5_KPI_FT_16	Edge Node in Low Power has ONNX models	Unit Test	UC5_T4	Fullfilled (Tested)	Indirectly via quantization
UC5_KPI_FT_17	Edge Node allows secure storage of data	-	-	Fullfilled (No test)	Access control by OVH cloud
UC5_KPI_FT_18	Edge Node allows Authentication / Authorization	Unit Test	UC5_T1	Fullfilled (Tested)	Linux Auth
UC5_KPI_FT_19	Fractality communication via Ethernet	-	-	Fullfilled (No test)	
UC5_KPI_FT_20	Edge Node is implemented on Versal	-	-	Fullfilled (No test)	Fulfilled by design
UC5_KPI_FT_21	Edge Node executes LINUX Operating System	Unit Test	UC5_T1	Fullfilled (Tested)	Petalinux generated image tested
UC5_KPI_IP_Req_01	Edge Node Platform Ruggerized	-	-	Not fulfilled	Ruggerisation is oriented to higher TRLs
UC5_KPI_IP_Req_02	Edge Node Inference Time	Unit Test	UC5_T5	Fullfilled (Tested)	
UC5_KPI_IP_Req_03	Edge OpenCV Support	Unit Test	UC5_T2	Fullfilled (Tested)	
UC5_KPI_IP_Req_04	Edge ONNX Support	Unit Test	UC5_T4	Fullfilled (Tested)	
UC5_KPI_IP_Req_05	HW Accelerator Compatible wiht TensorFlow	Unit Test	UC5_T4	Fullfilled (Tested)	Via ONNX + Quantization
UC5_KPI_IP_Req_06	Edge Node with al least 4 cores	-	-	Fullfilled (No test)	Fulfilled by design, 2+2 CPU system
UC5_KPI_IP_Req_07	Edge Node with mutlithreading	-	-	Fullfilled (No test)	OpenMP Support by ARM
UC5_KPI_IP_Req_08	Edge Node at least 60 GFLOPS	Unit Test	UC5_T5	Fullfilled (Tested)	Tested with 12fps performance result
UC5_KPI_IP_Req_09	Edge Node at least 16GB DDR RAM	-	-	Fullfilled (No test)	8 GB LPDDR4 + 8GB DDR4
UC5_KPI_IP_Req_10	Edge Node with HW Acccelerator	-	-	Fullfilled (No test)	Fulfilled by design
UC5_KPI_IP_Req_11	Edge Node Multi-Interfaces and their Linux Drivers	-	-	Fullfilled (No test)	Fulfilled by design
UC5_KPI_IP_Req_12	Edge Node with Linux OS	Unit Test	UC5_T1	Fullfilled (Tested)	Tested Petalinux Image
UC5_KPI_IP_Req_13	Platform Release with C++ compiler/crosscompiler	-	-	Fullfilled (No test)	

Table 1 - Justification of KPI Results from UC5

4.2 Results of the executed benchmark

4.2.1 Benchmark Definition

UC5 Benchmark is defined stablishing as baseline the KPIs obtained in an already available in market platform. The benchmark contender is an AGX Xavier platform from Nvidia. This platform has 8-core ARM based CPU with Nvidia Volta GPU integrated in a SoC. 32 GB of RAM memory are mapped in common CPU/GPU memory space that allow to transfer data from CPU to GPU without copies. The benchmark is strongly based on performance metrics. Key indicator is the inference time which in SoA is above 100 ms per image using YoloV3 608x608 model.

4.2.2 Benchmark Results

In the results achieved in Versal FRACTAL node, an improvement of 20% can be seen with respect to SoA system. To execute in Versal DPU quantization needs to be applied reducing the precision of the model. This reduction is not as critical as performance drop would be because postprocessing can be applied to model detection to introduce temporal redundancy (tracking) that guarantees that, even if some detections are lost, final prediction still is valid.

Results of other feature evaluation show that FRACTAL platform can provide further capabilities that are not present in SoA which allows to add connectivity to application. The edge node introduces low power and real time modes that are not currently required but can enable other hard real time functionalities to be integrated within the same platform.

BENCHMARK			UC FRACTAL SYSTEM	State-Of-Art System (NVIDIA AGX Jetson Xavier)
UC5_KPI_FO_01	Real-time Inference Time and Accurate high perfomance Cognitive AI based node implemented and running.	< 100ms	83ms (full yoloV3)	125ms (full yoloV3)
UC5_KPI_FO_02	**Edge Node /SoA System** application with Secure connection to the Cloud implemented and running.	True/False	TRUE	FALSE
UC5_KPI_FO_03	**Edge Node /SoA System** Software and Model update Max Time, guaranteeing data is not corrupted, implemented and running.	< 1 min	TRUE	Not applicable
UC5_KPI_FO_04	Software and Model version handling on cloud implemented and running.	True/False	TRUE	FALSE
UC5_KPI_FT_01	**Edge Node /SoA System** has USB-C port	True/False	TRUE	TRUE
UC5_KPI_FT_02	**Edge Node /SoA System** has Ethernet connector RJ45	True/False	TRUE	TRUE
UC5_KPI_FT_03	**Edge Node Vitis-AI / SoA System TensorRT** allows importing and executing ONNX models	True/False	TRUE	TRUE
UC5_KPI_FT_04	**Edge Node Vitis-AI / SoA System** TensorRT allows importing and executing Yolo V3/V4	True/False	TRUE	TRUE
UC5_KPI_FT_05	**Edge Node /SoA System** inference time allows real-time processing of frames	< 100ms	83ms (full yoloV3)	125ms (full yoloV3)
UC5_KPI_FT_07	**Edge Node /SoA System** AI Inference Accuracy on Model	> 75%	TRUE	75% mAp
UC5_KPI_FT_08	**Edge Node /SoA System** has the ability to track location	True/False	TRUE	TRUE
UC5_KPI_FT_09	**Edge Node /SoA System** has OpenCV	True/False	TRUE	TRUE
UC5_KPI_FT_10	Cloud Data Set Version Control	True/False	TRUE	FALSE
UC5_KPI_FT_11	**Edge Node /SoA System** frame processing rate > 10fps	>10fps	12 fps (full yoloV3)	8 fps (full YoloV3)
UC5_KPI_FT_12	**Edge Node /SoA System** allows video processing	True/False	TRUE	TRUE
UC5_KPI_FT_13	Safety Regulation ISO 26262 Automotion	True/False	FALSE	FALSE
UC5_KPI_FT_14	Safety Regulation ISO 61508 Generic	True/False	FALSE	FALSE
UC5_KPI_FT_15	Safety Regulation CENELEC EN50126/8/9: Railway Industry	True/False	FALSE	FALSE
UC5_KPI_FT_16	**Edge Node /SoA System** in Low Power has ONNX models	True/False	TRUE	FALSE
UC5_KPI_FT_17	**Edge Node /SoA System** allows secure storage of data	True/False	TRUE	FALSE
UC5_KPI_FT_18	**Edge Node /SoA System** allows Authentication / Authorization	True/False	TRUE	TRUE
UC5_KPI_FT_20	**Edge Node /SoA System** is implemented on Versal	True/False	TRUE	FALSE
UC5_KPI_FT_21	**Edge Node /SoA System** executes LINUX Ooperating System	True/False	TRUE	TRUE

Table 2 - Results of the Benchmark from UC5

4.3 Evaluation of the results

4.3.1 Evaluation of Business KPIs

Business KPIs related to UC5 are not derived directly from the implementation advantages. In UC5 the suitability of the platform evaluated in terms of technical KPIs enables the use of secondary system to operate the train in automatic way that implements real business KPIs. This system, called ATO (Automatic Train Operation), has relevant impact on driving style improving several aspects of train operation:

Energy Consumption

Efficient use of traction and braking commands reduces the energy consumption based on static track data and environment information gathered by UC5 implementation-based system (PERception). Implementing ATO system up to 30% energy consumption can be achieved depending on track properties (gradients, maximum speed, track shape) which produces significant budget savings to train operator.

Punctuality

Real track data information allows ATO system to estimate the arrival times in a way that reduces 95% of the delays caused by driving style by calculating the exact traction/brake profile that minimize the energy consumption, maximizes comfort, and finished at exact arrival time on destination. This improvement increases transport quality given by train operator and transport reliability, increasing indirectly the number of users and the earnings.

SW Maintenance costs

Nowadays, an update in SW is performed manually connecting to HW on-site by either manufacturer field technicians or operator maintenance staff which requires additional work shifts and transport. By implementing automatic updates, the savings in such operations are very high depending on fleet size.

"KPI for Business Improvements " for the UC	Description	Assessment methodology	Baseline	Target	Improvement	Achieved?	Comments
UC5_KPI_B1	Energy Consumption	Energy measurement in ATO operation	100%	70%	30% reduction	If Fractal UC5 implementation fullfils performance requirement then yes	ATO (automatic train operation) system requires UC5 Functionality working in order to allow proper ATO system operation.
UC5_KPI_B2	Train Punctuality	Incidence based on deviation from arrival time caused by driving style	100%	5%	95% reduction	If Fractal UC5 implementation fullfils performance requirement then yes	ATO (automatic train operation) system requires UC5 Functionality working in order to allow proper ATO system operation.
UC5_KPI_B3	Maintenance costs on SW update	Total cost of SW update (nightshift,trip)	100%	1%	99% reduction	If automatic sw updates can pull SW automatically yes	SW can be done without on-site human intervention which reduces maintenance costs.

Table 3 - "KPI for Business Improvement" for the UC5

4.3.2 Discussion of the results

Results of the justification plan and benchmark show that FRACTAL edge platform is a suitable candidate regarding main performance technical KPI. With the 12 fps inference achieved the platform improves the threshold of 10 fps required and established as baseline.

There are other metrics that need to be taken into account like the model degradation produced by quantization required by Versal DPU. Analyzing Precision, Recall and Confusion matrix applied to evaluation dataset, an overview of the changes applied by quantization can be inferred.

From evaluation results a wide variation in terms of class can be seen. For UC5 relevant classes, which are the station start and end lines, there is no relevant variation during quantization.

	Red Semaphore	Green Semaphore	Orange Semaphore	Blue Semaphore	Station Start Lines	Station End Lines	S Signal	A Signal	Pre Speed 30	Speed 30	Speed 40	Speed 45	Speed 50	Speed 55	Speed 65	Speed 70	Speed 75	Speed 85	Speed Unknown	C Signal	Speed 80
Red Semaphore	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Green Semaphore	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Orange Semaphore	0.05	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Blue Semaphore	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Station Start Lines	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Station End Lines	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
S Signal	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A Signal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pre Speed 30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Speed 30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.00	0.00
Speed 40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Speed 45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00
Speed 50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.75	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00
Speed 55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.67	0.00	0.00	0.00	0.10	0.00	0.00
Speed 65	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.40	0.00	0.00	0.00
Speed 70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.00	0.00	0.02	0.00	0.00
Speed 75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	1.00	0.00	0.00	0.00	0.00
Speed 85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.07	0.00	0.00
Speed Unknown	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.00	0.00
C Signal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Speed 80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 4 - Confusion Matrix for inference in Versal applying 0.75 confidence threshold to quantized model

	Red Semaphore	Green Semaphore	Orange Semaphore	Blue Semaphore	Station Start Lines	Station End Lines	S Signal	A Signal	Pre Speed 30	Speed 30	Speed 40	Speed 45	Speed 50	Speed 55	Speed 65	Speed 70	Speed 75	Speed 85	Speed Unknown	C Signal	Speed 80
Red Semaphore	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Green Semaphore	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Orange Semaphore	0.03	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Blue Semaphore	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Station Start Lines	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Station End Lines	0.00	0.00	0.00	0.00	0.04	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
S Signal	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A Signal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Pre Speed 30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Speed 30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00
Speed 40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.08	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.00	0.00
Speed 45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.25	0.00	0.00	0.00	0.00	0.08	0.04	0.00	0.00
Speed 50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.38	0.56	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00
Speed 55	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.76	0.43	0.00	0.00	0.00	0.13	0.00	0.00
Speed 65	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.00	0.21	0.00	0.00	0.00	0.02	0.00	0.00
Speed 70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.58	0.00	0.00	0.02	0.00	0.00
Speed 75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.38	0.96	0.00	0.00	0.00	0.00
Speed 85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.36	0.00	0.00	0.69	0.00	0.00	0.00
Speed Unknown	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.02	0.00	0.00	0.68	0.00	0.00
C Signal	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
Speed 80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.00	0.02	0.04	0.08	0.00	0.00	1.00

Figure 5 - Confusion Matrix for inference in X86 applying 0.75 confidence threshold to raw model

General metrics comparison shows a significant degradation on overall Recall. The cause for Recall degradation can be explained analyzing confusion matrix. The quantization produces poor performance on speed sign classes derived from not sufficient examples on those classes in quantization dataset.

	x86		
Conf_Threshold	Precision	Recall	F1 Score
0.25	70%	62%	66%
0.5	73%	58%	67%
0.75	77%	52%	62%

	Versal		
Conf_Threshold	Precision	Recall	F1 Score
0.25	69%	56%	62%
0.5	77%	47%	58%
0.75	82%	36%	49%

Figure 6 - Precision, Recall and F1 Score comparison between Versal Quantized and X86 Raw models for thresholds 0.25, 0.5 and 0.75

Conclusion extracted from the impact of translation in model metric shows that the model after required translation is still valid for UC5 application.

FRACTAL implementation introduces great improvements with respect to the State of the Art. The main improvement is aligned to performance use case requirement, a 20% improvement in performance while executing YoloV3-608x608 is observed comparing the Xilinx DPU to AGX Xavier GPU. Additionally other considerations are also remarkable like the ability to extend the functionality using other components when required and the flexibility to change the accelerator adapting it to other AI models during State of the Art evolution in image detection. Cloud capability is also a point to be taken into account because train digitalization is also in automation roadmap.

On the other hand, the platform has not yet the required maturity and physical properties to be installed on rolling stock. Railway grade encapsulation is not yet available in the market and that avoids the commercial exploitation of the platform. The cost for the HW is also higher than the available Railway Grade platforms on the market (12 K€ vs 3.5K€) which has not great impact taking into account the full price per train but it is still high. Developing time is also a relevant factor when modifying the UC application (beyond FRACTAL project), it requires design stages at HW level and knowledge of the platform insides while SoA is has more transparent view of underlaying Hardware requiring only SW developers to deploy application.

Highlights	Lowlights
++Improved performance	-Not yet the required TRL for railway commercial exploitation
+Configurability and Extendibility	-Platform HW cost very high vs baseline
+Cloud Capable	-Higher developing time and HW know-how

Table 4 - Highlights and Lowlights of UC5

4.4 Consideration of safety and security

4.4.1 Safety

Within the train standard interoperable architecture, safety functionalities are defined together with the systems that are in charge for covering them. Those systems are qualified as SIL and given an integrity level. Depending on the integrity level defined for given functionality, all the SW/HW related to it must follow a specific lifecycle that allows the application to be certified.

First automation level in train systems introduces the ATP (Automatic Train Protection) which, up to date covers the safety functionalities related to driving behavior applying supervision to driver. Next level of automation introduces ATO (Automatic Train Operation) system that applies traction and brake commands based on given track data. ATO system is not SIL qualified as it operates always under ATP supervision. Finally, the last automation level introduces PER (PERception) system which provides ATO with the missing information about environment stablishing the base for UC5 presented functionalities.

In this fully automated architecture, PER system is not yet Safety-Qualified. Such qualification would imply certification of AI models and regulation is not yet at that point. Additionally, it would imply that PER system should have direct communication with ATP system or emergency brake (safety chain) which would require changes on those standards. Latest architectural definitions rely the safe part of the new functionalities on a new system that is SIL qualified and process PER gathered information taking into account additional track data that is not available for

Perception system. For those reasons PER system, and therefore UC5 implementation is not safety related as the information gathered by Safe passenger transfer and Accurate stop applications from UC5 would be processed again in order to take operational reactions during train operation.

4.4.2 Security

Regarding security concept, Railway regulation is also evolving. There are several levels for security grade in terms of the required infrastructure for final application deployment.

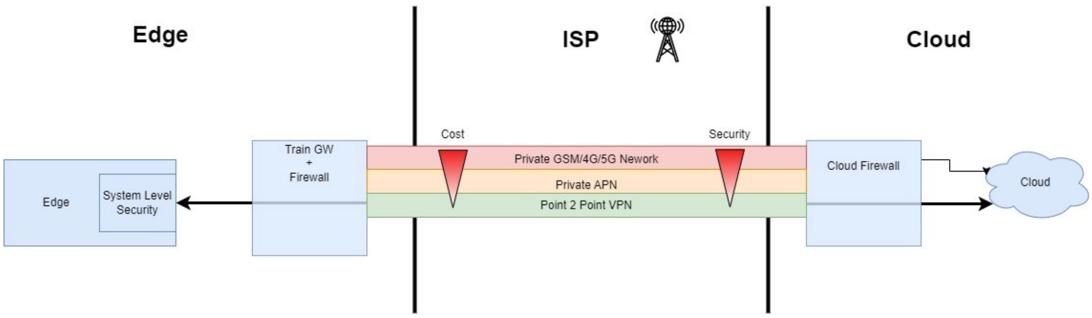


Figure 7 - UC5 Communication Infrastructure

System Level Security

This level introduces controls to secure systems against unwanted intrusion and sensible data leakage. In this level several measures are defined:

- Authentication: Ensure that access to all exposed resources is regulated by strong methods (2FA, ssh keys)
- Encryption: Ensure that all information flows entering or leaving the application are unknown for potential aggressors that could inject corrupt data and affect train behavior.
- Availability: Ensure that critical systems remain working under unexpected intrusion events.
- Registering & Diagnosis: Register login and connection events in a persistent way to allow forensic analysis. Runtime traffic analysis to detect security events and apply reaction.
- Avoid unauthorized SW modifications: Secure boot, checksum/CRC.

Local Network Security Level

On Board network configuration allowing only required routes/access. Proper network segmentation isolating driving-oriented systems from passenger comfort systems/surveillance/others.

Train-CTC Point to Point Security Level

Securing the connection between train Gateway and CTC/Cloud applying VPN encryption in order to allow only authorized peers in that channel. All traffic through public carrier tunneled and encrypted. Firewalls applied in both sides of the connection restricting all traffic except from the required.

APN Level Security

While using wireless networks, request for private APN for given application in order to isolate application dataflow from public communications (GSMR,3G,4G,5G). Only allowed SIM cards can register the private APN. Introducing this isolation increases exponentially the deployment costs and requires telephony provider configuration which also increases the deployment time.

Wireless Level Security

Further steps from APN level security rely on requesting/purchasing private RF bands and deploying custom RF infrastructure to generate a private GSMR, 3G or 4G network. This is the highest security isolation level and the costs associated are very high.

Fractal edge node should implement all the controls related to system level security as other consideration are project-dependent. In UC5 edge security is relevant in any situation where the node has external connectivity and, therefore, is vulnerable to unauthorized intrusions and data corruption.

4.5 Preparation for realization of commercial products

4.5.1 Relevant standards for railway deployment

While analyzing the required standards in railway industry there is a clear split between them. Some of them are related to Hardware and Physical properties of the platform and others are related to the design flow involved, most of them related to safety related applications and platforms.

4.5.1.1 Base Railway Standards

The base standards that any HW platform in railway must comply are those related to environmental hardening, fire hazard and power management. The requirement for those standards is independent from the safety level of the application. The certification document EN50155 gathers all the base requirements for railway hardware.

4.5.1.2 Safety Railway Standards

The specification for safety related application is contained in some standards that define the lifecycle that the system must follow to be suitability for certification. The standard for railway safety regulation is EN50126.

4.5.2 Exploitation plans

Exploitation in UC5 requires the base standards for railway Hardware and higher TRL and certifying the platform is beyond the available resources. For that reason, there are no further exploitation plans than testing purposes in laboratory.

5 VAL-UC6 Elaborate data collected using heterogeneous technologies (intelligent totem)

The objective of UC6 is to demonstrate how the Fractal approach allows to satisfy response time requirements of users in a sentient space. This aim is achieved thanks to a processing entirely performed at the edge-level through a network composed of Fractal nodes (named Totem and Roof nodes).

The proposed solution can provide tailored contents to users (age, gender, idiom) and provide those contents to several users with a response time of 1s. Even if the number of users increases, the network always provides a response time of 1s, without accessing the cloud.

The UC6's reference actual scenario is a shopping mall (Figure 8) turned into a sentient space by embedding processing resources within the set physical environment. Hence, this space can be seen as a network of interconnected nodes, each with its sensing and processing resources located inside the shopping mall itself.

The Totem is then equipped with heterogeneous sensors such as cameras and microphones, in order to collect a huge amount of data that can be processed, to better understand their surroundings. Advanced AI approaches, for data collection and processing, have been developed and deployed on the edge.

The proposed network is going to have a concrete user experience impact: a customer user is going to experience accurate guidance and product information (Figure 9 and Figure 10), as well as a retailer user is going to obtain a more efficient advertising marketing campaign.

Figure 8: A Totem in a shopping mall

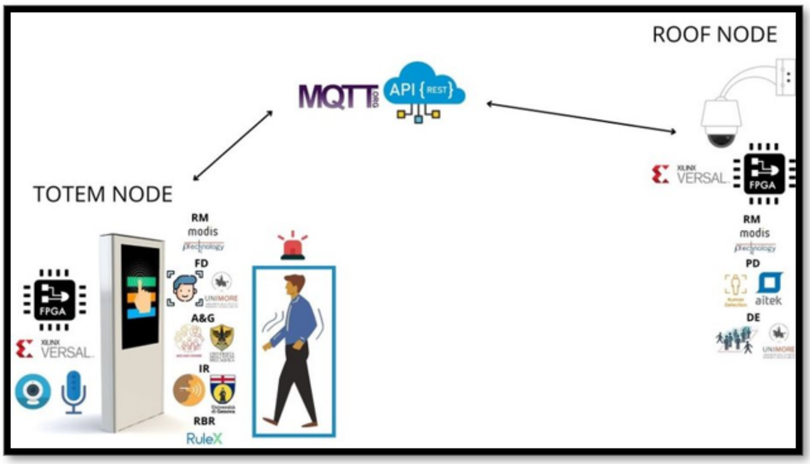


Figure 9: Smart Totem concept

Figure 10 - Totem providing customized ads to customer (picture by UC6 demo)

5.1 Results of the executed justification plan

5.1.1 Summary of results in Justification File

The results of "Justification of KPI from UC6", listed in Table 6, Table 7, and Table 8, show an overall fulfillment of all implementation KPIs, Features and Fractal Objective.

The remaining deviation was reduced to the minimum possible values thanks to continuous tests and enhancements in the single basic components (UC6_CMP_01 to UC6_CMP_09) and the overall architecture, without affecting the global score of the Use Case that meets all the Implementation and business KPIs provided.

In the Figure 11 and Figure 12 a summarized description of the Justification File is provided. In Figure 11 the taxonomy of the validation status of implementation, Feature and Objective Fractal's KPI is summarized, and in Figure 12 the overall implementation results according to the Tasks to be accomplished, Requirements covered, and coverage of Fractal features and objective, provide the general fulfillment of the UC6.

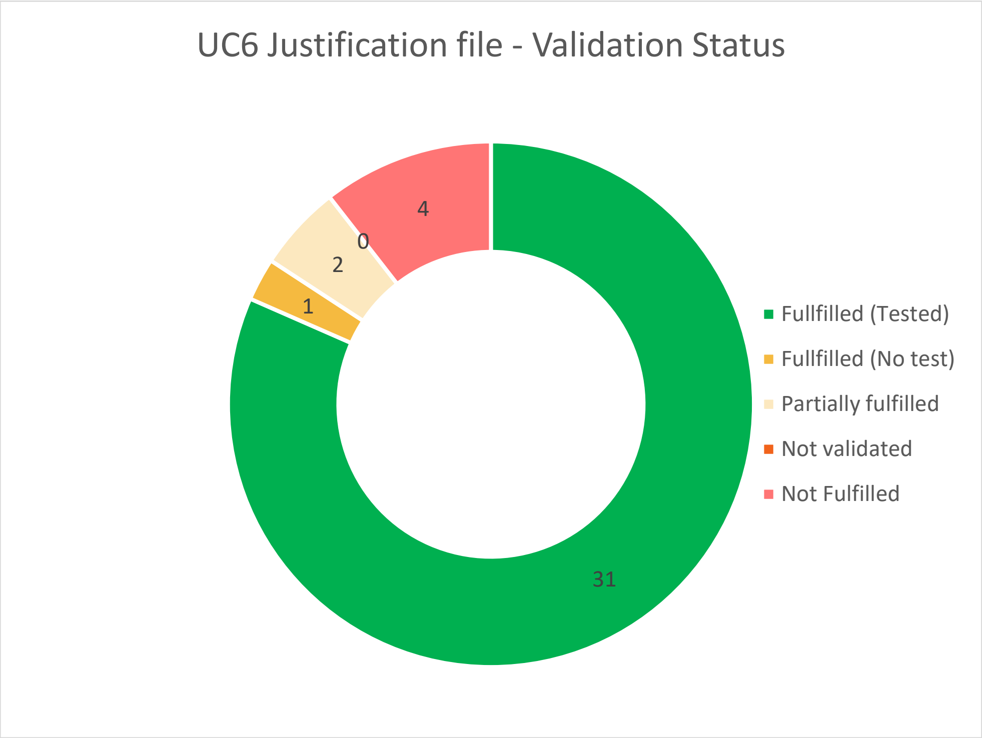


Figure 11 - Validation Status UC6

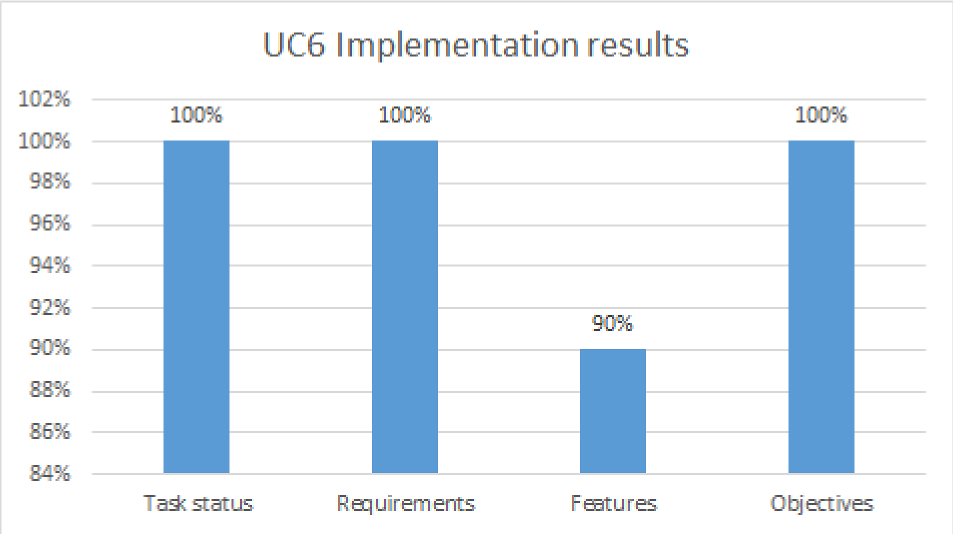


Figure 12 - Implementation results UC6

5.1.2 Implementation explanation to achieve the results

The implementation of full scenario, required a methodological approach in order to develop, test and refine the single components architecture, execution and performances, to eventually led the fully implementation of the Fractal UC6 solution.

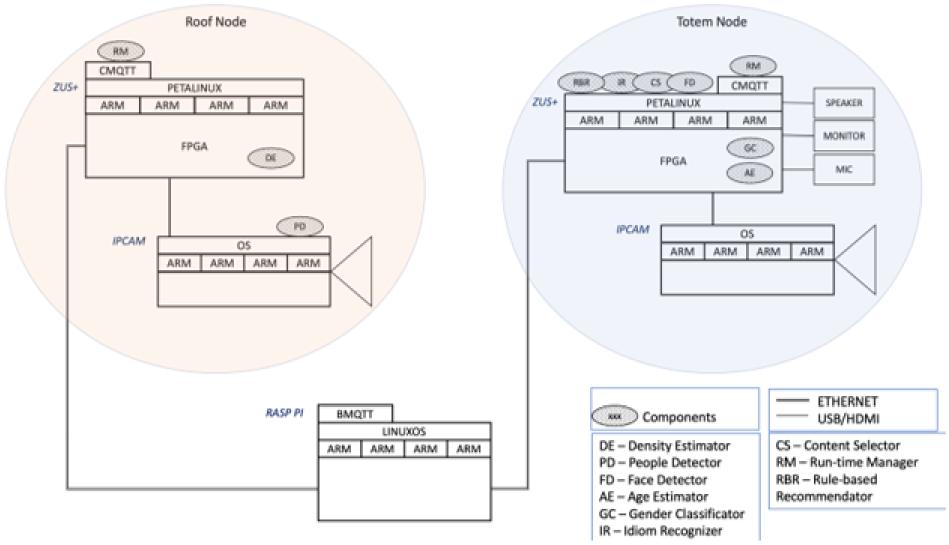


Figure 13 - UC6 architecture

A quick look to the overall architecture displayed in Figure 13 remarks the key components of the UC6 solution, made by a Roof and Totem were the Zynq UltraScale+ ZCU102 (ZUS+) perform the FRACTAL Node, and a third-party communication channel between two nodes, the external PC, was acting as Message Queuing Telemetry Transport protocol (MQTT) broker to publish Alarms and context awareness messages.

The main hardware we needed to validate the implementation of UC6 were:

- **2x Zynq UltraScale+ ZCU102** (ZUS+)**,** one to install all the components related to Totem Node, the second one for the Roof Node (Table 5).
- **2x IP CAMs**, for Audio and Video Streaming and connecting with the two ZUS+.
- **1x external PC,** to install MQTT Broker used to publish MQTT messages: Alarm and the context awareness.
- **1x Network Switch,** to interconnect the devices.

The basic components (from UC6_CMP_01 to UC6_CMP_09) populating the Totem and Roof node has been developed and tested at first alone in a controlled environment on Windows/Linux machines (desktop computers) to validate the overall architecture and performances, and then moved to the ZUS+ board. The integration of basic components on the Fractal node involved configuration and setup activities on the specific Linux distribution (PetaLinux provided by Xilinx) running on the board, enabling the full potential of ZUS+, using up to 3 Deep learning Processing Unit (DPU).

The verification and validation of single basic components was performed in a semi-structured way, identifying the related requirements, test conditions and test cases, planning, executing, and reporting them. This methodology has been exceptionally useful in first components integrations and let us easily reproduce the tests each time an unexpected result occurred and/or a fix was performed. The measures were taken by Python and Linux commands and scripts, both for single components and the overall orchestration.

To complete the full scenario the basic components were collected in the two boards according to Table 5, to verify that nodes behave in the same way and can then perform the final scenarios, orchestrating all components together.

Totem Node	Roof Node
WP3T32-07 Gender Classifier – GC	WP5T56-01 People Detector - PD
WP3T32-07 Age Estimator – AE	WP3T32-07 Gender Classifier – GC
WP3T35-05 Idiom Recognition – IR	WP3T32-07 Age Estimator – AE
WP3T36-02 Load Balancer – LB	WP3T36-02 Load Balancer – LB
WP6T62-03 Runtime Manager – RM	WP6T62-03 Runtime Manager – RM
UC6_CMP_03 – Face Detector – FD	UC6_CMP_01 – Density estimator – DE
UC6_CMP_08 – Rule-Based Recommender – RBR	

Table 5 - List of fractal components in UC6 solution

The final scenario with all components working altogether in a real, but controlled, environment was performed with a real and full Totem architecture that let us monitor both the overall performance and the single component one. Figure 14 and Figure 15 shows the actual setup to perform and evaluate the scenarios, and the evaluation boards up and running in the same session.

The expected results and the evidence of the Test Cases in the Justification File (Table 6, Table 7 and Table 8) required a run of all components in the real and controlled scenario, monitoring response time, accuracy, and other specific performance parameters of each basic component (from UC6_CMP_01 to UC6_CMP_09). The other KPIs, marked as System Test have been performed putting and running together all basic components to perform the specific characteristic of Smart Totem solution.

Figure 14 - Final demo setup in a real and controlled environment

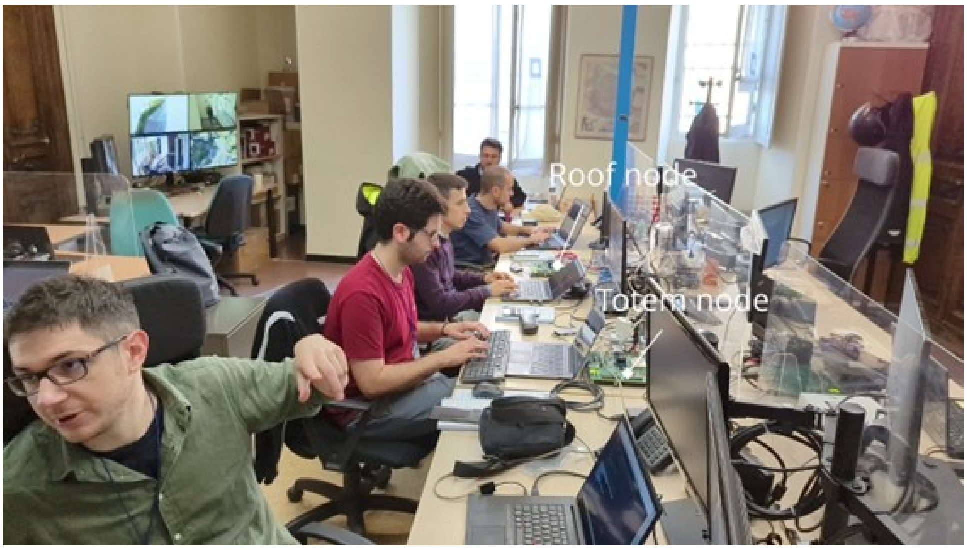


Figure 15 - One of the final testing phase of Smart Totem development

Justification of KPI Results (UC6)					
KPI ID	**Description**	**Validation Method**	**Evidence**	**Validation Status**	**Validation Comments**
UC6_KPI_IP_01	Density Estimator Implementation	Unit Test	UC6_T1	Fullfilled (Tested)	
UC6_KPI_IP_02	People Detector Implementation	Unit Test	UC6_T2	Fullfilled (Tested)	
UC6_KPI_IP_03	Face Detector Implementation	Unit Test	UC6_T3	Fullfilled (Tested)	
UC6_KPI_IP_04	Age Estimator Implementation	Unit Test	UC6_T4	Fullfilled (Tested)	
UC6_KPI_IP_05	Gender Classifier Implementation	Unit Test	UC6_T5	Fullfilled (Tested)	
UC6_KPI_IP_06	Idiom Recognizer Implementation	Unit Test	UC6_T6	Fullfilled (Tested)	
UC6_KPI_IP_07	Runtime Manager Implementation	Unit Test	UC6_T7	Fullfilled (Tested)	
UC6_KPI_IP_08	Rule-based Recommender Implementation	Unit Test	UC6_T8	Partially fulfilled	The strategy for building recommendations has been changed, eliminating the need for a machine learning training phase. The module has been implemented through a predefined rule system
UC6_KPI_IP_09	Data Compressor Implementation	Unit Test	UC6_T9	Fullfilled (Tested)	
UC6_KPI_IP_10	UC Components Integration	Integration Test	UC6_T10	Fullfilled (Tested)	
UC6_FT_00	The nodes are able to accelerate AI/ML models	System test	UC6_T1 UC6_T4 UC6_T5	Fullfilled (Tested)	

Table 6 - Justification of KPI Results from UC6 (Part 1)

UC6_FT_01	The nodes are able to perform inference in real-time	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5 UC6_T6 UC6_T8	Partially fulfilled	The UC is partialluy fulfilled under the conditions of the UC6_T8.
UC6_FT_02	The nodes are able to import and execute ONNX models	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5	Fullfilled (Tested)	
UC6_FT_03	The nodes are able to import and execute VERSAL models	System test	UC6_T1 UC6_T4 UC6_T5	Fullfilled (Tested)	
UC6_FT_04	The nodes are able to perform inference	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5 UC6_T6 UC6_T8	Fullfilled (Tested)	
UC6_FT_05	The nodes are able to exploit offline learning/training	System test		Not fulfilled	According to the interim results of the implementation, this feature was no longer needed by the UC.
UC6_FT_06	The nodes are able to exploit supervised learning/training	System test		Not fulfilled	According to the interim results of the implementation, this feature was no longer needed by the UC.
UC6_FT_07	The nodes are able to exploit CNN	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5	Fullfilled (Tested)	
UC6_FT_08	The nodes are able to exploit TENSORFLOW/KERAS libraries	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5	Fullfilled (Tested)	
UC6_FT_09	The nodes are able to perform load balancing	System test	UC6_T7 UC6_T9 UC6_T10	Fullfilled (Tested)	
UC6_FT_10	The nodes are able to monitor their performances	System test	UC6_T7 UC6_T8	Fullfilled (Tested)	
UC6_FT_11	The nodes can acquire video streams from a camera	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5	Fullfilled (Tested)	
UC6_FT_12	The nodes can acquire audio streams from a microphone	System test	UC6_T6	Fullfilled (Tested)	
UC6_FT_13	The nodes can generate and trasmit alarms	System test	UC6_T2	Fullfilled (Tested)	
UC6_FT_14	The nodes have Ethernet interface	Unit Test	N.A.	Fullfilled (Tested)	
UC6_FT_15	The nodes have WI-FI interface	Unit Test	N.A.	Not fulfilled	According to the interim results of the implementation, WIFI feature would not bring any innovations to the PoC of UC. The team decided to focus mainly to solve major issues to finalize the PoC itself.

Table 7 - Justification of KPI Results from UC6 (Part 2)

UC6_FT_16	The nodes support MQTT communication	System test	UC6_T1 UC6_T2 UC6_T7 UC6_T10	Fullfilled (Tested)	
UC6_FT_17	The nodes are implemented on Versal	System test	UC6_T1 UC6_T2 UC6_T7 UC6_T10	Not fulfilled	Due to the well-known purchasing issues (i.e., pandemic and SoCs shortage), the VERSAL board has been available very late so, all the UC components have been ported on it, but the full integration (i.e., the one used for the live demos) has been completed only for ZUS+102.
UC6_FT_18	The nodes are implemented on ZYNQ ULTRASCALE+	System test	UC6_T10	Fullfilled (Tested)	
UC6_FT_19	The nodes execute LINUX OS	System test	N.A.	Fullfilled (No test)	
UC6_FO_00	The nodes are able to detect users features	System test	UC6_T3 UC6_T4 UC6_T5 UC6_T6	Fullfilled (Tested)	
UC6_FO_01	The nodes are able to detect users activities	System test	UC6_T1 UC6_T2	Fullfilled (Tested)	
UC6_FO_02	The nodes are able to monitor their status	System test	UC6_T7	Fullfilled (Tested)	
UC6_FO_03	The nodes are able to monitor their performances	System test	UC6_T7 UC6_T8	Fullfilled (Tested)	
UC6_FO_04	The nodes are able to share the workload among them	System test	UC6_T7 UC6_T9 UC6_T10	Fullfilled (Tested)	
UC6_KPI_IP_Req_01	Cognitiveness Reqs	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5	Fullfilled (Tested)	Related to: REQ_UC6_01, REQ_UC6_03, REQ_UC6_04, REQ_UC6_05, REQ_UC6_06, REQ_UC6_07, REQ_UC6_08, REQ_UC6_09, REQ_UC6_16, REQ_UC6_21
UC6_KPI_IP_Req_02	Monitoring&Management Reqs	System test	UC6_T1 UC6_T2 UC6_T3 UC6_T4 UC6_T5 UC6_T6 UC6_T7 UC6_T8 UC6_T9 UC6_T10	Fullfilled (Tested)	Related to: REQ_UC6_02, REQ_UC6_11, REQ_UC6_12, REQ_UC6_13, REQ_UC6_14, REQ_UC6_15, REQ_UC6_17, REQ_UC6_18, REQ_UC6_19, REQ_UC6_20, REQ_UC6_22, REQ_UC6_23, REQ_UC6_24, REQ_UC6_25, REQ_UC6_27, REQ_UC6_28
UC6_KPI_IP_Req_03	User Experience Reqs	System test	UC6_T7 UC6_T8 UC6_T10	Fullfilled (Tested)	Related to: REQ_UC6_10, REQ_UC6_26

Table 8 - Justification of KPI Results from UC6 (Part 3)

5.2 Results of the executed benchmark

The solution proposed by Smart Totem in UC6 is the first one in its field, that brings up to three AI instances in a single Totem; moreover, the ability to share computational load with other nodes is new in retail industry. For this motivation, no caparisons have been possible.

5.3 Evaluation of the results

5.3.1 Evaluation of Business KPIs

KPI ID	Description	Assessment methodology	Baseline	Target	Expected improvement
KPI-1 max number of users	Number of users simultaneously considered while assuring response time strictly minor than 1s.	Experimental evaluation on lab prototype	1	6	500%
KPI-2 Energy saving	The totem node can be activated only when someone approaches the totem area	Estimation based on historical data about people flow	Totem active for 12 hours a day	Totem active for 4/12 hours a day	64%
KPI-3 Attractivity of Totem	Percentage of users approaching the totem area that actually use the totem.	Comparison with traditional totem (no adaptive content)	To be evaluated evaluated during the exploitation phase		

Table 9 - "KPI for Business Improvement" for the UC6

5.3.1.1 Business KPI – 1: Max number of users

The computational power of ZUS+ board and the achievements of the FRACTAL components mainly in People, Face Detector, Rule Base Recommender and in workload sharing, led to the processing of multiple faces at once without impact on execution time. This achievement is a big innovation for business because it improves the throughput of people that can be addressed by Totem ads, maximizing the effectiveness of those ads for that specific group of people: the Totem will select the right ads according to the specific group composition based on age, gender and idiom.

The max number of people the Smart Totem can manage without affecting the processing time is six, then we have an improvement of 500%. This KPI together with the group composition reflects on effectiveness of ads for the retailers that spans from 200% up to 500%. If the group components, as a whole, were classified to be in the same average range, we have addressed 6 people instead only one, so the ads became more effective because have reached 6 people instead of 1. If there is a real majority, for example, if the group is composed by one young person and five adults (see Table 10) then the ads will effectively reach 5 people instead of the one (1) left, so the effectiveness from retailer's standpoint has increased of 400%, and so on.

Baseline	Group made by 6 people Minority	Majority (people with same features)	Enhancements in effectiveness related to baseline of 1 people
n. People	n. People	n. People	
1	0	6	500%
	1	5	400%
	2	4	300%
	3	3	200%

Table 10 - Enhancements in effectiveness of ads for retailers

5.3.1.2 *Business KPI – 2: Energy saving*

Shopping malls are the most suitable place to exploit the peculiar features of advisory Totems in order to extend the retailers' opportunities to grow their business. This doesn't come without costs even for the energy part, so an increment in energy savings will be immediately appreciated. In this scenario the Fractal solution plays a major role in medium and small malls.

CASE	Shopping mall	Size	Data collection period	Reference period	Number of people in reference period	Number of people per single DAY
1-MILANO	CENTRO IL CENTRO https://centroilcentro.it/en/	Very large 200 Shops	April, 2017	12 months	13 000 000	**35 615**
2-BOLOGNA	FICO Etaly World www.fico.it	Medium Large 144 Shops	April, 2022	12 months	400 000	**1 096**

Table 11 - Two cases of shopping mall in Italy

Leveraging on historical data of people flow in medium and large shopping mall, together with the average period for a person to read the ads on Totem screen, we can compute the average percentage of energy saving.

Let's take a look at two major shopping malls in Italy (Table 11), and consider that in average, people stand in front of Totem to read advices for about 10s. Adding the registration time (3s) and the elaboration phase (1s), the overall process can take about 14s.

Considering the worst-case scenario with its processing time, with one person at time approaching the Totem and with no idle time, the Totem can process max 3042 people in a day (within 12 hours of shopping mall opening). This is the upper limit over which no energy saving happens, and this is the case of crowded shopping mall like CASE 1 – MILANO in Table 11.

On the other hand, medium shopping mall like CASE 2 – BOLOGNA in Table 3 can take advantage from FRACTAL solution because it may save at least 64% of Totem's time of activity. Figure 16 summarizes the variable percentage of energy savings according to the number of people approaching the Totem.

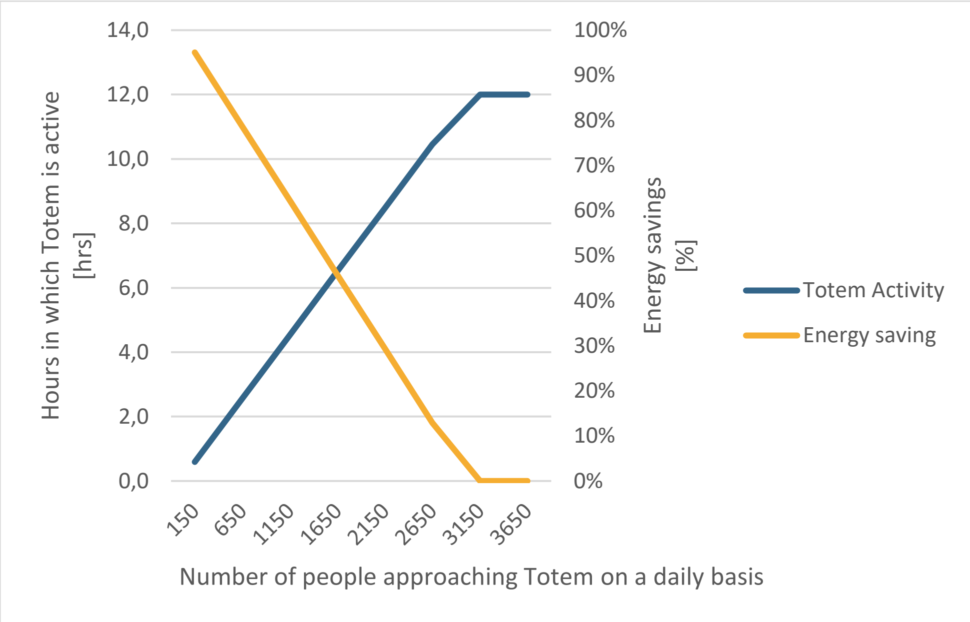


Figure 16 - Energy savings according to number of people processed by Totem in the simple scenario

5.3.1.3 Business KPI – 3: Attractivity of Totem

In past years, static and advertising signage, represented essentially by posters and billboards, was increasingly being replaced by flat screens and digital content advertising totems of different sizes, usually networked, controlled in remotely, often via the Internet. Once selected the right ads schedule and type of ads, they run on the screen in a cyclic way, no matter when they are playing, and no matter if there are or not people around and wasting energy.

With the current technology no interaction is allowed with the people, apart the menu-based approaches ones and no intelligent behavior may catch the interest of people passing by due to this "monolithic" approach and its hardware computational capabilities. Nevertheless, according to Nielsen study "Digital Out-of-home Advertising Report 2020", 80% of consumers involved, say they entered a store because a screen caught their eye.

With Fractal node, the Totem become "Smart" due to change of computational paradigm allowing up to three AI components to be run in a single hardware board. The Totem so can adapt its content based on main characteristics of people approaching it (Figure 7 and Figure 8), increasing its attractiveness on people passing by, because a "real" interaction can be meet.

The Smart Totem UC6 solution aims to improve this scenario offering a customer a specific experience with the use of AI algorithms to address more likely interests of customers based on Age, Gender, Idiom. We expect an increment of people attracted by Smart Totem, not because they just see it, but because they have an interaction with Totem. We expect on the side an efficiency in purchasing because the specific content provided by the Totem is expected to be the right one for that specific person who was approaching.

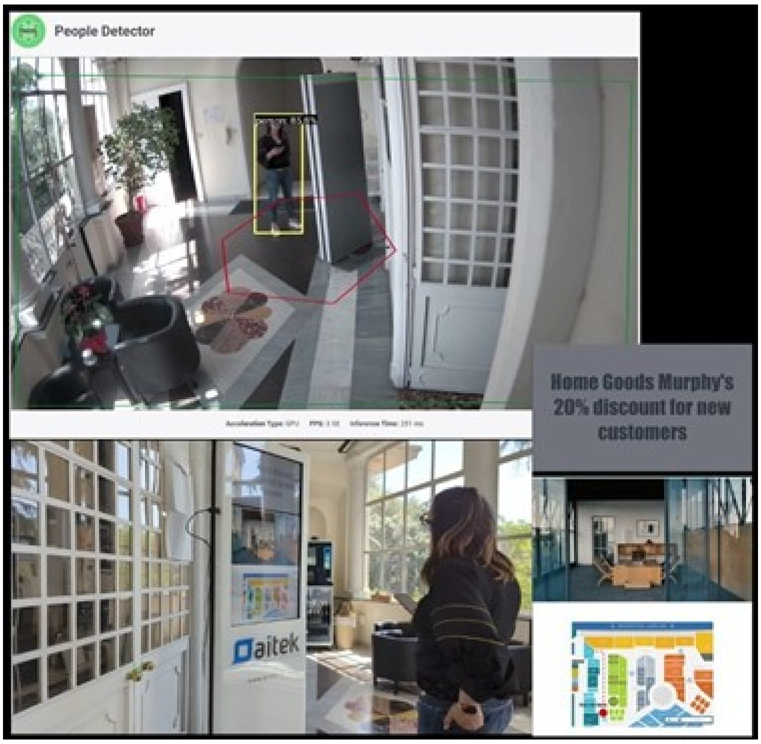


Figure 17- UC6 example of adaptivity of ads displayed on Totem Screen

5.3.2 Discussion of the results

The results shown by the Justification File (Table 6, Table 7 and Table 8) describe a situation in which the 89% of KPIs were met and 11% (only 4 out of 38) was not fulfilled.

The achievements of the UC6 and the test performed on the System lays on the performances of single components and over the orchestration parts.

From a high-level perspective the most evident achievement is about the user experience and the response time. This is the result of the orchestration of all FRACTAL nodes involved and all the UC's components. In a FRACTAL System based on only two nodes, it can be possible to process from 1 to 6 images. Figure 18 shows the measured results of system's response time from face detection from a dedicated ads showed on the Totem screen. It is noteworthy to highlight that the max deviation from the target of 1s is because Totem needs manage the communication with the second node (the Roof node). By growing the number of images, Totem and Roofs can start collaborating: Roof node reschedule its internal activities in order to share the workload with the Totem while sends images and get back the results.

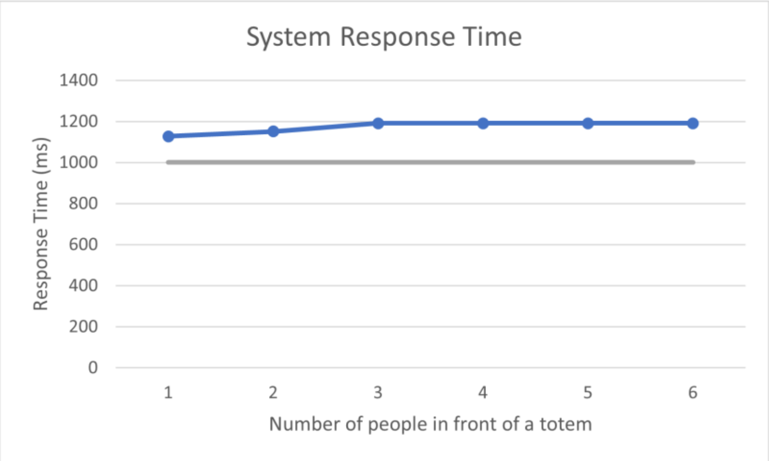


Figure 18 - Smart Totem System response time

Moreover, in a system made of 4 nodes it is possible to manage up to 18 images: over that number, effects of communication become predominant, and no more images can be processed under the deadline of 1s (Figure 19).

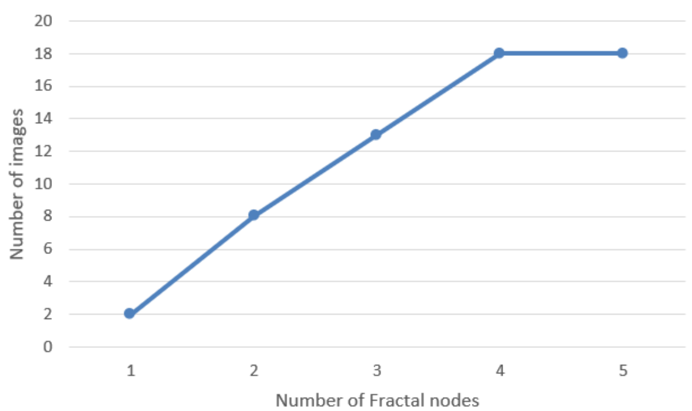


Figure 19 - Number of images that can be processed from a N-nodes system, fulfilling the 1s target

Nodes are capable to perform the offloading thanks to the Runtime Manager and the Load balancing modules: this is the case in which a node recognizes that cannot provide feedback within 1s (Figure 21).

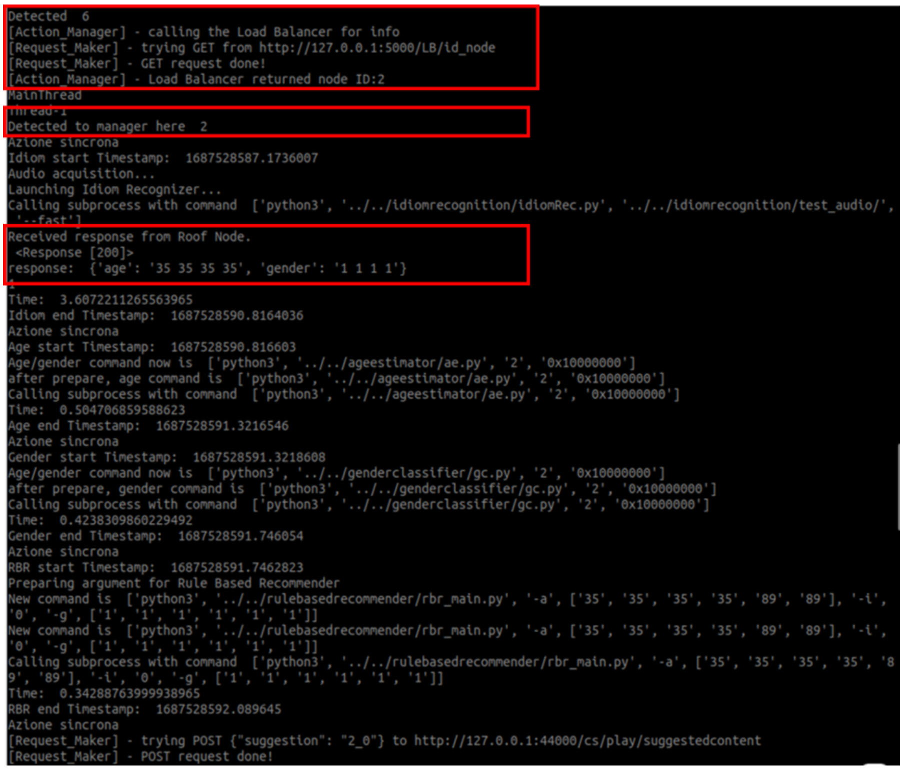


Figure 20: Runtime Manager in action

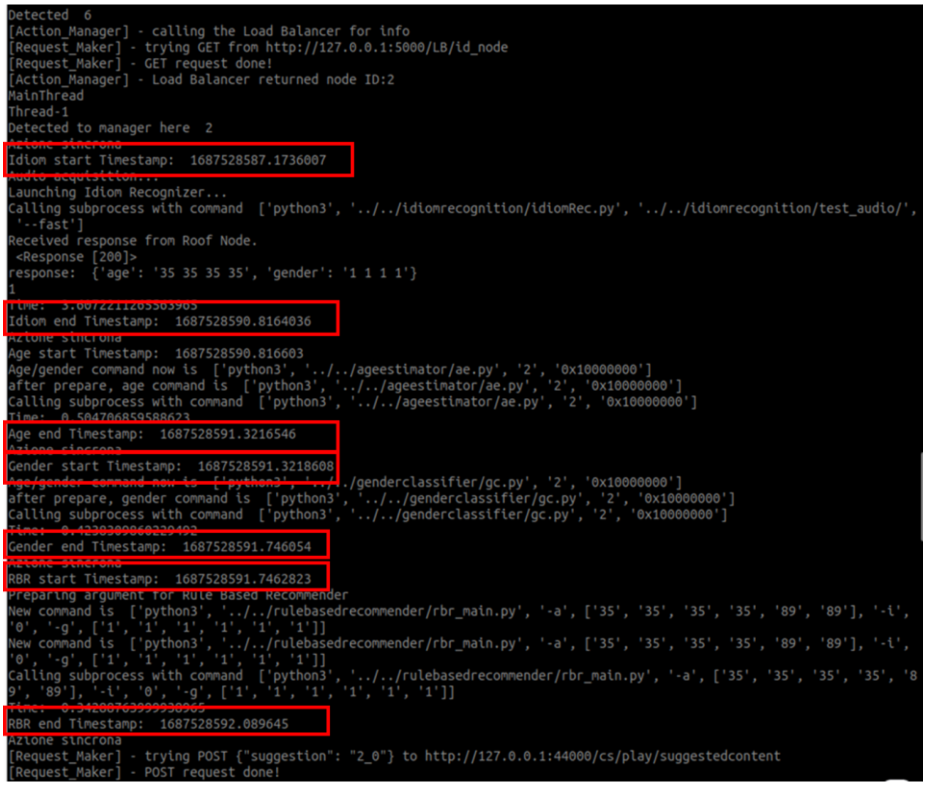


Figure 21: Decision starting workload

To achieve the results of adapting the ads to different people with different languages (two languages at the moment) the UC needs to host different number of AI instances. We tested a 2-nodes System providing input images from 0 to 6: for each one we can see in Figure 22 the overall number of AI instance in Totem and Roof node and in Figure 23, Figure 24 and Figure 25 the corresponding results.

Figure 22 - Number of AI instance over images

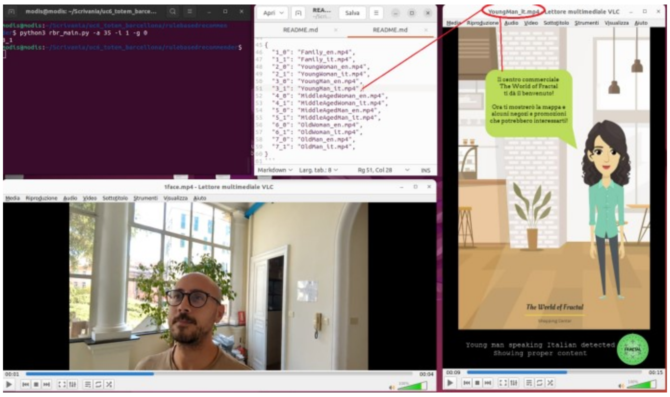


Figure 23 - Detecting 1 person, ITA speaking, young man

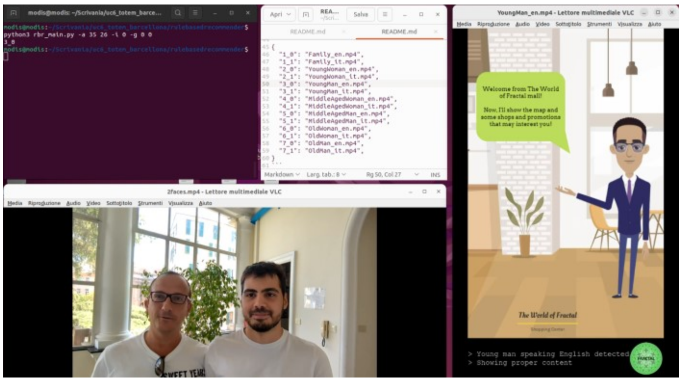


Figure 24 - Detecting 2 people, ENG speaking, young man

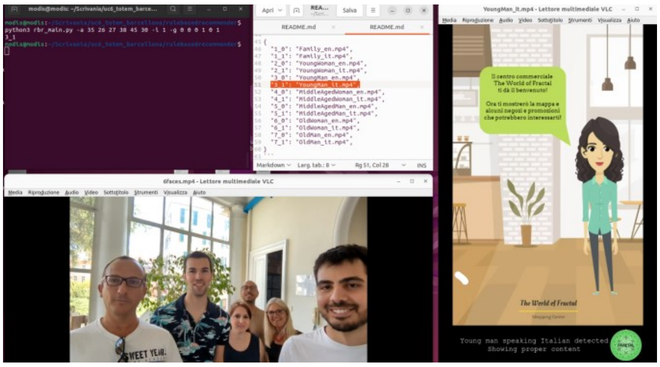


Figure 25 - Detecting 6 people, ITA speaking, young man

HIGHLIGTHS	LOWLIGHTS
Dynamic and tailored content management (features-based) exploiting soft real-time AI-based video and audio analytics at the Edge.	Due to the well-known purchasing issues (i.e., pandemic and SoCs shortage), the VERSAL board has been available very late so, all the UC components have been ported on it, but the full integration (i.e., the one used for the live demos) has been completed only for ZUS+102.
Multiuser-interactions thanks to a use component (the Runtime Manager – WP6T62-03) in charge of load distribution at the Edge, among many other functionalities.	Offline and supervised learning cannot be fulfilled.
Smart management policies in terms of event oriented (i.e. activity-based) components activation.	WiFi interface not present. This not affect the proof of concepts of Smart Totem.

Table 12 - Highlights and Lowlights of UC6

5.4 Consideration of safety and security

Safety and security are major topic in all technology solutions affecting real case scenarios. The concept of Safety, intended as practices to avoid any human incident or machinery failure prevention measures, are embraced by product designers, and planned to deal in hazardous situations.

From this perspective the FRACTAL UC6 scenario does not add any safety risk to further consider, because the relevant safety goals have already considered by the OEMs providing the basic hardware (Totem, Cameras, sensors and so on). The additions, in this case, are in the software area using existing hardware.

On the other hand, security is intended to protect both the hardware and data assets in the machine. To protect the machines from malfunctions due, for example, to Denial of Service (DoS) attacks are major issues in connected systems; to protect temporary data frames cached by the cameras are also a security goal to consider. To address the latter issue, the frames are not retained more than needed processing time, avoiding collecting historical data that may extend the attack surface.

Future developments on security may address the prevention of sniffing frames in the communication from Totem and Roof nodes, preventing the Man-In-The-Middle attack, using cryptographic approaches with the AES standard or, for securing the communication between Totem (or roof) and the Cloud, it can be used a standard SSH or HTTPS connection together with AES encryption.

5.5 Preparation for realization of commercial products

According to major strategy consultancy firms, organizations face an urgent need to revision their tech architectures and business model to address the market's incoming changing landscape.

Focusing on retail industry, many retailers are now facing the challenge of new customer's habits, by the rise of digital technology that reshaped the customer's behavior making use of omnichannel and hyper-personalization, accelerated by the pandemic. So, a robust technology architecture that may lead to a brand-new business model can be used to give the retailers more opportunity to make their business grow.

The FRACTAL UC6 let the retailers to grow in this direction, to become more responsive to these trends, enabling several areas of next-generation retail.

Taking as starting point the Use Case 6, the FRACTAL solution may support the integration of online and offline channels with smart digital services that facilitate end-to-end customer decision journeys. Reliable, personalized offerings that have been optimized through advanced analytics can be displayed in close to real time and supported by attractive digital content.

The transformation of a shopping mall in a *sentient space* then multiplies the possibilities of advertising people with the right content according to their characteristics (age, gender, idiom), avoiding broadcasting costly messages to anyone who does not actually care. In such a way we hope to rise the effectiveness of ads policies, and then give the retailers the opportunities they deserve. Furthermore, FRACTAL UC6 solution comes with a reduction of energy costs, due to the system powering-up only when people approach to the Totem, saving energy in uncrowded hours.

The ease of installation and use, let the FRACTAL UC6 Smart Totem and Roof solution suitable for shopping mall, airports, and train station, with very little modifies according to the context.

Further advancements may extend the functionalities for increasing customer data to address the hyper-personalization, that can affect and improve the customer's experience and then the retail business models beyond the traditional core business to generate additional revenues.

6 VAL-UC7 Autonomous robot for implementing safe movements

The "Smart Physical Demonstration and Evaluation Robot" (SPIDER)[1] is an autonomous robot prototype developed by Virtual Vehicle Research. It is used as a Mobile Hardware-in-the-Loop (HIL) platform for testing and verification of sensors and automated driving functions, mainly in the automotive sector. The SPIDER can imitate driving behavior of a vehicle under tests using its omnidirectional wheels and flexible mounting rots for placing test equipment, like sensors.

Figure 26 – Smart Physical Demonstration and Evaluation Robot (SPIDER)

The SPIDER is used as demonstration vehicle to evaluate the two **objectives** of the VAL_UC7, which are:

1. Co-execution of safety-relevant, security-relevant, as well as AI based tasks,
2. Guarantee extra functionality of fail-operational capabilities.

To fulfill those objectives, two vehicle **functions** were implemented on a NOEL-V based hardware platform. Those are:

1. The collision avoidance function, a safety critical function, which stops the vehicle in case of any approaching obstacle on the path,
2. The path tracking function, using a ML algorithm, which follows a planned path and evades static obstacles.

For implementation of the vehicle functions, FRACTAL components from WP3 and WP4 providing capabilities for safety, security, and ML were used.

[1] https://www.v2c2.at/spider/

6.1 Results of the executed justification plan

Within the use-case two vehicle functions, the collision avoidance, and the path tracking function, were implemented on a NOEL-V based platform and tested first in simulation and later at the SPIDER robot platform. The KPI from Table 13 and Table 14 are evaluated by the testcases in Appendix A: Test Cases.

6.1.1 Training of AI model

The path tracking function is trained with a reinforcement learning approach, a specific type of machine learning that keeps learning from new and continuous inputs. The model was trained using the Tensorflow Python library and evaluate with a simple bicycle model implemented in Python. Input to the model is a grid containing the obstacle information, and the error of the robot to the path. A sample result of the evaluation is shown in Figure 27. In the next step the model is converted to an ONNX format file using an automated conversion script in Jupyter Notebook software.

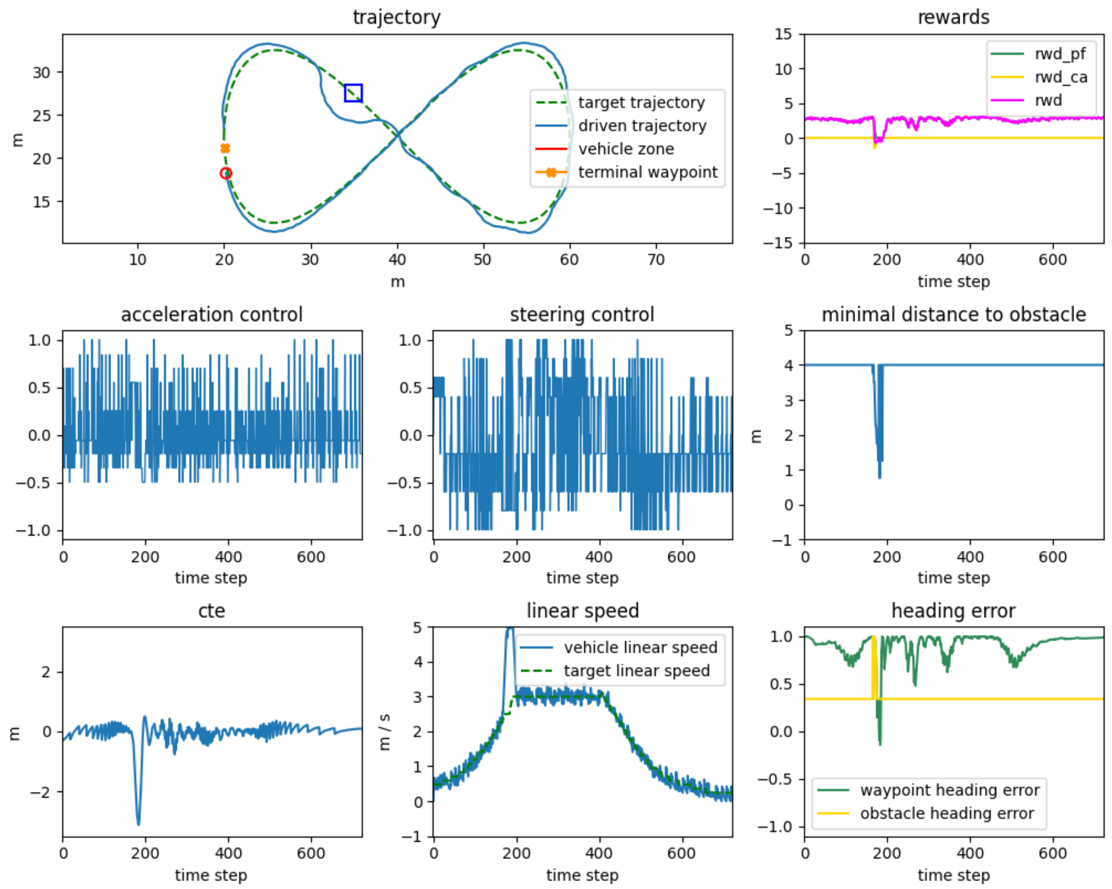


Figure 27 - UC7 model validation using a simple bicycle simulation

6.1.2 Evaluation in simulation

Both, path tracking and collision avoidance function are implemented in Linux using the ROS2 middleware and C++. This middleware allows a seamless integration to the overall SPIDER software architecture. For validation of the implemented functions a 3D simulation was prepared using the Gazebo Simulation software. This simulation provides more realistic vehicle dynamics, than the simple bicycle model implemented in the Python environment. Further, the used lidar sensors can be simulated to test the complete pipeline from perception to control and actuation of the vehicle. A sample image from the simulation is shown in Figure 28.

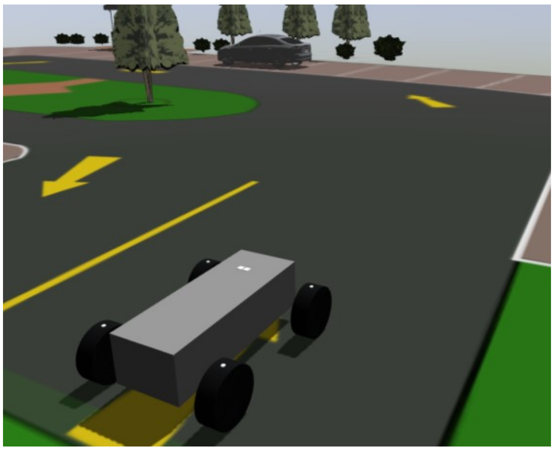


Figure 28 - SPIDER driving in Gazebo 3D simulation

The path tracking node loads the ONNX file, including its trained weights, to the LEDEL library for processing inference. The collision avoidance node uses the diverse redundancy library to execute the processing of the collision detection in redundant and diverse processes. Further, processing load on the critical core is monitored using the EdgeSU monitoring unit from the FRACTAL developments.

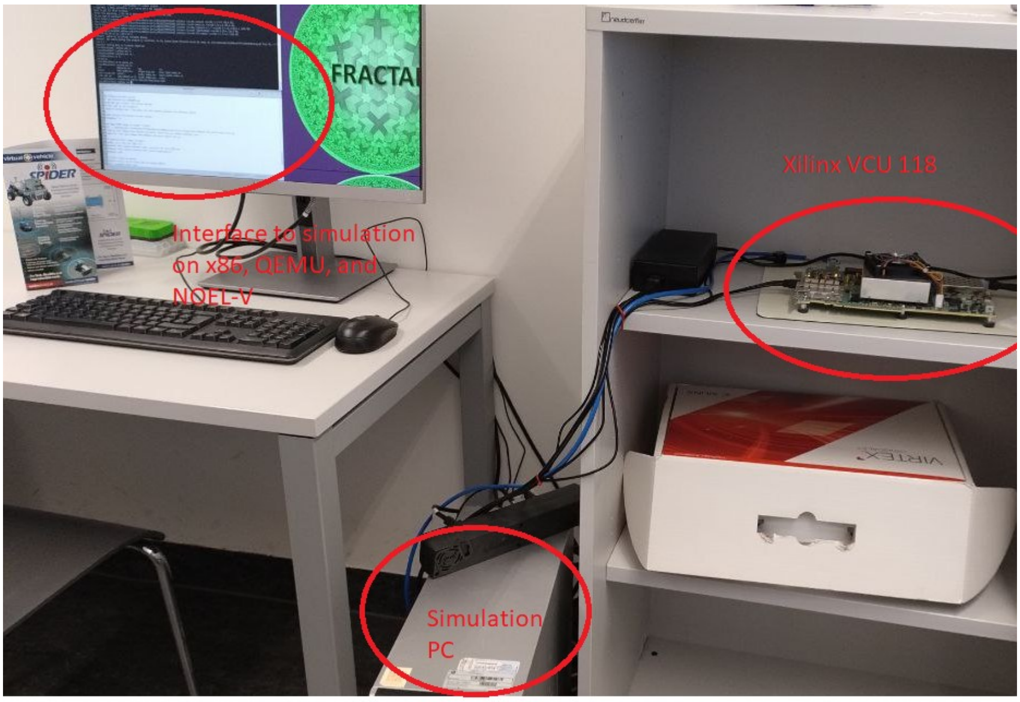


Figure 29 - UC7 Development setup

The simulation is used to evaluate the defined KPI at a state, near to the real hardware. The software is kept unchanged when switching to the robot hardware. Only the quality of the sensor data, robot positioning, and hardware actuation will lead to a different timing situation.

The hardware setup of the development environment is shown in Figure 29.

6.1.3 Evaluation with SPIDER hardware

The last step in the evaluation are the tests on the SPIDER robot using the NOEL-V hardware. The functions are ported from x86 to the NOEL-V hardware running Linux and executed at a Xilinx VCU118 FPGA. The FPGA is connected via ethernet to the SPIDER robot.

For an intermediate testing of the ported software, a QEMU simulation of the NOEL-V architecture was used to perform the tests defined in Appendix A: Test Cases. To evaluate capabilities of the FRACTAL diverse redundancy component, the FPGA fault injector component was used, as described in Appendix B: FPGA fault injection to NOEL-V (VAL_UC7). In the next step the NOEL-V hardware was electrically integrated to the SPIDER robot and tests using the simulation were performed in the garage, jacked up for safety reasons, as seen in Figure 30.

Figure 30 - SPIDER hardware tests, jacked up in garage

In the final step, a trajectory was planned for testing the SPIDER on the proving ground using satellite images, displayed in Figure 31. An x86 PC was used as replacement of the NOEL-V hardware for power supply reasons, running the software in a QEMU simulation. The SPIDER drives these predefined scenarios for evaluation of the path tracking and collision avoidance function on a closed testing area, Figure 32. The localization system is based on a dual-antenna, differential RTK system, incorporating relative movements from IMU sensor and vehicle odometry data in an extended Kalman filter. The required cost map calculation for obstacle information to the collision avoidance and path tracking functions are gathered from four 16-line Lidar sensors, were the field of view always overlaps by at least two sensors.

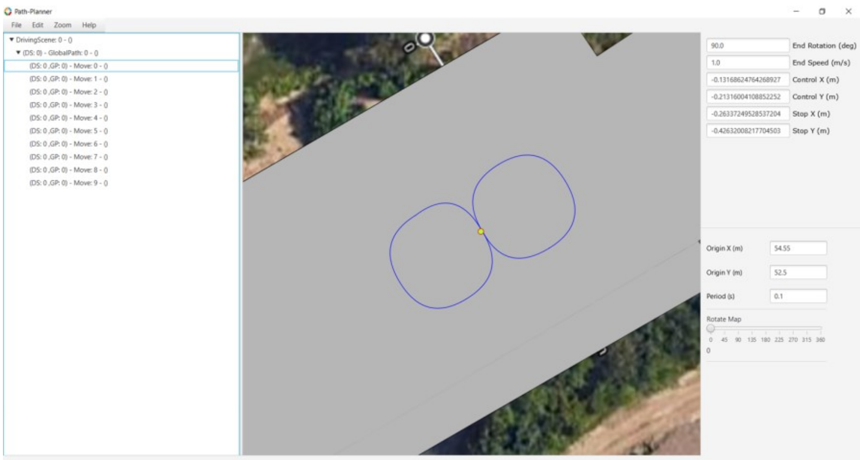


Figure 31 - SPIDER path planning based on satellite images

Figure 32 - SPIDER tests on proving ground

The tests have proven that the KPI of the two functions could be met under real world conditions. The measured distance to the path is at maximum ~0.5 meter larger to the path, as we have seen it in the simulation. This behavior was expected due to errors in the localization system and delays in the actuation of the steering and motor controllers.

Justification of KPI Results (UC7)					
KPI ID	Description	Validation Method	Evidence	Validation Status	Validation Comments
UC7_KPI_IP_01	All subtask success.	-	-	Fullfilled (No test)	
UC7_KPI_IP_02	Linux on NOEL-V is booting on FPGA.	Integration Test	UC7_T1	Fullfilled (Tested)	
UC7_KPI_IP_03	Simple publisher/subscriber example is running on target platform.	Integration Test	UC7_T2	Fullfilled (Tested)	
UC7_KPI_IP_04	Max data transfer rate deviation of 10 Hz	Unit Test	UC7_T3	Fullfilled (Tested)	
UC7_KPI_IP_05	All subtask success.	-	-	Fullfilled (Tested)	
UC7_KPI_IP_06	Simulated robot is following trajectory and avoiding obstacles.	Simulation	UC7_T4	Fullfilled (Tested)	
UC7_KPI_IP_07	Avg. Path Proximity in meter	Simulation	UC7_T5	Fullfilled (Tested)	
UC7_KPI_IP_08	Collision free rate	Simulation	UC7_T6	Fullfilled (Tested)	
UC7_KPI_IP_09	Valid ONNX model	Integration Test	UC7_T8	Fullfilled (Tested)	
UC7_KPI_IP_10	Unit test coverage of Path Tracking Function	Unit Test	UC7_T9	Fullfilled (Tested)	
UC7_KPI_IP_11	Unit test coverage of Collision Avoidance Function	Unit Test	UC7_T10	Fullfilled (Tested)	
UC7_KPI_IP_12	Loop rate of Collision Avoidance Function	Integration Test	UC7_T11	Fullfilled (Tested)	
UC7_KPI_IP_13	Resource monitoring tests in simulation sucessful	Integration Test	UC7_T12	Fullfilled (Tested)	
UC7_KPI_IP_14	Redundancy library tests in simulation successful	Integration Test	UC7_T13	Fullfilled (Tested)	
UC7_KPI_IP_15	All subtask success.	-	-	Partially fulfilled	See subtasks
UC7_KPI_IP_16	Functions on target platform running with sensor data from 3D simulation	Simulation	UC7_T14	Fullfilled (Tested)	
UC7_KPI_IP_17	Functions on target platform running with sensor data from real world tests	System Test	UC7_T15	Fullfilled (Tested)	
UC7_KPI_IP_18	Metrics calculated with Jupyter availaible	Analysis	UC7_T16	Fullfilled (Tested)	
UC7_KPI_FO_01	FRACTAL path tracking node accelerated to perform with a high frequency	Integration Test	UC7_T17	Fullfilled (Tested)	
UC7_KPI_FO_02	Tests in simulation for redundant execution and monitoring succeed	Simulation	UC7_T12 UC7_T13	Fullfilled (Tested)	
UC7_KPI_FO_03	FRACTAL path tracking nodes AI model generates a collision free path with and acceptable path proximity	Simulation	UC7_T5 UC7_T6	Fullfilled (Tested)	
UC7_KPI_FO_04	Framework for platform independent development and verification of node functions availaile	Integration Test	UC7_T18	Fullfilled (Tested)	
UC7_KPI_FT_01	Target platform supports ONNX	Integration Test	UC7_T19	Fullfilled (Tested)	
UC7_KPI_FT_02	Path tracking function AI model executed at node level	Integration Test	UC7_T20	Fullfilled (Tested)	
UC7_KPI_FT_03	Reinforcement learning approach trained model path proximity	Simulation	UC7_T5	Fullfilled (Tested)	
UC7_KPI_FT_04	Reinforcement learning approach trained collision avoidance	Simulation	UC7_T6	Fullfilled (Tested)	
UC7_KPI_FT_05	LEDEL library available for target platform	Integration Test	UC7_T19	Fullfilled (No test)	
UC7_KPI_FT_06	Sensor data from test drives can be stored on hard drive	System Test	UC7_T21	Fullfilled (Tested)	
UC7_KPI_FT_07	Frame rate of collision avoidance function	Integration Test	UC7_T11	Fullfilled (Tested)	
UC7_KPI_FT_08	Switch to emergency state at time exceedance of AI function	System Test	UC7_T22	Fullfilled (Tested)	
UC7_KPI_FT_09	Switch to emergency state at time exceedance of safety relevant function	System Test	UC7_T22	Fullfilled (Tested)	
UC7_KPI_FT_10	Safety relevant processes run redundant	-	-	Deleted	Not required due to diverse
UC7_KPI_FT_11	Switch to emergency state at fault detected by diverse redundancy model	System Test	UC7_T23	Fullfilled (Tested)	
UC7_KPI_FT_12	Switch to emergency state at fault detected in the communication messages	-	-	Fullfilled (No test)	Integrity provided by the used middleware
UC7_KPI_FT_13	Safety concept according to ISO 26262 availaible	-	-	Fullfilled (No test)	Integrity provided by the used middleware
UC7_KPI_FT_14	Lidar sensor messages availaible at target platform at data rate	Integration Test	UC7_T15	Fullfilled (Tested)	
UC7_KPI_FT_15	Path tracking node tested in target platform on proving ground	System Test	UC7_T24	Partially fulfilled	x86 PC was used for proving ground tests
UC7_KPI_FT_16	Security assessment according ISO SAE 21434 availaible	-	-	Fullfilled (No test)	WP3
UC7_KPI_FT_17	Max data transfer rate with ethernet, deviation of 10 Hz	Integration Test	UC7_T3	Fullfilled (Tested)	
UC7_KPI_FT_18	Target RISC-V hardware platform based on NOEL-V available	Integration Test	UC7_T1	Fullfilled (Tested)	
UC7_KPI_FT_19	Linux operating system running on target platform	Integration Test	UC7_T1	Fullfilled (Tested)	

Table 13 - Justification of KPI Results from UC7 (Part 1)

UC7_KPI_IP_Req_01	Processing time of costmap distance	Integration Test	UC7_T11 UC7_T12 UC7_T22 UC7_T23	Fullfilled (Tested)	
UC7_KPI_IP_Req_02	SPIDER stops in defined emergency situation	System Test	UC7_T25	Fullfilled (Tested)	
UC7_KPI_IP_Req_03	Avg. Path Proximity in meter of the path tracking node	Simulation	UC7_T5	Fullfilled (Tested)	
UC7_KPI_IP_Req_04	Collision free rate of the path tracking node	Simulation	UC7_T6	Fullfilled (Tested)	
UC7_KPI_IP_Req_05	SPIDER stops at connection loss to edge nodes	System Test	UC7_T22	Fullfilled (Tested)	
UC7_KPI_IP_Req_06	SPIDER stops at timeout of edge nodes	System Test	UC7_T22	Fullfilled (Tested)	
UC7_KPI_IP_Req_07	Update rate of costmap input data to edge nodes	Integration Test	UC7_T15	Fullfilled (Tested)	
UC7_KPI_IP_Req_08	Edge nodes can exchange data via TCP/UPD with SPIDER	System Test	UC7_T22	Fullfilled (Tested)	
UC7_KPI_IP_Req_09	ROS2 stack installed on target platform	Integration Test	UC7_T2	Fullfilled (Tested)	
UC7_KPI_IP_Req_10	Library for diverse redundancy is built on target platform	Integration Test	UC7_T26	Fullfilled (Tested)	
UC7_KPI_IP_Req_11	LEDEL library is built for target platform	Integration Test	UC7_T19	Fullfilled (Tested)	
UC7_KPI_IP_Req_12	Resource monitoring library built for the target platform	Integration Test	UC7_T27	Fullfilled (Tested)	
UC7_KPI_IP_Req_13	Hardware accelerator for NN model of UC7 integrated to platform	Integration Test	UC7_T19	Not validated	No acceleration of model due to the architecuture of the NN

Table 14 - Justification of KPI Results from UC7 (Part 2)

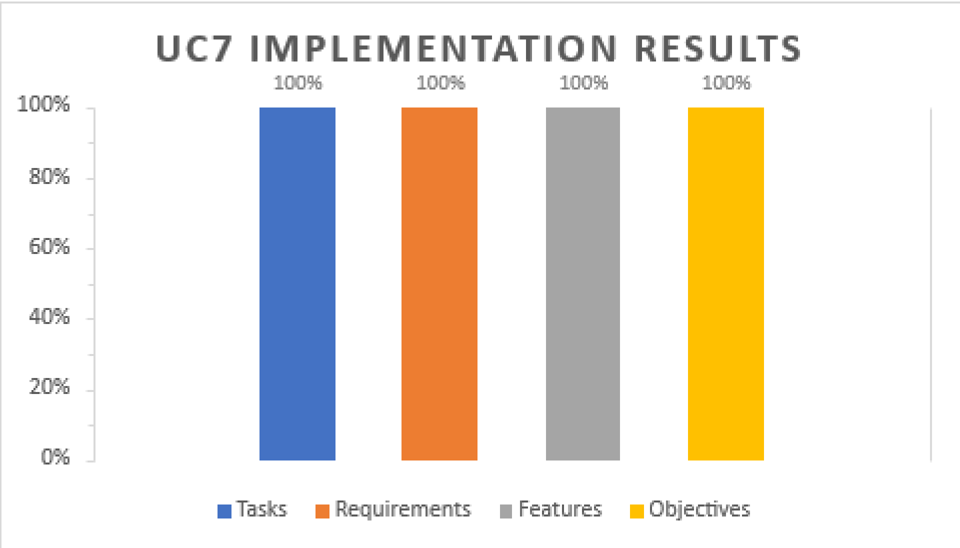


Figure 33 - Implementation Result of UC7

Figure 33 and Figure 34 summarize the implementation and validation status of UC7 and proof the objectives could be satisfied.

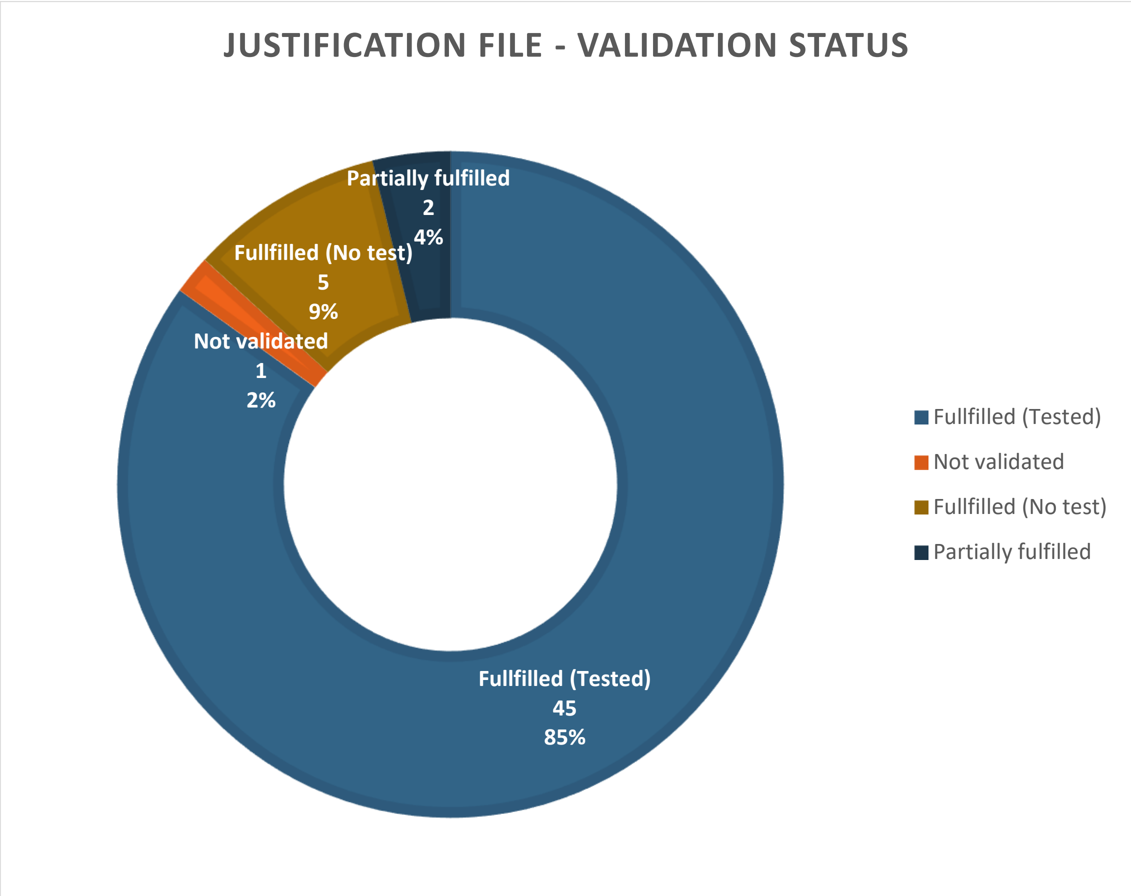


Figure 34 - Validation Status of UC7

6.2 Results of the executed benchmark

For comparison the computing platforms which are currently used with the SPIDER are selected as state-of-the-art system. For high level functionality the SPIDER used an industrial PC with an industry mainboard with Q87-chipset, and an Intel Core i7-5850EQ CPU and a Nvidia Quadro P1000 graphics card. Low-level hardware near applications were running on an Infineon 32-bit AURIX TriCore microcontroller.

The use-case functionality was implemented on a NOEL-V platform, providing up to 5 CPU cores and further cores for hardware acceleration, simulated on the Xilinx VCU118 FPGA.

BENCHMARK			UC7 NOEL-V	State-Of-Art System (Intel Core i7 + Infineon 32-bit AURIX TriCore)
UC7_KPI_FO_01	Runtime frequency of path tracking algorithm	>= 10 Hz	10 Hz	10 Hz
UC7_KPI_FO_02	Tests in simulation for redundant execution and monitoring succeed.	True/False	True	Not applicable
UC7_KPI_FO_03	Collision free path with path proximity	< 1m	< 0.5m	Not applicable
UC7_KPI_FO_04	Framework for platform independent development and verification of node functions available.	True/False	True	True
UC7_KPI_FT_01	Target platform supports ONNX.	True/False	True	True
UC7_KPI_FT_02	Path tracking function AI model executed at node level.	True/False	True	True
UC7_KPI_FT_03	Reinforcement learning approach trained model path proximity.	< 1m	< 0.5m	Not applicable
UC7_KPI_FT_04	Reinforcement learning approach trained collision avoidance	True/False	True	Not applicable
UC7_KPI_FT_05	LEDEL library availaible for target platform	True/False	True	False
UC7_KPI_FT_06	Sensor data from test drives can be stored on hard drive.	True/False	True	True
UC7_KPI_FT_07	Frame rate of collision avoidance function.	>= 10 Hz	10 Hz	10 Hz
UC7_KPI_FT_08	Switch to emergency state at time exceedance of AI function.	True/False	True	Not applicable
UC7_KPI_FT_09	Switch to emergency state at time exceedance of safety relevant function	True/False	True	Not applicable
UC7_KPI_FT_10	Safety relevant processes run redundant on different cores	True/False	True	False
UC7_KPI_FT_11	Switch to emergency state at fault detected by diverse redundancy model	True/False	True	Not applicable
UC7_KPI_FT_12	Switch to emergency state at fault detected in the communication messages	True/False	True	True
UC7_KPI_FT_13	Safety concept according IS 26262 availaible	True/False	True	True
UC7_KPI_FT_14	Target platform supports ONNX.	True/False	True	True
UC7_KPI_FT_15	Lidar sensor messages availaible at target platform at data rate.	20 Hz	20 Hz	20 Hz
UC7_KPI_FT_16	Path tracking node tested in target platform on proving ground	True/False	True	Not applicable
UC7_KPI_FT_17	Security assesment according ISO SAE 21434	True/False	True	True
UC7_KPI_FT_18	Max data transfer rate with ethernet, deviation of	1 Hz	True	True
UC7_KPI_FT_19	Target RISC-V hardware platform based on NOEL-V availaible.	True/False	True	Not applicable
UC7_KPI_FT_20	Linux operating system running on target platform	True/False	True	True (i7), False (AURIX)

Table 15 - Results of the Benchmark from UC7

Table 15 shows the results of the benchmark highlighting improvements in green color. All KPI for the use case features could be implemented on the target platform while this was not possible on the state-of-the-art system. The AURIX system lacks on performance and hardware acceleration, while the industrial PC provides no safety capability for execution of safety-critical tasks.

6.3 Evaluation of the results

6.3.1 Evaluation of Business KPIs

"KPI for Business Improvements" for the UC	Description	Assessment Method	Baseline	Target	Improvement	Achieved?
UC7_BKPI_1	Co-Execution of safety-relevant and machine learning based task	Validation of UC objectives	N.A.	N.A.	Use of a single computing platform	Yes
UC7_BKPI_2	Fail-operational capabilities	Validation of UC objectives	N.A.	N.A.	Demonstration purpose	Yes
UC7_BKPI_3	Energy Consumption	Estimation	160W+ 5W	50%	Reduction compared to industrial PC	Not in FPGA stage
UC7_BKPI_4	Maintenance Effort	Estimation	300h / for prototype	250	Reduction by simpler hardware arch	Not validated while project phase

Table 16 – "KPI for Business Improvement" for the UC7

Table 16 shows the "KPI for Business Improvement" of UC7. UC7_BKPI_1 and UC7_BKPI_2 is directly related to the objectives of the use case, UC7_BKPI_3 and UC7_BKPI_4 are indirect results from the implementation which are beneficial for the further development of the SPIDER prototype, or the SPIDER software used with other robot systems or vehicles.

UC7_BKPI_1
The SPIDER robot is consisted of a safety-critical part (e.g., collision avoidance, hardware control, …) and uncritical functions like a user-interface. The previous approach was to run the safety-critical part on an Infineon AURIX microcontroller, resulting in maintaining two different hardware and software architectures. The FRACTAL components allow to run safety critical functions on cores using software diverse redundancy and monitoring services, while executing uncritical parts and machine learning algorithms on the same platform.

UC7_BKPI_2
FRACTAL provides several components to enhance safety and security, required to implement fail-operational functions.

UC7_BKPI_3
The previous SPIDER computing platform is set by an AURIX microprocessor and an industrial PC with an external graphics card. The total energy consumption is about 160 W for the industrial PC and 5 W for the AURIX microcontroller at maximum. The

expected energy consumption after hardware synthesis of the NOEL-V is at maximum 30 W. Exact statements are not possible currently due to the lack of a hardware implementation.

UC7_BKPI_4

The software of the SPIDER is split to the AURIX and industrial PC on the SPIDER robot. However, the SPIDER software is also used in other robot systems and vehicles with higher requirements on the size of the computing platform. Individual adaptions for different platforms were necessary. Using the FRACTAL platform, the hardware can be synthesized on an optimized chip which is likely to fulfill the size requirements of all platforms. Further there is no separation between safety-critical and quality-of-service software parts necessary. The maintenance costs can only by estimated for the existing SPIDER robot, where we have about 300 h maintenance time per year. We expect to reduce this time to 250 h with the simper architecture, a number that scales up if the software is running on more robots and vehicles.

6.3.2 Discussion of the results

The performed tests for the KPI validate the objectives of the use case. With the help of the FRACTAL components, it was possible to run the safety-critical collision avoidance function on the same platform as the machine-learning algorithm used for path tracking. The FRACTAL components allowed to execute the safety-relevant tasks diverse and redundant to avoid common cause failures. Results of the fault injection prove the ability to detect such faults. To avoid blocking of the critical RISC-V cores, the monitoring capability of FRACTAL was used by sending interrupts at detection of timing or processing issues to do inference from other applications. On the same platform the neuronal network of the path tracking function could be executed using the LEDEL library. The possibilities to offload machine learning inference to dedicated AI accelerator cores opens allows the execution of even more demanding AI models, based on huge data inputs from cameras or 3D point-cloud based images.

HIGHLIGTHS	LOWLIGHTS
Co-Execution of safety-critical and non-critical functions enabled using software-only diverse redundancy mechanism and monitoring services. **Standardized ML-model inference and acceleration** due to the capabilities of the LEDEL library to import ONNX models and the possibility to offload tasks to AI acceleration cores.	The **hardware setup using FPGA** is a necessary step to evaluate the capabilities of the NOEL-V and FRACTAL system. But the development setup of the FPGA takes much more time for integration of the vehicle functions compared to existing hardware platforms like x86 or arm running Linux.
Multi-core system running ROS2 middleware allows a simple integration of large set of community libraries from the robotics scene and hardware vendors from the automotive domain. This reduces the amount of time required for changeover from x86/arm to RISC-V.	**Safety certification** according to automotive standards will be not possible using Linux on a multi-core system. However, the safety components developed in FRACTAL made one step forward in that direction.
The use case can be seen as **forerunner project for ROS2 on RISC-V.** The robotics community started during the FRACTAL project phase to show interest in the RISC-V architecture. Having a working demonstrator strengthens the position in research project proposals or service offers.	The chosen architecture of the ML-model fulfills the defined KPI, but still the tests discovered **oscillation in the movement of the robot**. This is due the missing knowledge of the previous path from the robot and is seen as future improvement of the algorithm.
Testing of the ONNX model led to a discovery of an **incompatibility between the ONNX formant given by TensorFlow and the LEDEL library**. SML could publish modifications to the latest version of the EDDL repository. The ONNX can be loaded and reaches same precision as with TensorFlow.	

Table 17 - Highlights and Lowlights of UC7

6.4 Consideration of safety and security

To take safety and security into account, relevant analysis from the automotive industry were prepared in WP4. The following subsections provide a summary of the method and results. State of the art for functional safety and security related to the SPIDER is described in deliverable D8.1.

6.4.1 Safety

Functional safety in the automotive sector is addressed by the ISO 26262 "Road vehicles – Functional safety". According to the ISO 26262 concept phase, the activities from Figure 35 were carried out for the use-case.

1. **Item Definition**
   The objective was to define the item "SPIDER", its dependencies and interaction with the environment and relevant other items.
2. **Hazard Analysis and Risk Assessment (HARA)**
   Malfunctions and potential hazards on vehicle levels were identified and combined in an assessment matrix to derive hazards events.
3. **Functional Safety Concept**
   Based on the HARA, safety goals were identified. From those goals functional safety requirements were derived. The functional safety requirements were appended to the list of function requirements of the collision avoidance function.

Figure 35 – Overview of safety activities in concept phase of ISO 26262

The functional safety concept can be seen as integral part of the SPIDER function development and is indispensable for the realization of commercial products, using part of the SPIDER software or hardware.

6.4.2 Security

The relevant standard for security considerations in the automotive industry, is the ISO/SAE 21434 "Road vehicles – Cybersecurity engineering". According to this standard a system model for the SPIDER was created and reconciled to the resulting threat model.

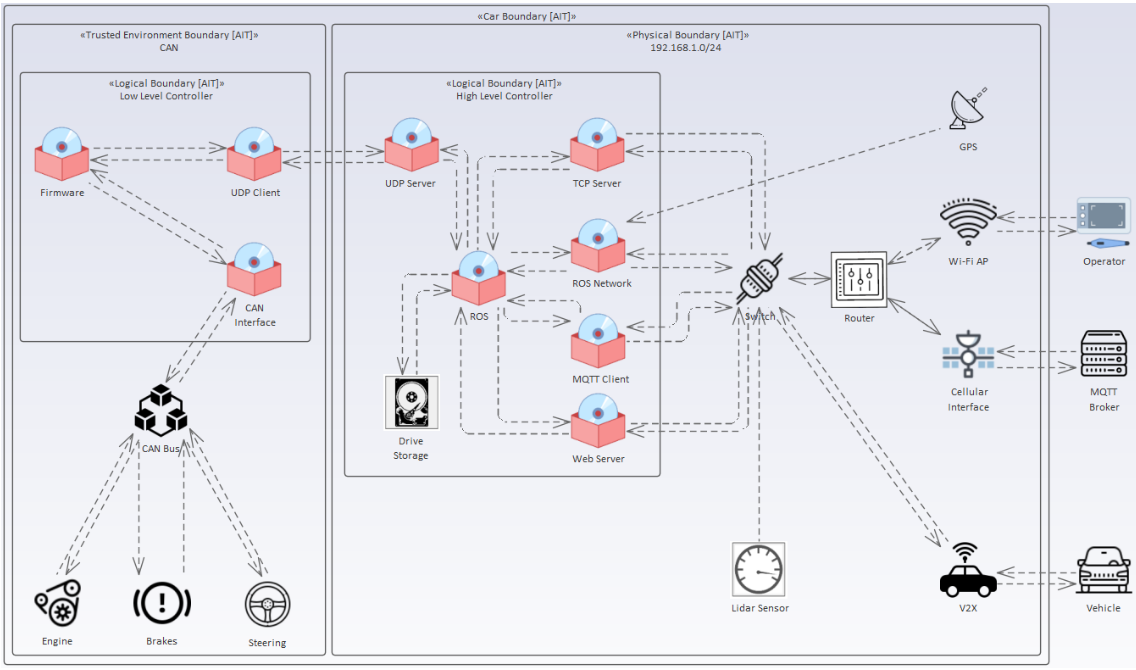


Figure 36 - SPIDER System Model

Based on the system model, the thread modelling aimed to create trees from so called anti-patterns. The tree was used to search for attack paths (or vectors). The final step was to create thread-rules. A thread-rule consists of title, description, threat type, feasibility, impact, rule. For example, the attack path of spoofing Lidar perception results in the rules of having lidar sensors redundant, and lidar application sources closed and sanitized.

The resulting thread-rules are mostly related to remote connection and physical access to hardware components. We found no thread-rules that specifically target the use-case functions collision avoidance and path tracking. Therefore, the decision was made to neglect security in the implementation of the use-case. However, if the functions are embedded to commercial products, the thread-rules need to be integral part of the product development.

6.5 Preparation for realization of commercial products

The developments in the FRACTAL project not only extended the functions of the SPIDER software, but also increased functional safety, despite minimizing the power consumption and form factor of the main computing platform. Nevertheless, further steps are necessary to obtain the platform for certification according to ISO 26262 in functional safety and ISO/SAE 21434 in security.

Currently, several projects are running with the SPIDER software which directly benefit from the results of FRACTAL.

- Virtual Vehicle Research closely works together with ALP.Lab GmbH, which commercially operates the test region for automated driving in Austria, where the SPIDER is rent for test purposes. The function upgrade of the path tracking software enables new testing capabilities in more complex scenarios which allows new types of offers to customers by ALP.Lab GmbH.

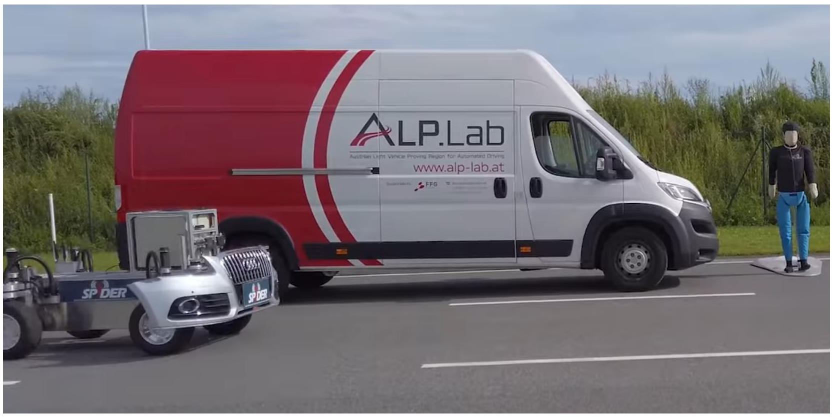


Figure 37 - SPIDER at tests with ALP.Lab

- In a cooperation with 4activeSystems, we are trying to integrate the SPIDER functionality for target vehicle imitation into 4activeSystems robotic platform. 4activeSystem is the global leading test solution provider for Autonomous Driving and ADAS, headquartered in Austria. The FRACTAL impact on the use-case with the capabilities to co-execute safety-critical and high-performance functions on the same computing platform, will simplify the integration of the software to a considerable extent.

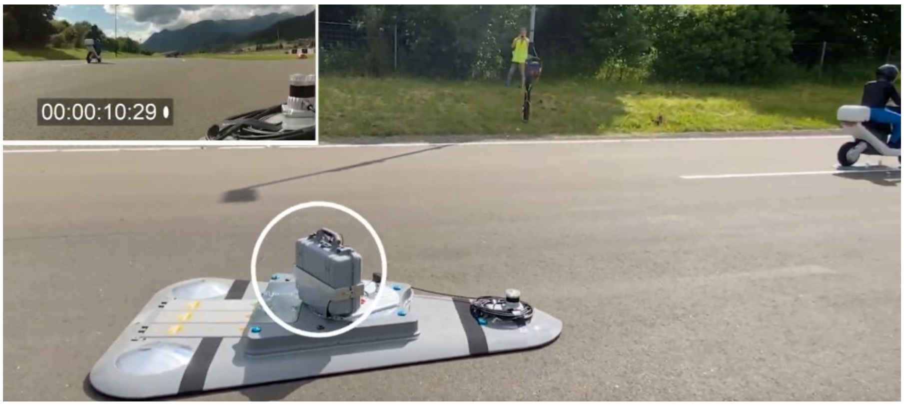


Figure 38 - 4activeSystem robotic platform with SPIDER software

- The SPIDER software will be also integrated to an electrical city bus within the Austrian national founded project TORUS. Aim of the innovation laboratory TORUS is to build an automated electrical city bus as test vehicle for new L3 automated driving functions. Later the city bus is rented to research projects, pilot projects, flagship projects and proof-of-concept evaluation activities. Gained experience from FRACTAL use-case development, new functions and increased safety of the SPIDER software will impact the TORUS project and later rental of the city bus.

Figure 39 - TORUS Autonomous Electric City Bus

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

The shuttle system presented for the FRACTAL project is a typical solution of an automated storage and retrieval system. As warehouses continue to adopt automation and autonomous systems, the role of autonomous shuttles has gained prominence. Key aspects of autonomous shuttles are speed, accuracy, reliability, and adaptability. Enhancing these factors not only results in smoother and more efficient

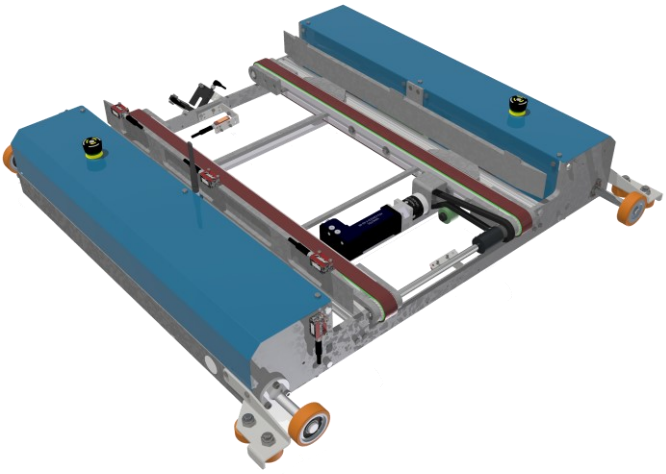


Figure 40 - FRACTAL shuttle base

operations within the warehouse but also contributes to overall productivity and cost-effectiveness. Improving the performance of such a system involves a multidimensional approach. It requires advancements in technology, including sensors, artificial intelligence, and machine learning algorithms, to enable better perception, decision-making, and navigation capabilities. Additionally, optimizing the software and algorithms that govern the shuttle's operation can lead to more efficient route planning, collision avoidance, and real-time adjustments to changing warehouse environments. In addition, consideration of the physical design and mechanics of shuttles can significantly impact their performance. Factors such as payload capacity, maneuverability, energy efficiency, and maintenance requirements play a critical role in maximizing the effectiveness and reliability of these vehicles. Customer requirements also influence the function and the appearance of the solution.

The objectives of VAL_UC8 are aligned with these points and have been defined as follows:

- Adaptivity:
  The shuttle system should adapt autonomously to new situations within the warehouse.
- Energy optimization and improved strategy for warehouse locations:

By optimizing the location of high-speed goods and their distribution; jams shall be avoided, and the efficiency of retrieving goods improved.

- Route optimization:
  Aggregated data on route patterns and delivery efficiency will be used by the AI application to achieve higher throughput for the warehouse.
- Increase pickup order productivity:
  Use of optimized strategies for system-driven picking based on the accumulated picking list.
- Defined bulk order fulfillment:
  Mass dispatch information, including the expected schedule is handed over to the swarm. The swarm resolves the solutions to be delivered as specified.

Some aspects of the state-of-the-art system had to be challenged in order to achieve these goals, such as job management or the computing capabilities in the shuttle itself. This influenced the FRACTAL component selection and put the focus mainly on WP4 and WP5 components and some components from WP3 and WP6.

In summary, the following functions were identified during the project:

1. Context awareness through person detection on the shuttle nodes, to gain more information from the environment and the ability to adapt to new situations.
2. Reliability on node level by implementing the adaptive time-triggered network on chip architecture for robust low-level communication.
3. AI powered job management/ orchestration.
4. Orchestration of applications as microservices for better software management and deployment in a swarm setup.

7.1 Results of the executed justification plan

7.1.1 Summary of results from justification plan

The summary of the results during WP8 are shown in Figure 41 for the implementation results and Figure 42 for the validation status. All test cases of the justification plan are added in Appendix A: Test Cases.

The implementation results reflect the approach, as well as their outcomes to achieve the objectives in terms of tasks, requirements, and selected FRACTAL features. The last value shows the direct link to the FRACTAL objectives applied to the use case. Two objectives could not be achieved during the research project and can be attributed to functional safety related parts. The fundamental prerequisite was a suitable development board, which the market did not provide. Even if from the software point of view, the solution could be implemented theoretically, proper testing and presentation of the concept without compatible hardware was not feasible, which meant that these points were not met. Additionally, some components weren't compatible with each other, and the network structure could not meet the requirements for safe communication between the nodes.

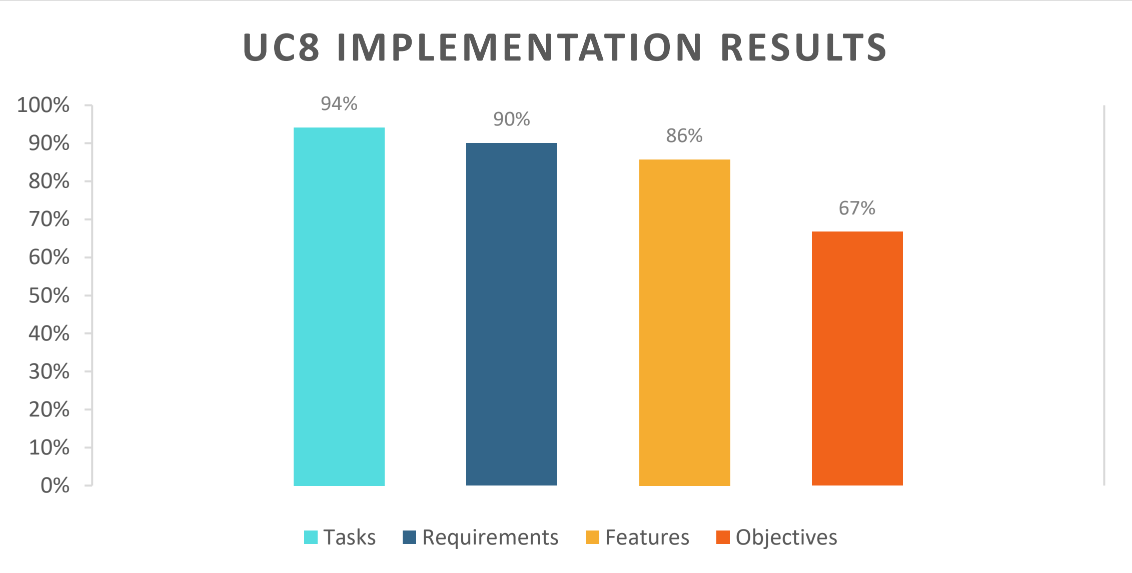


Figure 41 - UC8 Implementation results

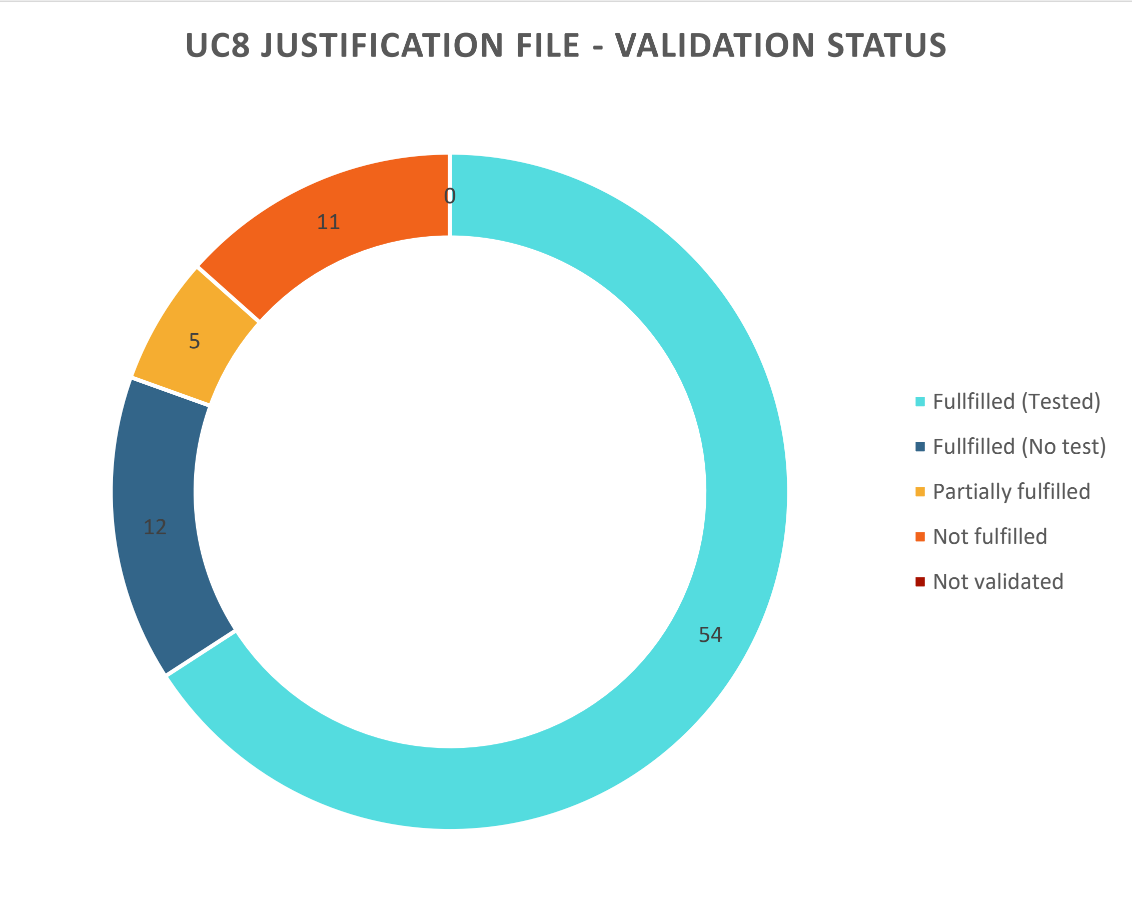


Figure 42 - UC8 Validation status

The results of the justification file reflect the status of the use case and is shown below from Table 18 to Table 21:

Justification of KPI Results (UC8)					
KPI ID	**Description**	**alidation Metho**	**Evidence**	**Validation Status**	**Validation Comments**
UC8_KPI_IP_01	Prepare hardware setup for Vitis AI on target (Versal node)	Integration Test	UC8_T1	Fullfilled (Tested)	HATMA integration done
UC8_KPI_IP_02	Build AA - shuttle orchestrator for target (Versal - ARM)	-	UC8_KPI_IP_03	Fullfilled (Tested)	
UC8_KPI_IP_03	Test AA - shuttle orchestrator for target (Versal - ARM)	Unit Test	UC8_T3	Fullfilled (Tested)	
UC8_KPI_IP_04	Build shuttle orchestrator application	Unit Test	UC8_T3	Fullfilled (Tested)	
UC8_KPI_IP_05	Prepare hardware setup for Vitis AI on target (Kria node)	Integration Test	UC8_T2	Fullfilled (Tested)	
UC8_KPI_IP_06	Build object detection model for target (Kria - ARM)	Unit Test	UC8_T4	Fullfilled (Tested)	Custom model postponed
UC8_KPI_IP_07	Test object detection model on target	Integration Test	UC8_T5	Fullfilled (Tested)	Integration successfully tested at 76 fps on KV260 with model from Xilinx model zoo
UC8_KPI_IP_08	Build zone evaluation logic application	Unit Test	UC8_T6	Fullfilled (Tested)	
UC8_KPI_IP_09	Setup cloud service orchestrator	Integration Test	UC8_T7	Partially fulfilled	Image repository connected to local cluster
UC8_KPI_IP_10	Build demonstration software for test setup	System Test	UC8_T8	Fullfilled (Tested)	
UC8_KPI_IP_11	Model training (Versal node) - Orchestrator	-	-	Fullfilled (No test)	Model don't need to be trained, as the data sets were generated by the orchestrator itself.
UC8_KPI_IP_12	Model training (Kria node) - Object detection	-	-	Fullfilled (Tested)	
UC8_KPI_IP_13	Integration of HW and SW base functionalities in the test setup	-	-	Fullfilled (No test)	Basic requirement for the project
UC8_KPI_IP_14	Test basic functionalities (shuttle control, lift control, interfaces)	Integration Test	UC8_T9	Fullfilled (Tested)	
UC8_KPI_IP_15	Test extended functionalities (FRACTAL components)	Integration Test	UC8_T10	Fullfilled (Tested)	
UC8_KPI_IP_16	Test cloud services	Integration Test	UC8_T11	Fullfilled (Tested)	
UC8_KPI_IP_17	Metrics Calculation	-	-	Fullfilled (No test)	Task for data collection and evaluation
UC8_KPI_FO_01	Cycle time of services on edge node with accelerated orchestrator implemented and running. (VERSAL)	System Test	UC8_T12	Fullfilled (Tested)	
UC8_KPI_FO_02	Cycle time of services on edge node with accurate cognitive AI application implemented and running. (KRIA)	System Test	UC8_T13	Fullfilled (Tested)	
UC8_KPI_FO_03	Self-sufficient decisions for each shuttle in respect to functional safety and additional degradation steps. High accuracy in detection is required.	System Test	UC8_T14	Partially fulfilled	Isolation and speed degration are "non safe" integrated
UC8_KPI_FO_04	Real-time inference for meta scheduler, which can react on various pre-defined events and make safe decisions for pathfinding and storage strategies for different goods.	System Test	UC8_T15	Fullfilled (Tested)	

Table 18 - Justification of KPI Results from UC8 (Part 1)

UC8_KPI_FO_05	Real-time inference for object detection on edge node with all services and accelerators implemented.	System Test	UC8_T16	Fullfilled (Tested)	
UC8_KPI_FO_06	Safe wireless communication between nodes.	System Test	UC8_T17	Not fulfilled	Wireless communication over black channel isn't feasible in current hardware setup
UC8_KPI_FT_01	Edge node has CAN Bus connectivity	Integration Test	UC8_T18	Fullfilled (Tested)	
UC8_KPI_FT_02	Edge node has AI/ ML accelerator	Integration Test	UC8_T19	Fullfilled (Tested)	DPU and AI Engine provided by Xilinx
UC8_KPI_FT_03	Edge node is capable of real time applications and process camera streams in real-time	Integration Test	UC8_T20	Fullfilled (Tested)	Currently the whole loop takes around 150 ms. 12 to 15 ms of them are just the ai inference of a tiny yolov3 model.
UC8_KPI_FT_04	The AI model are located in the node	-	-	Fullfilled (No test)	
UC8_KPI_FT_05	The AI models will be prepared for the VERSAL platform	-	-	Fullfilled (No test)	
UC8_KPI_FT_06	AI models will be trained in the cloud and then deployed on the node	System Test	UC8_T21	Not fulfilled	No model will be trained in the cloud during the fractal project. Preparation was done locally
UC8_KPI_FT_07	AI models will be trained on a device and then deployed on the node	Unit Test	UC8_T22	Fullfilled (Tested)	Orchestrator model generates data set by himself
UC8_KPI_FT_08	The AI models use supervised learning for training	Unit Test	UC8_T23	Fullfilled (Tested)	Model retrained for only "person" class.
UC8_KPI_FT_09	Vitis AI is able to import and execute YOLO model for KRIA platform	Integration Test	UC8_T24	Fullfilled (Tested)	with tiny yolov3
UC8_KPI_FT_10	Vitis is able to import and deploy convolutional neural networks for KRIA platform	Integration Test	UC8_T25	Fullfilled (Tested)	DPU was integrated with Vivado Flow
UC8_KPI_FT_11	Vitis is able to import and deploy artificial neural networks for Versal platform	Integration Test	UC8_T26	Not fulfilled	Not supported
UC8_KPI_FT_12	Vitis is able to import and deploy graph neural networks for Versal platform	Integration Test	UC8_T27	Not fulfilled	Not supported
UC8_KPI_FT_13	Edge node provides the library Tensorflow - Keras	Integration Test	UC8_T28	Fullfilled (Tested)	
UC8_KPI_FT_14	Edge node provides the library OpenCV	Integration Test	UC8_T29	Fullfilled (Tested)	
UC8_KPI_FT_15	Edge node provides the library NumPy	Integration Test	UC8_T30	Fullfilled (Tested)	
UC8_KPI_FT_16	Edge node provides the library PyTorch	Integration Test	UC8_T31	Fullfilled (Tested)	
UC8_KPI_FT_17	Service orchestration part of the fleet management system	Unit Test	UC8_T32	Fullfilled (Tested)	
UC8_KPI_FT_18	Edge node adapts to various predefined scenarios	Integration Test	UC8_T33	Fullfilled (Tested)	
UC8_KPI_FT_19	Edge node is fault tolerant	Integration Test	UC8_T34	Fullfilled (Tested)	
UC8_KPI_FT_20	Edge node adapts to required load level with different low power approaches	Integration Test	UC8_T35	Not fulfilled	RPU Power services integration failed in the kria node
UC8_KPI_FT_21	AI model for object detection have to be validated concerning the accuracy	Integration Test	UC8_T36	Fullfilled (Tested)	
UC8_KPI_FT_22	TT off chip comm. required for safe communication between the edge nodes	Integration Test	UC8_T37	Not fulfilled	not compatible with WP4T41-05 - Agreement protocol for Low-Power Services and network infracture
UC8_KPI_FT_23	TT on chip comm. required for safety monitoring the node level of an edge node	Integration Test	UC8_T38	Fullfilled (Tested)	
UC8_KPI_FT_24	Safety service is required for evaluation of the object detection	Integration Test	UC8_T39	Partially fulfilled	no hardware isolation done

Table 19 - Justification of KPI Results from UC8 (Part 2)

UC8_KPI_FT_25	Self testing for the TTNOC on the edge	Integration Test	UC8_T40	Fullfilled (Tested)	
UC8_KPI_FT_26	Scheduling services on node level to provide fail-safe operation	Integration Test	UC8_T41	Fullfilled (Tested)	
UC8_KPI_FT_27	Safe wireless communication between nodes	Integration Test	UC8_T17	Not fulfilled	not compatible with WP4T41-05 - Agreement protocol for Low-Power Services
UC8_KPI_FT_28	Safety service is required for evaluation of the object detection	Integration Test	UC8_T39	Fullfilled (Tested)	RPU AI access
UC8_KPI_FT_29	Scheduling services on node level to provide fail-safe operation	Integration Test	UC8_T33	Fullfilled (Tested)	
UC8_KPI_FT_30	Edge node must provide a degration level for processes	Integration Test	UC8_T45	Not fulfilled	not compatible with other components, only partially fullfilled by CAN-Bus demo
UC8_KPI_FT_31	Safety Regulation ISO 61508 Generic	Integration Test	UC8_T46	Not fulfilled	Implementation does not meet safety regulation
UC8_KPI_FT_32	Part of the meta scheduling approach	Integration Test	UC8_T33	Fullfilled (Tested)	
UC8_KPI_FT_33	Battery level of the shuttle will be tracked for data collection	Integration Test	UC8_T48	Fullfilled (Tested)	
UC8_KPI_FT_34	Shuttle edge node requires cameras for environmental awareness	Integration Test	UC8_T49	Fullfilled (Tested)	
UC8_KPI_FT_35	Shuttle edge node utilizes sensors for positioning in the racking	Integration Test	UC8_T50	Fullfilled (Tested)	
UC8_KPI_FT_36	Shuttle edge node utilizes sensors for fine positioning to the totes	Integration Test	UC8_T51	Fullfilled (Tested)	
UC8_KPI_FT_37	AI model for object detection via cameras for the shuttles	Integration Test	UC8_T52	Fullfilled (Tested)	
UC8_KPI_FT_38	AI model for object detection triggers on detection and generates an alarm	Integration Test	UC8_T53	Fullfilled (Tested)	
UC8_KPI_FT_39	Deployed design and models has to be verified during boot process	Integration Test	UC8_T54	Not fulfilled	Battery missing on carrier board to properly store BBRAM key
UC8_KPI_FT_40	Connection to higher-level processes, such as the mfc or for downloading diagnose data	Integration Test	UC8_T55	Fullfilled (Tested)	
UC8_KPI_FT_41	Connection between nodes, Versal <--> Kria	Integration Test	UC8_T56	Fullfilled (Tested)	
UC8_KPI_FT_42	Data protocol between nodes will be MQTT	Unit Test	UC8_T57	Fullfilled (Tested)	
UC8_KPI_FT_43	Fleet management system service orchestration	Integration Test	~~UC8_T58~~	Fullfilled (No test)	
UC8_KPI_FT_44	Fleet management system data orchestration	Integration Test	UC8_T59	Fullfilled (Tested)	
UC8_KPI_FT_45	Fleet management system model orchestration	Integration Test	UC8_T60	Fullfilled (Tested)	
UC8_KPI_FT_46	Hierarchical architecture on system level of the edge nodes	Integration Test	UC8_T61	Partially fulfilled	only on Versal
UC8_KPI_FT_47	Versal node will be implemented in the lift node	-	-	Fullfilled (No test)	Fullfilled by other
UC8_KPI_FT_48	Kria node (Zynq Ultrascale + MPSoC) will be implemented in the shuttle nodes	-	-	Fullfilled (No test)	Fullfilled by other
UC8_KPI_FT_49	Edge nodes execute a Linux OS	-	-	Fullfilled (No test)	Xilinx provides PetaLinux tools to generate suitable Linux OS
UC8_KPI_IP_Req_01	The edge node should have followed hardware specification: - at least 2 cores @ 800 MHz - at least 4 GB RAM - at least eMMC Memory or similar.	-	-	Fullfilled (No test)	Fullfilled by specs of the boards

Table 20 - Justification of KPI Results from UC8 (Part 3)

UC8_KPI_IP_Req_02	These communication protocols shall be used from Linux OS: - MQTT over WiFi mesh network for communication between nodes - CAN Bus for internal communication.	Unit Test	UC8_T18	Fullfilled (Tested)	
UC8_KPI_IP_Req_03	The edge node shall provide enough interfaces for two cameras.	Unit Test	UC8_T62	Fullfilled (No test)	Fullfilled by specification of the boards
UC8_KPI_IP_Req_04	The edge node shall be capable to detect objects (human body and other obstacles) from video input stream of the provided cameras and evaluate the detected object to generate a safe output, if the obstacle is in a defined range of the shuttle.	System Test	UC8_T63	Partially fulfilled	Safe output can't be generated, as the carrier board is not suitable for such application.
UC8_KPI_IP_Req_05	The edge node shall be able to use an adaptive orchestrator (scheduler) for storing strategies and optimized pathfinding for each shuttle depending on material (weight, type), frequency of requests, division of same type in different levels for alternative access/ faster access on big order amount.	System Test	UC8_T64	Fullfilled (Tested)	
UC8_KPI_IP_Req_06	The edge node shall offer optimized pathfinding: Improving path of the shuttles, for different scenarios; obstacle in same layer; malfunction of a shuttle; avoiding crossing in same level.	System Test	UC8_T65	Fullfilled (Tested)	
UC8_KPI_IP_Req_07	The node shall feature Linux operating system with real time capability (e.g. time-triggered communication capabilities).	Integration Test	UC8_T66	Fullfilled (Tested)	
UC8_KPI_IP_Req_08	Safety wireless communication should be over a black channel (ASIL 3, ISO 26262) between nodes.	System Test	UC8_T67	Not fulfilled	Wireless communication over black channel isn't feasible in current hardware setup
UC8_KPI_IP_Req_09	For the edge nodes a cross compiler shall be available to port control software.	-	-	Fullfilled (No test)	provided by Xilinx PetaLinux tools
UC8_KPI_IP_Req_10	The edge node shall support libraries, like Tensorflow/ Keras.	Integration Test	UC8_T68	Fullfilled (Tested)	WP3T34-03 Versal Model Deployment modification

Table 21 - Justification of KPI Results from UC8 (Part 4)

7.1.2 Implementation

In order to achieve our own and the FRACTAL objectives, a local test setup was created to provide a realistic testing environment. This setup allows controlled experiments, data collection, and algorithm optimization. Insights gained from this phase contribute to enhancing autonomous shuttle performance and informing best practices in warehouse automation.

The system architecture of the test setup consists of three nodes, which form a local cluster, like shown in Figure 43. In the main control cabinet is the Versal node placed, which controls both lifts (in the top and bottom of Figure 43). The shuttles have been equipped with the Kria boards, which can be found two times in the rack.

The hardware used for the UC8 test setup:

- 1x Xilinx Versal VCK190 (Versal Node) in the main cabinet
- 2x Xilinx Kria KV260 (Kria nodes) in the autonomous shuttles
- 2x Intel RealSense D435i were added to the shuttles to perform person detection on the node.
- 1x Network switch to provide network access to the Versal node

- 1x Wireless access point to provide network access to the Kria nodes

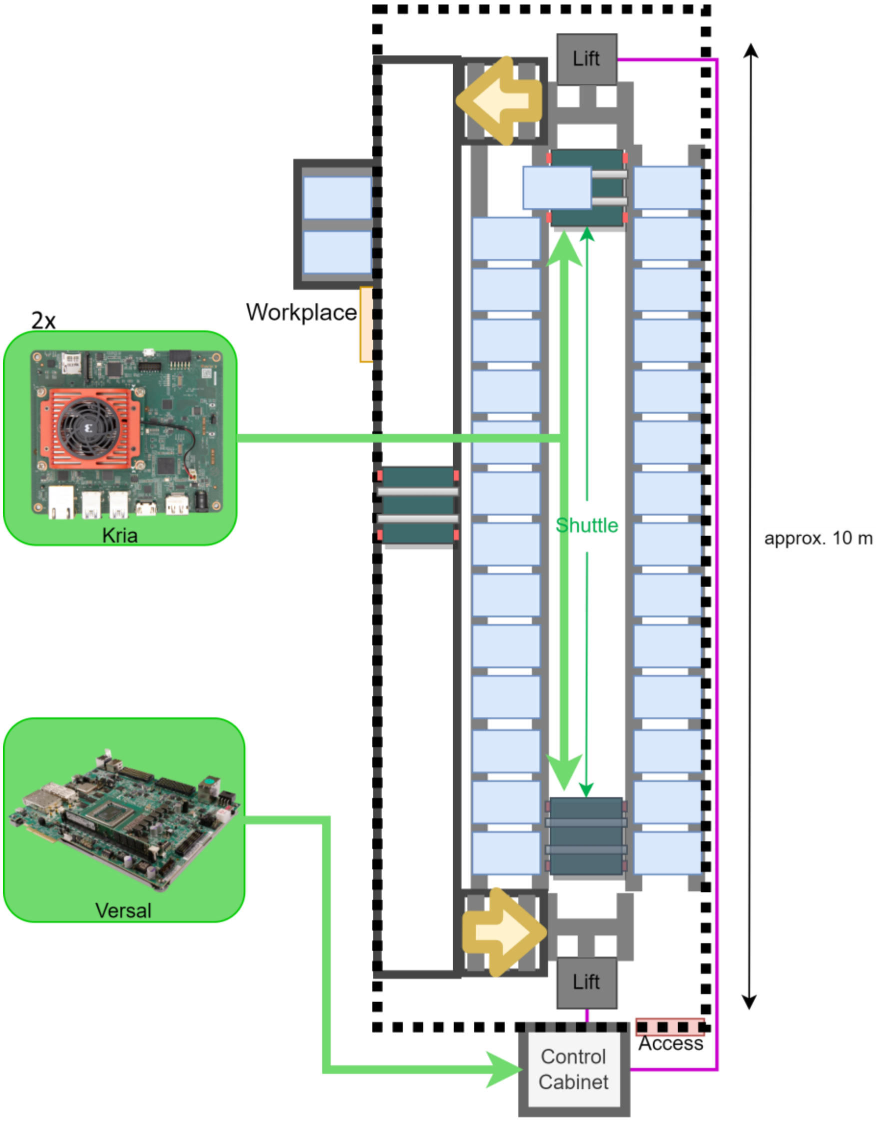


Figure 43 - UC8 test setup

For the wireless connectivity the component WP4T41-05 was added to the Kria nodes over ethernet. Regarding the hardware and software architecture of the Versal board the Xilinx tools were used as Vivado, Vitis, PetaLinux in Version 2021.2 and the AI part with Vitis-AI Version 2.0. For the Kria board the same tools, but in Version 2022.2 and for Vitis-AI Version 2.5. This separation was necessary due to some improvements in board compatibility.

The elaborated functions are localized as follows:

- The Versal node inherits robust low-level communication by the adaptive time-triggered network on chip in the combination of the HATMA adaptation logic, as well as the AI powered job orchestration.

- The Kria nodes inhere the person detection part, which is also accelerated by a Xilinx DPU.
- The local cluster will be managed by the Versal node, and the Kria nodes added as worker. In addition, the connection to the fractal cloud will be established.

Cloud	
WP5T52-04-05	Datasets version control
WP5T52-04-07	Images repository
WP5T52-05-02	Data pipelines and workflows orchestrator
WP5T52-06-01	Model preparation for Fractal Edge (Versal Xilinx Vitis AI)
WP5T54-01-01	MLBuffet
WP5T52-07-01	Kubernetes-based cloud platform container orchestrator

Table 22 - UC8 FRACTAL cloud components

Versal Node		Kria Node	
WP3T34-03	Versal Model deployment layer	WP4T41-04	Versal RPU access for Power Services
WP4T42-03	Scenario Generator	WP4T41-05	Agreement protocol for Low-Power Services
WP4T42-04	GA-Scheduler	WP4T42-02	Versal RPU access to AI acceleration
WP4T42-05	AI-Scheduler Model	WP4T43-08	OS Security Layer
WP4T42-06	Schedule Verifier	WP6T62-01	Data-Ingestion
WP4T42-07	Hierarchical Metascheduler	WP4T41-06	Versal Isolation Design - Functional Safety
WP4T43-04	ATTNoC	WP5T54-02-02	Kubernetes
WP6T62-01	Data-Ingestion		
WP5T54-02-02	Kubernetes		

Table 23 - UC8 FRACTAL edge components

To verify and validate the components, we tested them at the beginning individually and then moved to a consolidation of all components. In particular, the last step quickly showed whether the combination of several components was compatible or required further investigation, revision, and application. In Table 22 and Table 23 are the components behind the aspired functions listed.

7.2 Results of the executed benchmark

The benchmark tables (Table 24 and Table 25) make a comparison between the old solution and the FRACTAL platform.

BENCHMARK			UC FRACTAL SYSTEM	State-Of-Art System (armStone™A9r2 SBC)
UC8_KPI_FO_01	Cycle time of services on edge node with accelerated orchestrator implemented and running. (VERSAL)	< 20 ms	15 ms	20 ms
UC8_KPI_FO_02	Cycle time of services on edge node with accurate cognitive AI application implemented and running. (KRIA)	< 5 ms	2 - 3 ms	5 ms
UC8_KPI_FO_03	Self-sufficient decisions for each shuttle in respect to functional safety and additional degradation steps. High accuracy in detection is	> 95 %	FALSE	Not applicable
UC8_KPI_FO_04	Real-time inference for meta scheduler, which can react on various pre-defined events and make safe decisions for pathfinding and storage strategies for different goods.	< 2 s	2 s	Not applicable
UC8_KPI_FO_05	Real-time inference for object detection on edge node with all services and accelerators implemented.	100 ms	150 ms	Not applicable
UC8_KPI_FO_06	Safe wireless communication between nodes.	% telegram losses	FALSE	max. 5 telegram losses per s additional hardware required
UC8_KPI_FT_01	Edge node has CAN Bus connectivity	TRUE/ FALSE	TRUE	TRUE
UC8_KPI_FT_02	Edge node has AI/ ML accelerator	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_03	Edge node is capable of real time applications and process camera streams in real-time	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_04	The AI model are located in the node	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_05	The AI models will be prepared for the VERSAL platform	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_06	AI models will be trained in the cloud and then deployed on the node	TRUE/ FALSE	FALSE	Not applicable
UC8_KPI_FT_07	AI models will be trained on a device and then deployed on the node	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_08	The AI models use supervised learning for training	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_09	Vitis is able to import and execute YOLO algorithms for KRIA platform	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_10	Vitis is able to import and deploy convolutional neural networks for KRIA platform	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_11	Vitis is able to import and deploy artificial neural networks for Versal platform	TRUE/ FALSE	FALSE	Not applicable
UC8_KPI_FT_12	Vitis is able to import and deploy graph neural networks for Versal platform	TRUE/ FALSE	FALSE	Not applicable
UC8_KPI_FT_13	Edge node provides the library Tensorflow - Keras	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_14	Edge node provides the library OpenCV	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_15	Edge node provides the library NumPy	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_16	Edge node provides the library PyTorch	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_17	Service orchestration part of the fleet management system	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_18	Edge node adapts to various predefined scenarios	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_19	Edge node is fault tolerant	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_20	Edge node adapts to required load level with different low power approaches	TRUE/ FALSE	FALSE	FALSE
UC8_KPI_FT_21	AI model for object detection have to be validated concerning the accuracy	> 95 %	< 70 %	Not applicable
UC8_KPI_FT_22	TT off chip comm. required for safe communication between the edge nodes	TRUE/ FALSE	FALSE	Not applicable
UC8_KPI_FT_23	TT on chip comm. required for safety monitoring the node level of an edge node	TRUE/ FALSE	TRUE	Not applicable

Table 24 - Results of the Benchmark from UC8 (Part 1)

In many points the FRACTAL gives an improvement, even if it is sometimes beyond expectations. As reference to the old system, which is an industrial single board computer with a 32-bit ARM Cortex-A9 CPU. The old solution offered a good balance between performance for simple control services at the field level and power consumption, while we still benefit by applying an operating system (in our case Windows Embedded Compact 2013) instead of using bare metal solutions. In the FRACTAL platform we gain a lot more capabilities regarding edge computing for e.g., AI tools or orchestration of software components as microservices.

UC8_KPI_FT_24	Safety service is required for evaluation of the object detection	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_25	Self testing for the TTNOC on the edge	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_26	Scheduling services on node level to provide fail-safe operation	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_27	Safe wireless communication between nodes	TRUE/ FALSE	FALSE	TRUE
UC8_KPI_FT_28	Safety service is required for evaluation of the object detection	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_29	Scheduling services on node level to provide fail-safe operation	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_30	Edge node must provide a degration level for processes	TRUE/ FALSE	FALSE	Not applicable
UC8_KPI_FT_31	Safety Regulation ISO 61508 Generic	TRUE/ FALSE	FALSE	TRUE
UC8_KPI_FT_32	Part of the meta scheduling approach	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_33	Battery level of the shuttle will be tracked for data collection	TRUE/ FALSE	TRUE	TRUE
UC8_KPI_FT_34	Shuttle edge node requires cameras for environmental awareness	10 FPS	6.66 FPS	Not applicable
UC8_KPI_FT_35	Shuttle edge node utilizes sensors for positioning in the racking	TRUE/ FALSE	TRUE	TRUE
UC8_KPI_FT_36	Shuttle edge node utilizes sensors for fine positioning to the totes	TRUE/ FALSE	TRUE	TRUE
UC8_KPI_FT_37	AI model for object detection via cameras for the shuttles	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_38	AI model for object detection triggers on detection and generates an alarm	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_39	Deployed design and models has to be verified during boot process	TRUE/ FALSE	FALSE	FALSE (manual check)
UC8_KPI_FT_40	Connection to higher-level processes, such as the mfc or for downloading diagnose data	TRUE/ FALSE	TRUE	TRUE
UC8_KPI_FT_41	Connection between nodes, Versal <--> Kria	TRUE/ FALSE	TRUE	Not applicable
UC8_KPI_FT_42	Data protocol between nodes will be MQTT	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_43	Fleet management system service orchestration	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_44	Fleet management system data orchestration	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_45	Fleet management system model orchestration	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_46	Hierarchical architecture on system level of the edge nodes	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_47	Versal node will be implemented in the lift node	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_48	Kria node (Zynq Ultrascale + MPSoC) will be implemented in the shuttle nodes	TRUE/ FALSE	TRUE	FALSE
UC8_KPI_FT_49	Edge nodes execute a Linux OS	TRUE/ FALSE	TRUE	TRUE

Table 25 - Results of the Benchmark from UC8 (Part 2)

7.3 Evaluation of the results

7.3.1 Evaluation of Business KPIs

KPI for Business Improvements	Description	Assessment methodology	Baseline	Target	Improvement	Achieved?
UC8_BKPI_01	Throughput Incoming and outgoing containers in the system measured per hour.	In- and outgoing containers/ hour	147 containers/ h for the test setup	147 containers/ h	Consistent throughput nearly to the max. possible estimation of a simulation model, even with failures in the system.	No
UC8_BKPI_02	Availability System availability of equipment like the shuttles and lifts per SWARM.	%	95%	98%	High on demand requirement of customers to provide a system with nearly 100% availability.	Yes
UC8_BKPI_03	Reliability Mean time between failures will be measured and expected to gain after implementation of fractal components.	MTBF	1 Error(s) per Shuttle/ week	0.5 Error(s) per Shuttle/ week	Reduction of interrupts for fixing failures in the system.	Yes
UC8_BKPI_04	Average time between an update cycle How long does it take to perform an update cycle of a fleet, related to one shuttle.	Time measurement	60 s per shuttle	60 s per shuttle, but parallel operation as an update batch	Replace manual update process over USB/ SBC replacement.	Partially, single shuttle update time: approx. 2 min 10 s

Table 26 - "KPI for Business Improvement" for the UC8

The results of the business KPIs for UC8 are shown in Table 26 and are defined in four different categories, how the FRACTAL platform improves the value of such system.

UC8_BKPI_01

Throughput is presented as one of the most important KPIs in warehouse solutions and defines in- and outgoing containers per hour. In the test setup this value couldn't be reached, as the compensation of one failed shuttle reduces the performance to approximately 74 containers/h. In larger systems, the loss in value of throughput could be better compensated.

UC8_BKPI_02

The second business KPI shows the availability of the system. During the negotiation phase, this value is typically included in the contract and defined by means of testing. Adding troubleshooting routines and monitoring capabilities into the shuttles shows that a gain in the availability is feasible and would provide great value in such systems.

UC8_BKPI_03

The reliability can be measured in errors per shuttle/ week. By improving the monitoring capabilities and adding context awareness by cameras reduces hazardous situations, were humans or machines cause failures.

UC8_BKPI_04

By implementing cloud capabilities into the system all services were integrated as microservices. This solution provides better handling of software components and gives the opportunity to orchestrate the system from the cloud as fleet management system.

7.3.2 Discussion of the results.

In summary, the results of the project are valuable and provide a new perspective on how shuttle systems can be built and what to consider during the implementation. 87% of the defined KPIs were met or partially met, due to a discrepancy between expectations and implementation, which is negligible as further efforts could close this gap with the provided components and concepts for their application.

High- and lowlights are listed in Table 27. A positive thing to mention is the gain in flexibility by the new development boards, as a lot of hardware specific components could be reproduced in the hardware design of a FPGA but require a lot of expertise and effort to get into a usable state.

HIGHLIGTHS	LOWLIGHTS
AI accelerator and inference in the edge The capability to run AI models in the edge gives the opportunity for complex and smarter sub-systems like the shuttles.	**High integration effort** in FPGAs requires a lot of time and resources.
Functional safety concepts in a single platform were elaborated, which would reduce the hardware size in industrial applications and give the	**Limited connectivity and interfaces** must be considered during evaluation board selection. Additional effort in adding specific interfaces which may

ability to react better in complex situations.	requires additional hardware and integration effort.
Additional cloud connectivity and local cluster between nodes gives flexibility in software development and deployment practices.	**Functional safety certification** cannot be applied, as the TRL from hard- and software is not mature enough and wasn't expected in the definition phase.

Table 27 - Highlights and lowlights of UC8

7.4 Consideration of safety and security

7.4.1 Safety

For functional safety in the industrial sector, the individual application must always be considered. Even if the general requirements of IEC 61508 are the starting point during the system design, the application of specific requirements from the C-type standards has the priority in the definition of the safety functions for machine safety.

To carry out a certifiable solution, some points should be considered.

- AI models are still not accepted in safety applications, as there is no standard that can be applied.
- The isolation of hardware must be performed and was shown as a concept during the research project. This isolation requires additional certification besides the actual certification of the safety logic and possible may not be used as a general solution.
- Safe communication between nodes for connected safety logics requires special hardware to provide the safety communication layer (defined in IEC 61784-3-3), which has attributes like time-sensitive-networking capabilities or a black channel in Wi-Fi solutions, regardless of wired or wireless solution. Such technology is defined for both, but requires specific hardware, which cannot easily be implemented in general development boards.

7.4.2 Security

In state-of-the-art solutions security did not have to be given much attention, what changes, when the connectivity to the cloud must be considered for customers. Typically, the network of the warehouse is isolated and only VPN access must be provided from the customer. By opening specific ports to connect to the cloud platform and permanent network access, cyberattacks like man-in-the-middle attacks become increasingly probable.

Figure 44 shows the setup and the considerations. The connection to the cloud must be established over HTTPS with signed TLS certificates. Internally the usage of MQTT provides for the broker also TLS encrypted messages, which utilizes also certificates. Update processes must also be encrypted in the local network. As all services are containerized in the solution, the guidance of "WP4T44-08 TLS Implementation on containers" must be followed. From the other perspective, the possibility of gaining access to a single node locally must be prohibited. Serial debug ports shall be turned off, and also USB ports to prevent automatic execution of scripts or binaries.

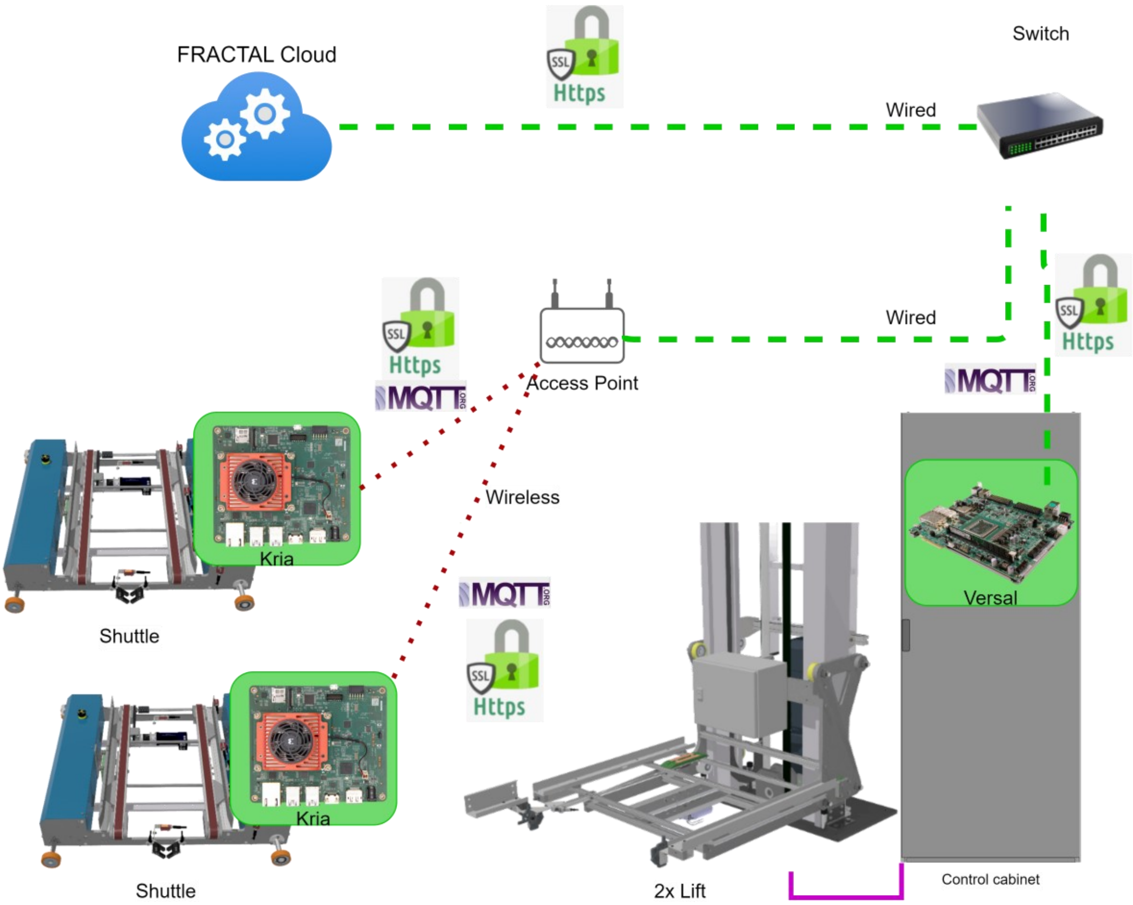


Figure 44 - Network structure UC8

7.5 Preparation for realization of commercial products

The realization of a commercial product requires still a lot of development effort in UC8. Regarding the preparation, one important step will be a custom hardware board, which inherits the beneficial outcomes from the FRACTAL project and the additional hardware requirements of the shuttle system. In parallel the exploitation of the AI models, the job orchestrator and the person detection, must achieve a mature level.

Another point would be to work out a 5G communication network in the shuttle system to verify range, quality and signal propagation time in a rack made of metal sheet, as there are a lot of unknown factors, which could prevent a real implementation.

The last point would be to elaborate the targeted functional safety design in the platform in cooperation with an official testing laboratory to achieve a certified board with functional safety capabilities, what also would set the first base for a legitimate CE declaration. In the best case, the standard DIN EN ISO 3691-4 for AGVs can be directly applied as well.

8 Conclusions

This document has presented the four use cases for the industrial validation of FRACTAL.

Each use case has listed the results of the executed justification plan and benchmark. The KPI were validated against defined test cases, attached in Appendix A: Test Cases. With this workflow the use cases could proof the use case objectives have been met. Furthermore, the integration of the required FRACTAL components into the use cases evinced that they can be applied to industrial tasks.

The implementation results have been discussed, listing highlights and lowlights, and considered by needs from safety and security. Finally, each use case has provided a perspective on what is needed for transition into a commercial product, including the necessary standards and regulations.

9 List of Figures

10 List of Tables

11 List of Abbreviations

2FA *2-Factor Authentication*
ADAS Advanced Driver Assistance Systems
AES *Sieh Advanced Encryption Standard*
API Application Programming Interface
APN *Access Point Name Access Point Name*
ATO *Automatic Train Operation*
CPU *Central Processing Unit*, *Central Processing Unit*
CRC *Cyclic Redundancy Check*
FPGA *Field Programmable Gate Array*
GPU *Graphics Processing Unit*
HARA Hazard Analysis and Risk Assessment
HIL *Hardware-in-the-Loop*
HTTPS *Hypertext Transfer Protocol Secure*
KPI *Key Performance Indicator*
LEDEL *Low-Power Energy Deep Learning Library*
MQTT *Message Queuing Telemetry Transport protocol*
ONNX *Open Neural Network Exchange*
PER perception
ROS *Robot Operating System*
SIL *Safety Integrity Level*
SIM *Subscriber Identity Module*
SoA *State of Art*
SPIDER *Smart PysIcal Demonstration and Evaluation Robot*
ssh *Secure Shell*
SW *Software*
TLS *Transport Layer Security*
TRL *Technology Readiness Level*
USB *Universal Serial Bus*
VPN *Virtual Private Network*

Appendix A: Test Cases

VAL_UC5

Test ID	UC5_T1
Test Name	Linux Generated Petalinux
Method	Integration Test
Objectives	Linux OS is ready for login at target platform
Prerequisites	Target HW platform: Xilinx Versal,
	Peta Linux generated imagen
Test steps	**Expected results**
Boot system	Linux OS on platform is ready for login

This test aims to verify base Versal Linux setup that allows installing/configuring Further services. Success is determined by correct access and login in Petalinux generated Linux image.

Test ID	UC5_T2
Test Name	OpenCV SGBM Test
Method	Unit Test
Objectives	Test OpenCV compilation for Versal node
Prerequisites	Target HW platform: Xilinx Versal,
	Peta Linux generated imagen
Test steps	**Expected results**
Build OpenCV native in versal	No errors, openCV dynamic libs
Build TestCode	No errors, binary
Launch test code with left/right sample images	Depthmap with coherent layout

UC5_T2 verifies the compatibility of OpenCV stereo vision library on the edge Versal platform. The test is based on building the library and executing stereo matching over a pair of images. The result is analyzed by inspection to determine if the calculus is correct.

Test ID	UC5_T3
Test Name	Cloud connection secured
Method	Unit Test
Objectives	Test connetivity to cloud in a secure way
Prerequisites	Target HW platform: Xilinx Versal,
	Peta Linux generated imagen
Test steps	**Expected results**
Test ssh/https connection to cloud	No errors, connection stablished to OVH cloud

This test aims to verify the connection between FRACTAL edge based on versal node with OVH FRACTAL cloud. Using ssh/https base libraries of VERSAL Linux image the connection is established in a secure way.

Test ID	UC5_T4
Test Name	Model Quantization Test
Method	Unit Test
Objectives	Test compatibility of CAF mode with VITIS quantization tool
Prerequisites	Target HW platform: Xilinx Versal,
	VITIS AI
Test steps	**Expected results**
Launch VITIS AI quantization	No errors, Generated models for selected DPU
Launch test inference over signal detection	Comparison between ground truth and model output matching

This test is oriented to check all the translation steps required to execute ONNX model trained by CAF in VERSAL DPU. The translation steps are:

- ONNX to h5 format from Keras
- Model quantization Using VITIS from h5 format to prepared files for selected DPU

Test ID	UC5_T5
Test Name	Versal inference speed test
Method	Unit Test
Objectives	Get throughput of Versal DPU with CAF model(yolo V3 608x608)
Prerequisites	Target HW platform: Xilinx Versal
	VITIS AI
	Generated petalinux
	Success in test UC5_T4
Test steps	**Expected results**
runtime using as input a recorded video	Framerate mesurements
Calculate average framerate/inference time	Average framerate for benchmark

This test gathers the results required for evaluating the performance within UC5 context. It allows to evaluate the performance KPIs which are the main ones for UC5.

Test ID	UC5_T6
Test Name	Inference accuracy test
Method	Unit Test
Objectives	Get model precission related to same evaluation in non-quantized X86
Prerequisites	Target HW platform: Xilinx Versal
	VITIS AI
	Generated petalinux
	Success in test UC5_T4
Test steps	**Expected results**
Execute inference over test image dataset on X86 and Versal	inference results
Calculate precission recall metrics over test dataset	PR Metrics

The test UC5_T6 allows UC5 to evaluate the accuracy variation during model translation to keep the suitability of the model for FRACTAL edge node.

Test ID	UC5_T7
Test Name	SW Release Deploy Test
Method	Unit Test
Objectives	Test the deploy of Docker images from cloud to edge
Prerequisites	Target HW platform: Xilinx Versal
	VITIS AI
	Generated petalinux
	Success in test UC5_T3
Test steps	**Expected results**
Execute automatic update service	Docker container pull from harbor in OVH cloud
Check version before and after	Version changed

This test is defined to test the correct behavior of application inside a Docker container and the ability of the node to pull the containers from OVH cloud.

VAL_UC6

Test ID	UC6_T1
Test Name	UC6_CMP_01
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.1 of D8.1	See section 5.7.3.1.1 of D8.1

Test ID	UC6_T2
Test Name	UC6_CMP_02
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.2 of D8.1	See section 5.7.3.1.2 of D8.1

Test ID	UC6_T3
Test Name	UC6_CMP_03
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.3 of D8.1	See section 5.7.3.1.3 of D8.1

Test ID	**UC6_T4**
Test Name	UC6_CMP_04
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.4 of D8.1	See section 5.7.3.1.4 of D8.1

Test ID	**UC6_T5**
Test Name	UC6_CMP_05
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.5 of D8.1	See section 5.7.3.1.5 of D8.1

Test ID	**UC6_T6**
Test Name	UC6_CMP_06
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.6 of D8.1	See section 5.7.3.1.6 of D8.1

Test ID	**UC6_T7**
Test Name	UC6_CMP_07
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.7 of D8.1	See section 5.7.3.1.7 of D8.1

Test ID	**UC6_T8**
Test Name	UC6_CMP_08
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.8 of D8.1	See section 5.7.3.1.8 of D8.1

Test ID	UC6_T9
Test Name	UC6_CMP_09
Method	Unit Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.9 of D8.1	See section 5.7.3.1.9 of D8.1

Test ID	UC6_T10
Test Name	UC6_TASK_10
Method	Integration Test
Objectives	Component validation
Prerequisites	Component running on target board
Test steps	**Expected results**
See section 5.7.3.1.10 of D8.1	See section 5.7.3.1.10 of D8.1

VAL_UC7

Test ID	UC7_T1
Test Name	Linux on NOEL-V is booting on FPGA
Method	Integration Test
Objectives	Linux OS is ready for login at target platform
Prerequisites	Target HW platform: Xilinx VCU118,
	NOEL-V based SELENE platform
Test steps	**Expected results**
Boot system using GRMON	Debian based Linux OS on platform is ready for login.

Test ID	UC7_T2
Test Name	ROS2 Example running on target platform
Method	Integration Test
Objectives	ROS2 example node is executed on target platform as excpected
Prerequisites	ROS2 example code:
	https://github.com/ros2/examples/tree/foxy/rclcpp/topics
Test steps	**Expected results**
Install example node at target platform colcon build --packages-select cpp_pubsub	Build process suceeds
Run talker node ros2 run cpp_pubsub talker	Info messages in terminal every 0.5 seconds [INFO] [minimal_publisher]: Publishing: "Hello World: 0" [INFO] [minimal_publisher]: Publishing: "Hello World: 1"
Run listener node ros2 run cpp_pubsub listener	Info messages in terminal starting with last message from talker: [INFO] [minimal_subscriber]: I heard: "Hello World: 10" [INFO] [minimal_subscriber]: I heard: "Hello World: 11"

Test ID	UC7_T3
Test Name	Max data transfer rate deviation of 10 Hz
Method	Unit Test
Objectives	Maximum deviation of 1Hz
Prerequisites	Data rate for input data of PTF and CAF is validated using ROS QoS
Test steps	**Expected results**
Run unit test of CAF/PTF	CAF/PTF is started and connecting to SPIDER system
	CAF/PTF subscribing required topics (costmap)
	Test runs for 30sec
	QoS functionality raises an error if data rate < 9 Hz or > 11 Hz

Test ID	UC7_T4
Test Name	Simulated robot is following trajectory and avoiding obstacles.
Method	Simulation
Objectives	Target reached approximately while avoiding collision with an obstacle placed along the path.
Prerequisites	Gazebo simulation
Test steps	**Expected results**
Configure fig8 path and costmap with a single obstacle (radius 2m)	
Run PTF on FPGA	
Run Gazebo sim on test PC	
Call PathTrackingCommand service	SPIDER moves in simulation until target is reached evading the obstacle

Test ID	UC7_T5
Test Name	Avg. Path Proximity in meter
Method	Unit Test
Objectives	Mean squared cte over one episode in a obstacle-free scenario is less than 0.5 m.
Prerequisites	Python simulation
Test steps	**Expected results**
Configure fig8 szenario without obstacles	
python main.py --evaluate --model model_name	Plot is generated below ../data/evaluation

Test ID	UC7_T6
Test Name	Collision free rate
Method	Unit Test
Objectives	Reach target without hitting an obstacle
Prerequisites	Python simulation
Test steps	**Expected results**
Configure path and costmap with obstacles	
python main.py --genOutputSample --model model_name	Output is generated below data/models/inout_files. Check for collision

Test ID	**UC7_T7**
Test Name	Waypoint reaching rate
Method	Unit Test
Objectives	Consider an obstacle-free scenario. Introduce m waypoints on the path and determine the number of waypoints reached approximately in successive manner by the robot. The PTF shall achieve that more than 95 % of the waypoints are reached in a single episode.
Prerequisites	
Test steps	**Expected results**
Run PTF in obstacle-free scenario, fix waypoints on the path and count the number of waypoints which have been approximately reached in successive manner by the robot.	Count reached waypoints, a waypoint is reached if the center of the robot is less then 0.5 m away of the point.

Test ID	**UC7_T8**
Test Name	ONNX model validation
Method	Integration Test
Objectives	Check if ONNX model is loading and generating valid output
Prerequisites	ONNX model with sample input/output
Test steps	**Expected results**
Load ONNX model with LEDEL in QEMU environment	ONNX is loaded without errors
Load input and execute model	Compare generated output from sample files

Test ID	**UC7_T9**
Test Name	Unit test coverage of PTF
Method	Unit Test
Objectives	Line Coverage > 75 %
Prerequisites	
Test steps	**Expected results**
Run GNU gcov tool for the path tracking node	Check line coverage from the test report

Test ID	**UC7_T10**
Test Name	Unit test coverage of CAF
Method	Unit Test
Objectives	Line Coverage > 75 %
Prerequisites	
Test steps	**Expected results**
Run GNU gcov tool for the collision avoidance node	Check line coverage from the test report

Test ID	UC7_T11
Test Name	Loop rate check of collision avoidance node
Method	Integration test
Objectives	Loop rate of >= 10 Hz
Prerequisites	
Test steps	**Expected results**
Run Gazebo simulation	
Subscribe to collision avoidance distance to stop topic	
Error on QOS of the subsriber showing frequency < 10 Hz	No errors

Test ID	UC7_T12
Test Name	Detect high CPU load with resource monitoring
Method	Integration test
Objectives	Resource monitoring detects high CPU load on critical CPU core
Prerequisites	Simulation environment
Test steps	**Expected results**
Run Gazebo simulation	
Start collision avoidance node on FPGA	
Start resource monitoring on FPGA	
Start dummy node to create workload on critical CPU core	Resource monitoring detects high load and triggers safe state

Test ID	UC7_T13
Test Name	Validation of diverse redundancy lib output
Method	Integration test
Objectives	Valid results of diverse redundancy lib
Prerequisites	Simulation environment
Test steps	**Expected results**
Run Gazebo simulation	
Start collision avoidance node on FPGA	
Compare outputs of collision detection from implementation with active diverse redundancy and expected output	Output compares to expected output for all test runs.

Test ID	UC7_T14
Test Name	Simulated sensor data integration
Method	Simulation
Objectives	Simulated sensor data available at target platform nodes
Prerequisites	Gazebo simulation
Test steps	**Expected results**
Run Gazebo simulation	Check ROS2 topics available at target platform
	costmap from simulated lidar sensor (>= 10Hz)
	robot position from simulated positioning system (>=50Hz)
	vehicle speed and acceleration from simulated odometry (>=50Hz)

Test ID	**UC7_T15**
Test Name	Sensor data integration
Method	System Test
Objectives	Hardware sensor data available at target platform nodes
Prerequisites	
Test steps	**Expected results**
Run rosbag recorded from vehicle	Check ROS2 topics available at target platform
	fused costmap from lidar sensors (>= 10Hz)
	robot position from positioning system using RTK and IMU (>=50Hz
	vehicle speed and acceleration from odometry (>=50Hz)

Test ID	**UC7_T16**
Test Name	ML Metrics analysis
Method	Analysis
Objectives	Metrics analysis calculated from ML model using Jupiter notebook
Prerequisites	Python simulation and Jupiter notebook
Test steps	**Expected results**
Start Jupiter notebook from Python sim	
Evaluate model using python main.py --evaluate --model model_name	
Create visualization	Visualization of metrics for a model like proximity, collision rate, time consumption, or loop rate

Test ID	**UC7_T17**
Test Name	Check performance gain from hardware accelerator
Method	Integration Test
Objectives	Increased performance to CPU processing, loop rate >= 10Hz
Prerequisites	Gazebo simulation
Test steps	**Expected results**
Start gazebo simulation	
A. Start path tracking function using HW accelerator and measure time consumption of ML model inference	
B. Start path tracking function not using HW accelerator and measure time	
Compare execution times	Compare measured times of A to B.
	t(A) < t(B)
	loop rate(A) >= 10Hz

Test ID	UC7_T18
Test Name	Jupiter Notebooks generates paths, costmap and models
Method	Integration Test
Objectives	Jupiter Notebooks to automate path and model generation
Prerequisites	Python simulation
Test steps	**Expected results**
Load cost_map_gerneration_notebook	Check if notebook is able to produce valid costmaps
Load path_generation	Check if notebook is able to produce valid paths
Load model_transformation	Check if notebook is transforming the model to a valid ONNX

Test ID	UC7_T19
Test Name	EDDL (LEDEL) builds on target platform (NOEL-V)
Method	Integration Test
Objectives	LEDEL can be used in ISAR-linux running on NOEL-V
Prerequisites	UC7_T1
Test steps	**Expected results**
Install according guidlines on https://github.com/deephealthproject/eddl/tree/ledel	Builds without errors
Link library to example application find_package(eddl REQUIRED) target_link_libraries(your_target PUBLIC EDDL::eddl)	Example builds without errors

Test ID	UC7_T20
Test Name	Path tracking execution on node level
Method	Integration Test
Objectives	Path tracking reaches goals when executed at target platform
Prerequisites	Gazebo simulation
Test steps	**Expected results**
Run Gazebo simulation	
Start path tracking node on target platform	
Re-run for different costmap and path samples	Simulated robot follows path, reaches goal and evades obstacles

Test ID	UC7_T21
Test Name	Recording of real world sensor data
Method	System Test
Objectives	Sensor data from robot can be recorded
Prerequisites	
Test steps	**Expected results**
Start SPIDER robot	
Start data recording from operato panel	Check ROS2 topics available in the recording
	Pointclouds from Lidar sensors (>= 10Hz)
	Vehicle speed and acceleration (<=50Hz)
	GNNS data from RTK of two antennas (>=5Hz)
	IMU data (>=50Hz)

Test ID	UC7_T22
Test Name	Check topic monitoring
Method	System Test
Objectives	Topic monitoring trigger safe stop on time exceedance
Prerequisites	
Test steps	**Expected results**
Start Gazebo simulation	
Start Safety Monitoring	
Start collision avoidance on FPGA	
Start path tracking on FPGA	
Interrupt processing of collision avoidance	Monitor detects time exceedance of safe stop topic and triggers safe state
Release safe stop	
Interrupt processing of path tracking	Motion control detects time exceedance of command velocity topic and sets target velocity to 0

Test ID	UC7_T23
Test Name	Safe stop on error detected by diverse redundancy
Method	System Test
Objectives	Collision avoidance triggers safe stop on error from SafeDR
Prerequisites	
Test steps	**Expected results**
Start Gazebo simulation	
Start collision avoidance on FPGA	
Inject fault when processing critical function	Diverse redundancy library detects difference in processed outputs and returns false flag. Collision avoidance activates safe stop signal.

Test ID	UC7_T24
Test Name	Path tracking vehicle test
Method	System test
Objectives	Vehicle is following path using the node from FPGA
Prerequisites	
Test steps	**Expected results**
Start SPIDER robot	
Load path via operator panel	
Start tracking of node running on FPGA	Robot follows defined path and evades obstacles.

Test ID	UC7_T25
Test Name	SPIDER stops for obstacles on the way
Method	System test
Objectives	Safe stop initiated for obstacles on the way
Prerequisites	Recorded rosbags with obstacles
Test steps	**Expected results**
Start rosbag	
Start path tracking node on FPGA	
Start tracking of a path going through obstacles in the recording	Collision avoidance initiates safe stop.

Test ID	UC7_T26
Test Name	Build check of Diverse Redundancy Lib
Method	Integration test
Objectives	SafeDR can be used on target library
Prerequisites	UC7_T1
Test steps	**Expected results**
Build library using QEMU https://gitlab.bsc.es/caos_hw/software-diverse-redundancy-library	No Errors

Test ID	UC7_T27
Test Name	Build check of SafeSU
Method	Integration test
Objectives	SafeSU can be used on target platform
Prerequisites	UC7_T1
Test steps	**Expected results**
Build kernel module using a cross compiler. See https://gitlab.bsc.es/caos_hw/hdl_ip/bsc_pmu/-/tree/develop/drivers/linux-driver/linux-kernel-module	No Errors
Check riscv@noelv:~/lkm$ ls -als /dev | grep safesu	0 crw-rw-rw- 1 root root 247, 0 Jul 13 17:44 safesu

VAL_UC8

Test ID	UC8_T1
Test Name	Hardware setup for Versal Node (VCK190)
Method	Integration Test
Objectives	Integration of hardware design with fractal components
Prerequisites	Linux OS (PetaLinux) prepared, Vivado Platform Export
Test steps	**Expected results**
Generate Bitstream	true
Export Platform	true
Boot Linux after hardware import	true

Test ID	UC8_T2
Test Name	Hardware setup for Kria Node (KV260)
Method	Integration Test
Objectives	Integration of hardware design with fractal components
Prerequisites	Linux OS (PetaLinux) prepared, Vivado Platform Export
Test steps	**Expected results**
Generate Bitstream	true
Export Platform	true
Boot Linux after hardware import	true

Test ID	UC8_T3
Test Name	Test shuttle orchestrator
Method	Unit Test
Objectives	Test shuttle orchestrator on Versal Node
Prerequisites	Linux OS (PetaLinux) prepared
Test steps	**Expected results**
Prepare model for Versal Node with Vitis AI	True
Export xmodel successfully	True
Deploy orchestrator in WP3T34-03 Versal Model Deployment container	True
Check syntax of generated jobs for MQTT Broker	True/ correct

Test ID	UC8_T4
Test Name	Build object detection model for target (Kria - ARM)
Method	Unit Test
Objectives	Build model over Vitis AI for KV260
Prerequisites	Vitis-AI and prepared AA model
Test steps	**Expected results**
Prepare model for Kria Node with Vitis AI	True
Export xmodel successfully	True

Test ID	UC8_T5
Test Name	Object detection model on target (Kria KV260)
Method	Integration Test
Objectives	Execution of the model on the target platform and fulfillment of the specified performance
Prerequisites	Linux OS (PetaLinux) prepared, DPU integration in hardware design
Test steps	**Expected results**
Boot Linux	true
Execute model with Intel Realsense 435i camera	true
Meassure inference	> 10 fps

Test ID	UC8_T6
Test Name	Build zone evaluation logic application (Kria KV260)
Method	Unit Test
Objectives	Develop zone evaluation application and integrate it into RPU of KV260
Prerequisites	Linux OS (PetaLinux) prepared, DPU integration in hardware design
Test steps	**Expected results**
Get results from person detection model with bounding boxes	true
Get results from depth stream of bounding boxes	true
Find shortest path from camera to pixel	true
Create definitions for range and reaction processes of the shuttle	true

Test ID	UC8_T7
Test Name	Setup cloud services
Method	Integration Test
Objectives	Setup cloud for fleet management capabilities. Deployment and management of control services/ AI models/ data sets. Versioning of data sets/ costumer specific data/ prepared containers.
Prerequisites	Access to fractal cloud services
Test steps	**Expected results**
Connect local cluster to cloud	true
Start pulling all required images by local cluster	true
Check generated images	true

Test ID	UC8_T8
Test Name	Build demonstration software for test setup
Method	System Test
Objectives	Final build of demonstrator (test setup) with all fractal components.
Prerequisites	System ready for demonstration
Test steps	**Expected results**
Start demo	true
check job generator for right syntax	true
check job procedure	true

Test ID	UC8_T9
Test Name	Test basic functionalities (shuttle control, lift control, interfaces)
Method	Integration Test
Objectives	Test the basic functions of shuttle, lift and the corresponding interfaces. Including the migration of control services from Windows Compact Embedded 2013 to an embedded Linux OS.
Prerequisites	Production and purchase parts supplied
Test steps	**Expected results**
Core functions of shuttle equipment is working properly	true
Core functions of lift equipment is working properly	true
Transition mechanisms via sensors	true

Test ID	UC8_T10
Test Name	Test extended functionalities (FRACTAL components)
Method	Integration Test
Objectives	Test fractal specific components
Prerequisites	Integration of fractal components succeeded
Test steps	**Expected results**
component-wise test	successful
Ramp up the system with all components integrated	successful
Test behaviour of the system during single failures	successful, when reaction as expected

Test ID	UC8_T11
Test Name	Test cloud services
Method	Integration Test
Objectives	Test processes and deployment of software components in local cluster
Prerequisites	UC8_T7
Test steps	**Expected results**
Successfull build of control services container	true
Successfull build of model container	true
Successfull deployment in local cluster as "update" of single components	true

Test ID	UC8_T12
Test Name	Cycle time of services on edge node with accelerated orchestrator implemented and running. (VERSAL)
Method	System Test
Objectives	Meassure cycle time of control services, after integration of all specified fractal components.
Prerequisites	Integration of fractal components successfull
Test steps	**Expected results**
Ramp up the system with all components integrated	successfull
Meassure cycle time	< 5 ms (best effort will be around 1 ms)

Test ID	UC8_T13
Test Name	Cycle time of services on edge node with accurate cognitive AI application implemented and running. (KRIA)
Method	System Test
Objectives	Meassure cycle time of control services, after integration of all specified fractal components.
Prerequisites	Integration of fractal components successfull
Test steps	**Expected results**
Ramp up the system with all components integrated	successfull
Meassure cycle time	< 5 ms (best effort will be around 1 ms)

Test ID	UC8_T14
Test Name	Functional safety integration test (KRIA)
Method	System Test
Objectives	Test of the evaluation logic in the isolated part
Prerequisites	UC8_T6
Test steps	**Expected results**
Ramp up system	True
Test speed degration	0.3 m/s
Test obstacle avoidance	True
Test rescheduling of tasks and block areas with obstacles	True

Test ID	UC8_T15
Test Name	Orchestrator integration test (VERSAL)
Method	System Test
Objectives	Test of the evaluation logic in the isolated part
Prerequisites	UC8_T3
Test steps	**Expected results**
Ramp up system	true
Send random batch of orders	true
Check order and syntax of job in the MQTT broker	true

Test ID	UC8_T16
Test Name	Performance test person detection model (KRIA)
Method	System Test
Objectives	Verifiy DPU performance on fully integrated node
Prerequisites	UC8_T13
Test steps	**Expected results**
Ramp up system	true
Collect data during normal operation	true
Check if inference + evaluation drops < 100 ms in various scenarios	true

Test ID	UC8_T17
Test Name	Safe wireless communication between nodes.
Method	System Test
Objectives	Safety wireless communication should be over a black channel (SIL 3) between nodes.
Prerequisites	TTNoC integrated in both bords
Test steps	**Expected results**
boot system on both boards	true
check clock sync	true
the system must work within a certain time frame peridioc	20 ms

Test ID	UC8_T18
Test Name	CAN Bus Connectivity
Method	Integration Test
Objectives	Send and receive CAN-Bus telegrams inside of a docker container
Prerequisites	Linux OS (PetaLinux), Vivado hardwaredesign
Test steps	**Expected results**
boot Linux	true
setup interfaces with virtual can tunnels into the docker container	true
send and receive test messages inside docker container	true

```
root@cf66e06ac05d:~# candump vxcan1
  vxcan1  1A3   [6]  04 01 00 00 23 80
  vxcan1  1A3   [6]  04 01 00 00 23 80
  vxcan1  1A3   [6]  04 01 00 00 23 80
  vxcan1  1A3   [6]  04 01 00 00 23 80
  vxcan1  1A3   [6]  04 01 00 00 23 80
  vxcan1  1A3   [6]  04 01 00 00 23 80
```

Test ID	UC8_T19
Test Name	Edge node has AI/ ML accelerator
Method	Integration Test
Objectives	Integrate DPU or similiar for edge AI inference
Prerequisites	
Test steps	**Expected results**
custom hardware design successfully generated	true
run AI model on DPU in custom hardware design	true

Test ID	UC8_T20
Test Name	Edge node is capable of real time applications and process camera streams in real-time
Method	Integration Test
Objectives	Real-time execution of binaries on OS for hardware control services in the field level
Prerequisites	UC8_T19
Test steps	**Expected results**
generate Linux OS with real-time capabilities	true
boot linux image	true
execute application in RTOS	true

Test ID	UC8_T21
Test Name	AI models will be trained in the cloud and then deployed on the node
Method	Unit Test
Objectives	Train models in the cloud and provide container image with xmodel.
Prerequisites	UC8_T19
Test steps	**Expected results**
prepare AI model and quantize model to xmodel	true
generate Docker image with AI model	true
deploy Docker image in image repo	true
execute Docker image on Versal node	true

Test ID	UC8_T22
Test Name	AI models will be trained on a device and then deployed on the node
Method	Unit Test
Objectives	Train models on local device and provide container image xmodel.
Prerequisites	UC8_T19
Test steps	**Expected results**
prepare AI model and quantize model to xmodel	true
generate Docker image with AI model	true
execute Docker image on Versal node	true

Test ID	UC8_T23
Test Name	The AI models use supervised learning for training
Method	Unit Test
Objectives	AI models are prepared with supervised training methods
Prerequisites	
Test steps	**Expected results**
AI mode training with data set	true
test trained model with verification data set	true

Test ID	UC8_T24
Test Name	Vitis is able to import and execute YOLO algorithms for KRIA platform
Method	Integration Test
Objectives	Import and prepare Yolo model in VItis AI for edge nodes
Prerequisites	
Test steps	**Expected results**
Quantize YOLO model in Vitis AI	true

Test ID	UC8_T25
Test Name	Vitis is able to import and deploy convolutional neural networks for KRIA platform
Method	Integration Test
Objectives	Import and prepare CNN for KRIA Board
Prerequisites	
Test steps	**Expected results**
Quantize CNN model in Vitis AI	true

Test ID	UC8_T26
Test Name	Vitis is able to import and deploy artificial neural networks for Versal platform
Method	Integration Test
Objectives	Import and prepare ANN for Versal
Prerequisites	
Test steps	**Expected results**
Quantize ANN model in Vitis AI	true

Test ID	UC8_T27
Test Name	Vitis is able to import and deploy graph neural networks for Versal platform
Method	Integration Test
Objectives	Import and prepare GNN for Versal
Prerequisites	
Test steps	**Expected results**
Quantize GNN model in Vitis AI	true

Test ID	UC8_T28
Test Name	Edge node provides the library Tensorflow - Keras
Method	Integration Test
Objectives	Edge node provides the library Tensorflow - Keras
Prerequisites	
Test steps	**Expected results**
Check VItis AI supported libraries for Keras	true

Test ID	UC8_T29
Test Name	Edge node provides the library OpenCV
Method	Integration Test
Objectives	Edge node provides the library OpenCV
Prerequisites	
Test steps	**Expected results**
Check PetaLinux tools for supported library OpenCV	true

Test ID	UC8_T30
Test Name	Edge node provides the library NumPy
Method	Integration Test
Objectives	Edge node provides the library NumPy
Prerequisites	
Test steps	**Expected results**
Check PetaLinux tools for supported library NumPy	true

Test ID	UC8_T31
Test Name	Edge node provides the library PyTorch
Method	Integration Test
Objectives	Edge node provides the library PyTorch
Prerequisites	
Test steps	**Expected results**
Check VItis AI supported libraries for PyTorch	true

Test ID	UC8_T32
Test Name	Service orchestration part of the fleet management system
Method	Unit Test
Objectives	Service orchestration in the fractal cloud with all cloud components for UC8
Prerequisites	
Test steps	**Expected results**
Test workflows	true

Test ID	UC8_T33
Test Name	Edge node adapts to various predefined scenarios
Method	Integration Test
Objectives	Part of the HATMA on the Versal Node, shall switch redundant CAN node on Versal.
Prerequisites	UC8_T1, UC8_T34
Test steps	**Expected results**
Test CAN switch node scenario	true

Test ID	UC8_T34
Test Name	Edge node is fault tolerant
Method	Integration Test
Objectives	Part of the HATMA on the Versal Node, shall trigger CAN node switch event.
Prerequisites	UC8_T1
Test steps	**Expected results**
Simulate CAN node fault on board	true
trigger CAN node adaption	true

Test ID	UC8_T35
Test Name	Edge node adapts to required load level with different low power approaches
Method	Integration Test
Objectives	Adaptation on power requirements
Prerequisites	UC8_T2
Test steps	**Expected results**
boot successfully KV260	true
check WP4T41-04-Rpu-Power-Service connectivity	true
try basic logic to influence power consumption	true

Test ID	UC8_T36
Test Name	AI model for object detection have to be validated concerning the accuracy
Method	Integration Test
Objectives	Test object detection model on DPU of KV260
Prerequisites	UC8_T2, UC8_T4
Test steps	**Expected results**
boot successfully KV260	true
start object detection container	true
collect data for mean average precision	> 70%

Test ID	UC8_T37
Test Name	TT off chip comm. required for safe communication between the edge nodes
Method	Integration Test
Objectives	Safe TT off chip communication between edge nodes, for safety services
Prerequisites	UC8_T17
Test steps	**Expected results**
pass when UC8_T17 fulfilled	true

Test ID	UC8_T38
Test Name	TT on chip comm. required for safety monitoring the node level of an edge node
Method	Integration Test
Objectives	Safe TT off chip communication between edge nodes, for safety services
Prerequisites	UC8_T17
Test steps	**Expected results**
pass when UC8_T17 fulfilled	true

Test ID	UC8_T39
Test Name	Safety service is required for evaluation of the object detection
Method	Integration Test
Objectives	Isolated evaluation process of the object detection
Prerequisites	UC8_T2
Test steps	**Expected results**
check isolated rpu part	true
check openAMP connection to APU side	true
check application project for RPU	true
test shuttle reaction on object detection in danger zone	pass, when stop is triggered

Test ID	UC8_T40
Test Name	Self testing for the TTNOC on the edge
Method	Integration Test
Objectives	Built-in self-testing of the ATTNoC
Prerequisites	UC8_T1
Test steps	**Expected results**
check self testing capabilities in implementation	true

Test ID	UC8_T41
Test Name	Scheduling services on node level to provide fail-safe operation
Method	Integration Test
Objectives	Part of the HATMA on the Versal Node, shall switch redundant CAN node on Versal.
Prerequisites	UC8_T34
Test steps	**Expected results**
pass when UC8_T34 fulfilled	true

Test ID	UC8_T42
Test Name	Safe wireless communication between nodes
Method	Integration Test
Objectives	Duplicate of UC8_T17

Test ID	UC8_T43
Test Name	Safety service is required for evaluation of the object detection
Method	Integration Test
Objectives	Duplicate of UC8_T39

Test ID	UC8_T44
Test Name	Scheduling services on node level to provide fail-safe operation
Method	Integration Test
Objectives	Duplicate of UC8_T33

Test ID	UC8_T45
Test Name	Edge node must provide a degration level for processes
Method	Integration Test
Objectives	Degration of system relevant processes on safety specific events
Prerequisites	UC8_T35
Test steps	**Expected results**

Test ID	UC8_T46
Test Name	Safety Regulation ISO 61508 Generic
Method	Integration Test
Objectives	Application of UC8 safety analysis
Prerequisites	
Test steps	**Expected results**
Check if system can meet the safety regulation with new safety approach.	true

Test ID	UC8_T47
Test Name	Part of the meta scheduling approach
Method	Integration Test
Objectives	Duplicate of UC8_T33

Test ID	UC8_T48
Test Name	Battery level of the shuttle will be tracked for data collection
Method	Integration Test
Objectives	Collect data about the battery level in the shuttle to generate statistics.
Prerequisites	Data ingestion integraton completed
Test steps	**Expected results**
boot Linux OS	true
connect to local cluster	true
pull and start Data ingestion & control services pod	true
check connectivity between local mqtt broker and local database	true
check data of control services between broker and database	pass, if same data with time stamp is located in database

Test ID	UC8_T49
Test Name	Shuttle edge node requires cameras for environmental awareness
Method	Integration Test
Objectives	Integration of cameras in shuttles
Prerequisites	Test setup ready
Test steps	**Expected results**
physical implementation of cameras on the shuttle performed?	true
start shuttle and check if librealsense is integrated in OS	true
check if cameras are listed and loaded with correct device drivers	true

Test ID	UC8_T50
Test Name	Shuttle edge node utilizes sensors for positioning in the racking
Method	Integration Test
Objectives	Reference positioning of the shuttle in a level.
Prerequisites	Test setup ready
Test steps	**Expected results**
boot Linux OS	true
connect to local cluster	true
pull and start Data ingestion & control services pod	true
check connectivity between local mqtt broker and local database	true
check data of control services between broker and database	pass, if same data with time stamp is located in database

Test ID	UC8_T51
Test Name	Shuttle edge node utilizes sensors for fine positioning to the totes
Method	Integration Test
Objectives	Fine positioning for the trays in the rack.
Prerequisites	Test setup ready
Test steps	**Expected results**
boot Linux OS	true
connect to local cluster	true
pull and start Data ingestion & control services pod	true
check connectivity between local mqtt broker and local database	true
check data of control services between broker and database	pass, if same data with time stamp is located in database

Test ID	UC8_T52
Test Name	AI model for object detection via cameras for the shuttles
Method	Integration Test
Objectives	AI model execution of camera model over DPU
Prerequisites	Test setup ready, UC8_T3
Test steps	**Expected results**
DPU hardware integration in PL part of FPGA	true
generate Linux OS with DPU integrated	true
verify access from OS to DPU	true

Test ID	UC8_T53
Test Name	AI model for object detection triggers on detection and generates an alarm
Method	Integration Test
Objectives	Integration between AI model to RPU distance estimation logic
Prerequisites	UC8_T52
Test steps	**Expected results**
verify RPU access from Linux OS	true
modify RPU power services	true
verify object detection binary access RPU power services	true

Test ID	UC8_T54
Test Name	Deployed design and models has to be verified during boot process
Method	Integration Test
Objectives	Secure boot for OS, to prevent modifications
Prerequisites	generated Linux OS
Test steps	**Expected results**
generate Linux OS successfully with PetaLinux tools	true
make Vitis platform	true
add secure key and generate boot files for KV260	true
verify successfull boot of KV260	true

Test ID	UC8_T55
Test Name	Connection to higher-level processes, such as the mfc or for downloading diagnose data
Method	Integration Test
Objectives	Access to local network for all necessary processes, like the job orchestration and cloud services
Prerequisites	UC8_T1
Test steps	**Expected results**
boot custom Linux	true
verify network address	true
ping fractal-project.eu	pass, when 0% packet loss

Test ID	UC8_T56
Test Name	Connection between nodes, Versal <--> Kria
Method	Integration Test
Objectives	Access local access between nodes
Prerequisites	UC8_T1
Test steps	**Expected results**
boot OS	true
verify network address	true
ping random node	pass, when 0% packet loss

Test ID	UC8_T57
Test Name	Data protocol between nodes will be MQTT
Method	Unit Test
Objectives	Utilization of a MQTT broker to provide job instruction for each node
Prerequisites	UC8_T1, UC8_T2
Test steps	**Expected results**
boot custom Linux	true
verify network address	true
subscribe to specific topic on broker	true
publish test telegram on topic	true

Test ID	UC8_T58
Test Name	Fleet management system service orchestration
Method	Integration Test
Objectives	Orchestration of cloud services

Test ID	UC8_T59
Test Name	Fleet management system data orchestration
Method	Integration Test
Objectives	Management of data sets, project specific configurations and persitent log files of each node.
Prerequisites	UC8_T58
Test steps	**Expected results**
verification of correct versioning and content of files in the cloud	true

Test ID	UC8_T60
Test Name	Fleet management system model orchestration
Method	Integration Test
Objectives	Management of pre-trained models in cloud.
Prerequisites	UC8_T58
Test steps	**Expected results**
verification of correct versioning and content of files in the cloud	true

Test ID	UC8_T61
Test Name	Hierarchical architecture on system level of the edge nodes
Method	Integration Test
Objectives	HAMA integration in Versal node
Prerequisites	Duplicate of UC8_T33
Test steps	**Expected results**
successfull device image generated	true

Test ID	UC8_T62
Test Name	The edge node shall provide enough interfaces for two cameras.
Method	Unit Test
Objectives	Check interfaces of target platform
Prerequisites	
Test steps	**Expected results**
simple check, if target platform provides two or more physical usb port	true

Test ID	UC8_T63
Test Name	The edge node shall be capable to detect objects (human body and other obstacles) from video input stream of the provided cameras and evaluate the detected object to generate a safe output, if the obstacle is in a defined range of the shuttle.
Method	System Test
Objectives	Verify logic of multiple components
Prerequisites	UC8_T2, UC8_T18,UC8_T5
Test steps	**Expected results**
	true

Test ID	UC8_T64
Test Name	The edge node shall be able to use an adaptive orchestrator (scheduler) for storing strategies and optimized pathfinding for each shuttle depending on material (weight, type), frequency of requests, division of same type in different levels for alternative access/ faster access on big order amount.
Method	System Test
Objectives	Verify logic of multiple components
Prerequisites	UC8_T1, UC8_T3
Test steps	**Expected results**
	true

Test ID	UC8_T65
Test Name	The edge node shall offer optimized pathfinding: Improving path of the shuttles, for different scenarios; obstacle in same layer; malfunction of a shuttle; avoiding crossing in same level.
Method	System Test
Objectives	Verify logic of multiple components
Prerequisites	UC8_T1, UC8_T3
Test steps	**Expected results**
	true

Test ID	UC8_T66
Test Name	The node shall feature Linux operating system with real time capability (e.g. time-triggered communication capabilities).
Method	Integration Test
Objectives	The OS shall be capable of real-time operations, in order to work properly.
Prerequisites	UC8_T2
Test steps	**Expected results**
build Petalinux OS with openAMP or RT-Patch	true

Test ID	UC8_T67
Test Name	Safety wireless communication should be over a black channel (ASIL 3, ISO 26262) between nodes.
Method	System Test

Test ID	UC8_T68
Test Name	The edge node shall support libraries, like Tensorflow/ Keras.
Method	Integration Test
Objectives	Redundant, fulfilled by prequisites
Prerequisites	UC8_T28-T31

Appendix B: FPGA fault injection to NOEL-V (VAL_UC7)

Summary

This appendix reports the results of fault injection experiments in the NOEL-V platform, carried out to characterize its robustness against hardware faults. First, the simulation-based fault injection (SFI) is used to evaluate the capability of staggered redundant execution (SRE) to protect the system from common cause faults. SRE is an error detection mechanism in multicore CPUs based on enforcing a predefined execution delay (time diversity) between replicated processes to reach different fault manifestation (detectable by voting) across replicas. In particular, the SFI experiments have analysed the effects of bit-flips in CPU registers on the execution of six different baremetal workloads, assuming different delays between head and trail CPU cores. SFI results have shown that increasing inter-core delay significantly improves the rate of error detection, reducing the probability of silent data corruption (SDC) by nearly an order of magnitude as the delay approaches to 200 clock cycles.

The FPGA fault injection (FFI) is used to evaluate the effects of permanent and transient faults on the execution of Linux applications protected by the Software Diverse Redundancy Library (SafeSoftDR). This library emulates the SRE mechanism at software level, managing process replication, staggering enforcement, monitoring and error detection. The target demo application (matrix multiplication kernel) runs two replicated processes (head and trail) and the required inter-process delay is monitored and enforced by the library. The processing results from the replicas are compared upon kernel completion, providing a safety status: safe (pass) when the results match, or unsafe (fail) in the case of mismatch. In addition, the application reports any timeouts (hangs) detected in the execution of the head/trail processes. The faultload applied in FFI experiments comprises single-bit upsets (bit-flips) in the content of LUTs (combinational logic) of the CPU cores. The experimental results show that SafeSoftDR library successfully detects timeouts (hangs) or replicas and incorrect results (data corruption) whenever these effects take place. At the same time, FFI also evinces that the rate of application failures is much lower than the rate of OS failures (not managed by SafeSoftDR library). In particular, across non-masked faults (less than 15% of tested faults) the most frequent fault effect is a crash (hang) of Linux OS. This effect becomes more pronounced with increasing fault duration. FFI results also show different sensitivity of the system to faults in different CPU cores. In particular, the faults in core-2 caused roughly 2x times lower rate of Linux crashes and 10x lower rate of segmentation errors than faults in core-1, which can be attributed to the scheduling of Linux kernel processes.

Simulation-based fault injection experiments

Staggered redundant execution (SRE) is an error protection mechanism in multicore CPUs that replicates execution of critical task on several CPU cores and enforces a predefined execution delay between replicas. The purpose of inter-process delay is to ensure that each replica remains in different execution state at any given time, making that any common fault simultaneously affecting several replicas, would manifest differently in each of them, and therefore would be detected by comparing the outputs (processing results). The efficiency of SRE mechanism in application to the NOELV platform is evaluated by means of simulation-based fault injection (SFI). The goal is to study the impact of increasing inter-process delay on the error coverage (detection rate) and on the rate of dangerous failures (i.e. silent data corruption).

Experimental setup

The targeted NOELV design under test executes six different baremetal workloads listed in Table B1. The result of each workload is a dataset comprising a linear array of integer/float items stored at a predefined memory address. The faultload comprises bit-flip faults uniformly sampled in time and space, i.e. each SFI run simulates one bit-flip at time T_i (randomly selected within the workload execution interval) in the register R_i (randomly selected across pipeline registers of the CPU core-0). Along with the fault at time T_i each SFI experiment performs 11 SFI runs targeting the same register R_i with different time offsets *S = [1, 2, 3, 4, 5, 10, 50, 100, 200, 500, 1000] clock cycles*, where S represents a staggering delay of the head process with respect to the trail (replicated) processes, as it is depicted in Fig.B1. The fault sample comprises 12000 faults per workload (1000 fault targets R_i x 12 injection timepoints T_i+S), which makes up a total of 72000 faults for the entire SFI campaign.

Table B1 – Baremetal workloads used in SFI experiment

Workload	Description	Duration	
Matmult	Integer matrix multiplication	20 000 ns	(2000 clock cycles)
Dijkstra	Finding shortest paths on the graph (Dijkstra algorithm)	60 000 ns	(6000 clock cycles)
AES	AES-256 encryption adapted from Tiny-AES	90 000 ns	(9000 clock cycles)
QSORT	Quick sort based on stdlib	70 000 ns	(7000 clock cycles)
BinarySearch	Binary search within an array of key-value structures	10 000 ns	(1000 clock cycles)
FIR	Finite impulse response filter adapted from Malardalen WCET	15 000 ns	(1500 clock cycles)

The fault effects are defined in terms of failure modes listed in Table B2. Each SFI run compares execution trace of the head process (injection at time T_i) with the trace of the trail process (injection at time T_i+S) as well as with the reference trace of the fault-free run. When the observed results of the head process match both the reference and trail results, then the fault is classified as masked. When the result of the head process is correct (matches the reference), but mismatches the trail, then the fault effect is said to be false alarm. On the opposite, if there is an agreement on results between the head and trail processes, but these results don't match the reference, the fault effect is registered as silent data corruption (SDC), which is the dangerous failure that SRE mechanism is supposed to prevent. Finally, if the head process produces incorrect result (in comparison to the reference trace), but at the same time it mismatches the trail results, then the fault effect is registered as signalled failure. Accordingly, the goal of SFI campaign is to study whether increasing staggering delay (S) between the head and trail processes improves SDC detection, i.e. reduces the rate of SDC in favour of signalled failures.

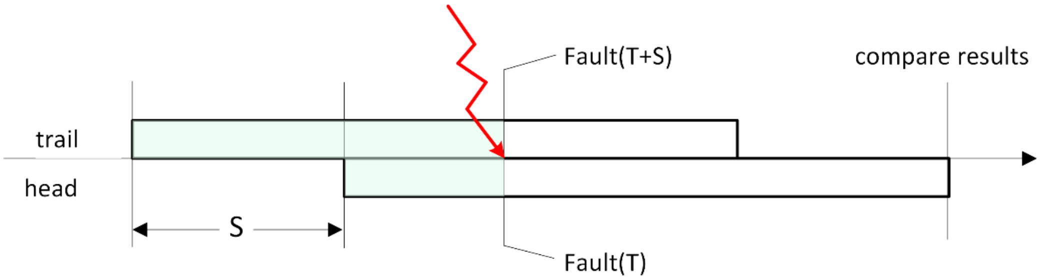


Fig. B1 – injection of common cause faults in the redundant processes running with a staggering delay S

Table B2 – Classification of fault effects in SFI experiment

Failure mode	Head == Reference	Head == Trail
Masked	✓	✓
False alarm (trail failure)	✓	X
Silent data corruption (unsafe)	X	✓
Signalled failure (safe)	X	X

The experiments have been executed by means of SFI tool from the DAVOS toolkit[2]. The NOELV design under study with the corresponding workloads is available in the public repository of the SELENE platform[3].

Experimental results

The results of SFI experiments for six different workloads are summarized in Fig.B2. As it can be seen from these diagrams, under short staggering delay (1-10 clock cycles) the silent data corruption accounts for roughly a half of total failures, i.e. ranges between 1.0% (out of 2% failures in case of binary search) and 3.5% (out of 7% failures in the case of matrix multiplication). When increasing the staggering delay to 200 clock cycles and above, the SDC rate is reduced by an order of magnitude on average, ranging between 0% (binary search, FIR) and 0.8% (AES-256).

[2] DAVOS toolkit available at: https://gitlab.com/selene-riscv-platform/DAVOS
[3] SELENE platform available at: https://gitlab.com/selene-riscv-platform/selene-hardware

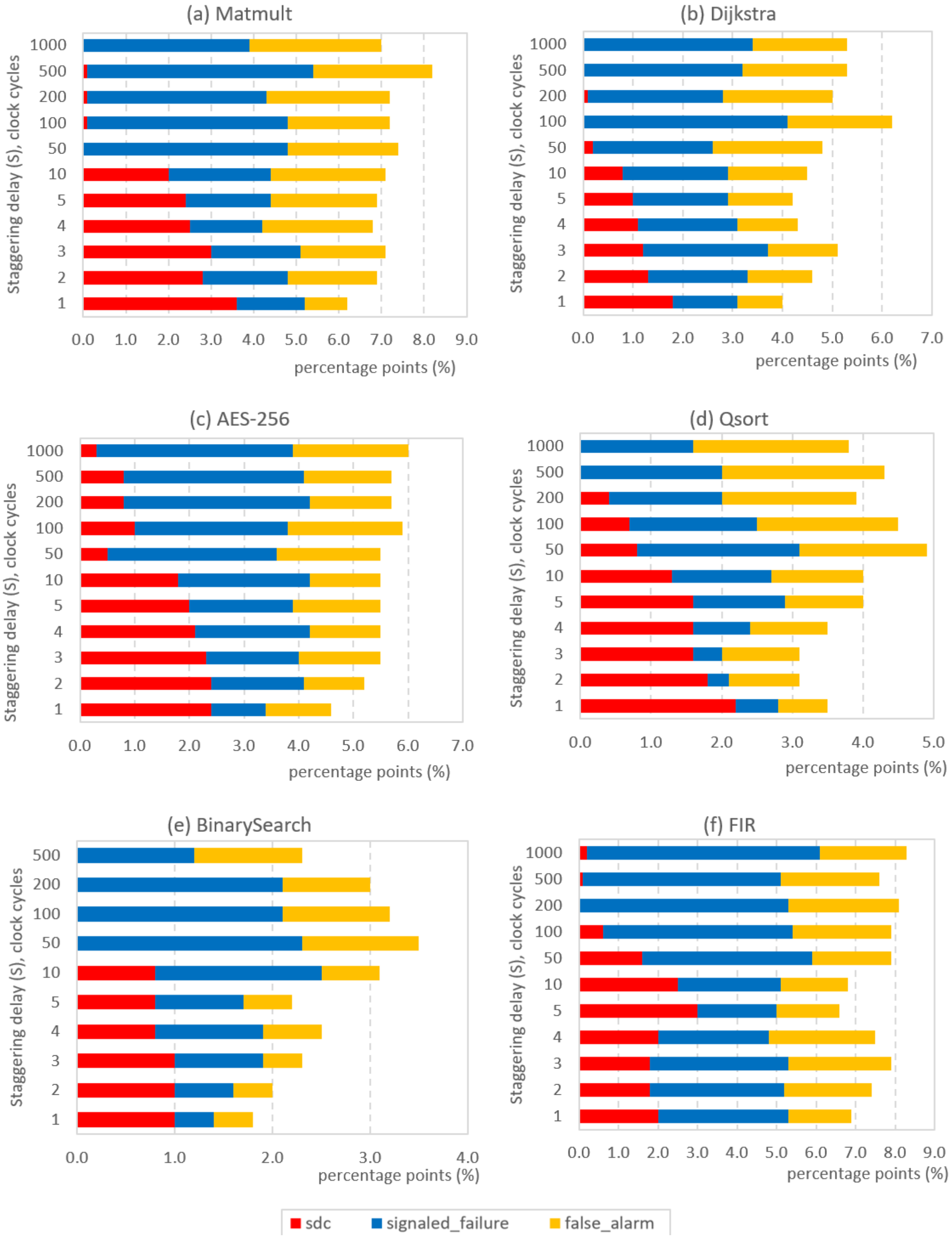


Fig. B2 – Estimated distribution of failure modes for different staggering delays

In three cases (matmult, Dijkstra, binary search) the SDC rate is reduced below 0.1% already starting from 50 clock cycles of staggering delay. This effect might be attributed to certain workload properties. For instance, all three aforementioned workloads calculate resulting items in such a way that once each item is stored in the memory, it remains unchanged after proceeding with calculation of next items; whereas Qsort and AES-256 keep modifying all bytes of resulting datasets until the workload completion. However, explaining whether and how such workload properties are related to SRE efficiency is out of the scope of this SFI experiment.

It is worth noting, that increasing staggering delay per se doesn't improve fault masking, as it would require majority voting across at least three replicas (TMR system), i.e. the sum of SDC and silent data corruption is not reduced, but most SDC events are converted to signalled failures. For the safety-critical system this means much safer behaviour in presence of common cause faults.

FPGA Fault injection experiments

The SafeSoftDR library supports SRE mechanism at the software level. It automates redundant execution of user-defined kernels in Linux OS, being in charge of replicating input datasets, spawning redundant worker processes (head and trail), periodically checking the execution progress of each replica in terms of the number of executed instructions, as well as pausing and resuming the execution of replicas whenever it is necessary to preserve the predefined staggering delay between them. Upon completion of worker processes, it compares their resulting datasets, notifying the user regarding the correctness of results.

It is worth noting that as a result of preliminary FPGA fault injection (FFI) experiments, several bugs have been fixed in the source code of the SafeSoftDR library, including those that prevented it from detecting errors in resulting datasets. In addition, in response to significant rate of application hangs discovered during preliminary FFI experiments, the library has been instrumented with additional execution control features allowing detection/handling of hags in the replicated processes (workers).

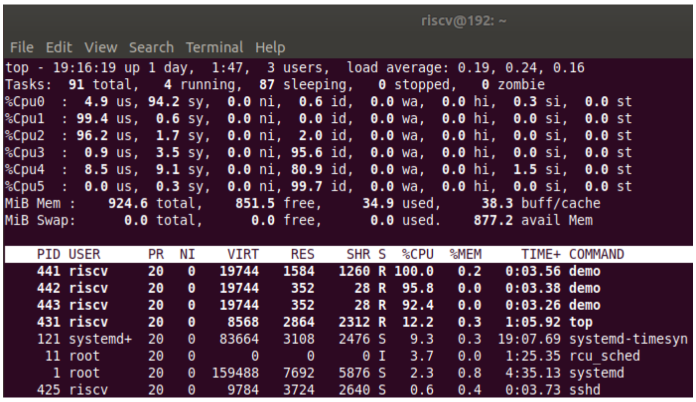
This section describes FFI experiments, carried out to evaluate the effects of HW faults on the current (stable) version of NOELV system protected at the application level by means of SafeSoftDR library. The fault injection experiments are carried out by means of the bit-accurate fault injection tool (BAFFI) from the DAVOS fault injection toolkit[2].

Experimental procedure

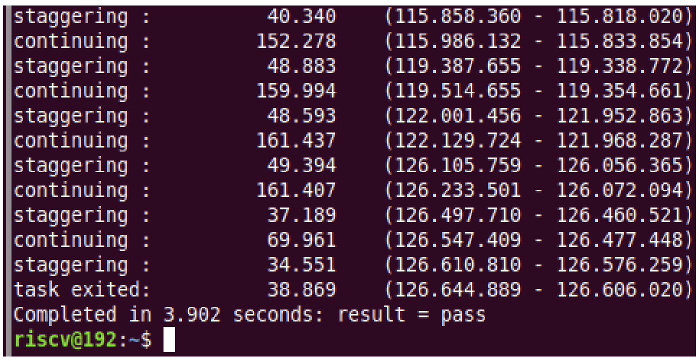
The system under study comprises six-core NOELV SoC, running a demo SafeSoftDR application on the top of ISAR Linux OS. The target application (workload) has the following properties**:**

- ✓ Executes matrix multiplication workload (150x150) protected by *protect_def_inp_out( ) function from the* SafeSoftDR library;
- ✓ Application runs three processes in parallel: two workers (head and trail replicas) and a monitoring process, as it is depicted in Fig. B3.
- ✓ Results from the replicas are compared at the end of execution, any mismatch is reported to the terminal as "result = fail", correct results are reported as "result = pass".

- ✓ An inactivity of more than 20ms (not related to the staggering) of any worker is reported to the terminal as timeout of head or trail process respectively.



(a) Performance monitoring of demo application: replicated processes (workers) executed on core[1] and core[2], monitoring process (main) executed on core[0]



(b) Sample output of demo execution

Fig. B3 – Profiling of CPU resources under fault-free run of the demo application

A faultload comprises single-bit upsets in LUTs sampled uniformly across all LUTs of CPU core-1 and core-2. According to the BAFFI report, targeted NOELV cores include: 48474 LUTs (core-1) and 48388 LUTs (core-2), utilizing roughly 3.1 million of configuration memory bits to be considered for fault sampling.

Three different fault durations are considered: transient (1 us, and 100 us) injected at the random time instant in the middle of workload execution, and permanent upset (injected before the workload start, active until workload completion). A total of 1000 faults is injected per each combination of injection scope (core-1 | core-2) and fault duration (1us | 100us | permanent), making up a total sample size of 6000 faults.

To automate evaluation of fault effects on the NOELV Linux environment, a special testbench has been implemented (in the form of python script) that, conforming with the BAFFI API, is in charge of interacting with the DUT (through the ssh interface in this case), invoking the workload (demo application), parsing its output responses and OS exception messages, determining the effects of injected faults, and resetting the DUT whenever necessary (rebooting Linux OS on the target, resetting the user session, etc.). The observed effects of injected faults on the application execution are described in terms of seven different failure modes, attending to the diagram depicted in Fig. B4.

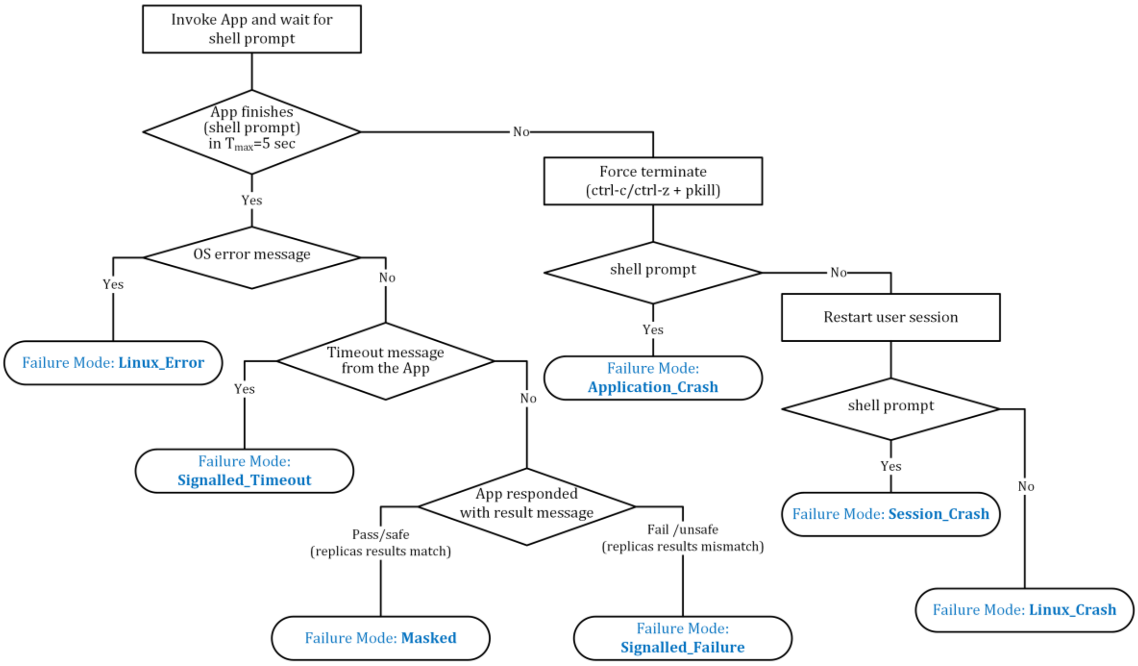


Fig. B4 – Classification of fault effects in the fault injection experiment, based on analysis of responses from the target application and exceptions from Linux OS

Experimental Results

The percentage (rate) of each failure mode observed during FFI experiment is summarized in Fig. B5. As it can be seen, most faults (85% - 95%) are masked (have no manifestation). Among the rest of faults, the most frequent fault effect is the crash of Linux OS, meaning that OS becomes irresponsive (OS reboot is required to recover the system). It is worth noting that the percentage of Linux crashes rises notably with increasing fault duration, ranging from just 1.3% in the case of very short faults (pulses) in core-2 up to 10.2% in the case of permanent upsets in core-1. It is also worth noting that the percentage of application crashes and session crashes is not strongly related to the fault duration. For instance, the percentage of application crashes changes by less than 0.2 percentage points (from 0.9% to 1.1%) when the upset duration in core-2 increases from 1us to permanent.

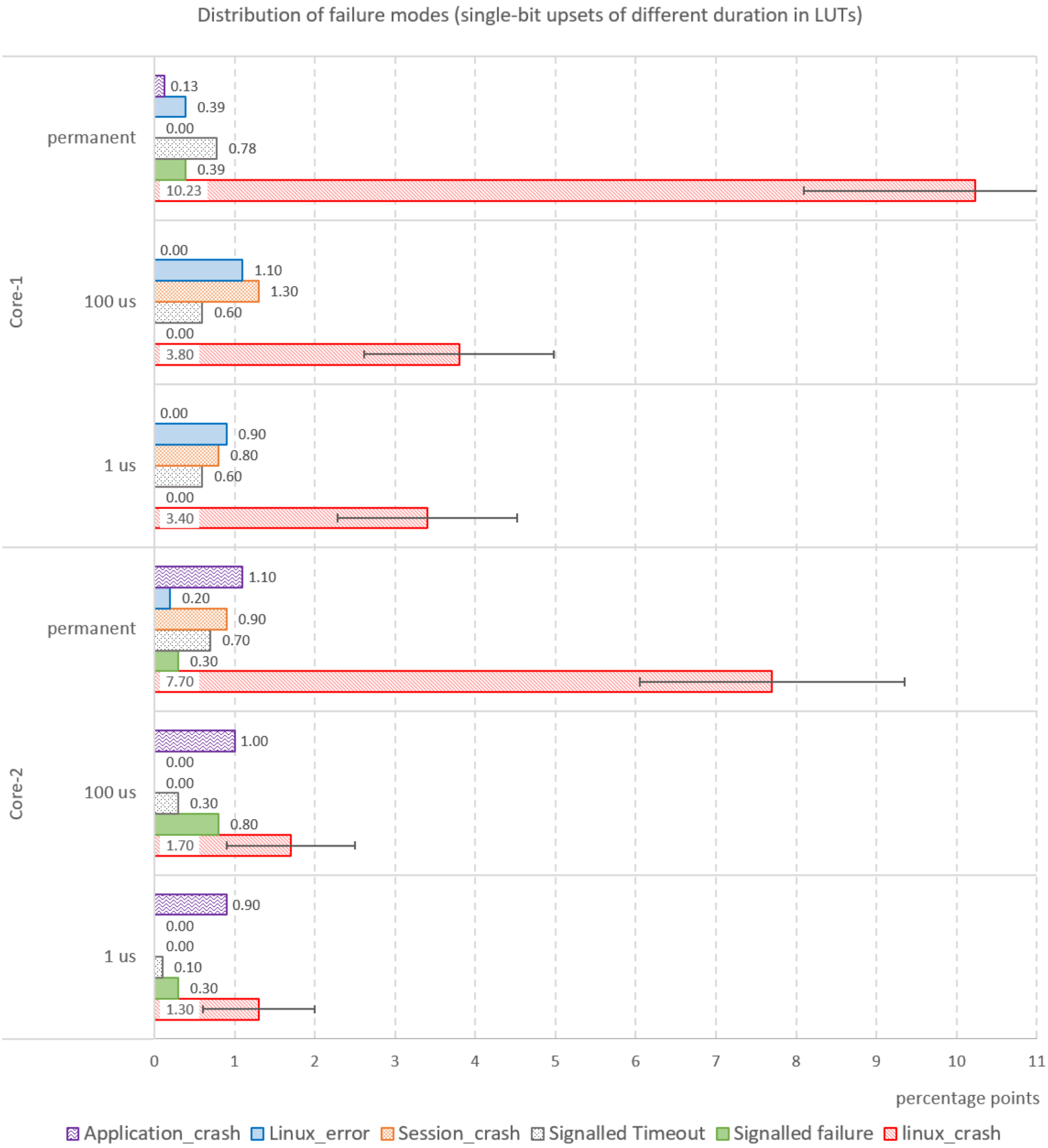


Fig. B5 - Estimated distribution of failure modes

At the same time, it can be seen that the distribution of crashes is related to the CPU core affected by the fault, i.e. faults injected in core-1 cause roughly 2x higher rate of Linux crashes and 10x higher percentage of segmentation errors than faults injected in core-2.

The percentage of application timeouts (signalled by SafeSoftDR library) ranges between 0.1% (1us upset in core-2) and 0.78% (permanent upset in core-1). The signalled failures (incorrect results from one of the replicas detected by SafeSoftDR library) appear more frequently in response to faults in the core-2 than in the core-1, and their percentage ranges between 0% and 0.8%.

Fig. B6 – Contribution of individual NOELV components into resulting failure rates, aggregated results (all fault durations) per core-1 (left) and core-2 (right)

Fig. B6 aggregates all FFI results (all three fault durations) for each NOELV core, and illustrates the contribution of each NOELV component into the resulting failure rates. In particular, when targeting core-1, the system was most sensitive to the faults in the integer pipeline (IU) and memory management unit (MMU). Each pipeline stage (except fetch) in the core-1 notably contributes to Linux crashes, Linux errors (exceptions) and signalled timeouts, being the decode, register access, and execute stages responsible for more than a half of total Linux exceptions. Whereas the signalled failures were only observed when injecting faults into the exception stage. It can be also seen that Linux crashes are much more evenly distributed between components in the core-1 than in the core-2. When targeting the core-2, there are three components that contribute the most into the total failure rates: the execute stage of integer pipeline, the instruction cache, and the MMU. Most application crashes are caused by the instruction cache, MMU and floating-point unit (FPU). Whereas the branch history table causes the highest number of signalled timeouts in core-2. Finally, there are nine components in the core-2 that cause signalled failures, being the highest contribution (more than a half of signalled failures) attributed to the different subcomponents of the MMU.

Conclusions

Fault injection (FI) experiments described in this document have shown that staggered redundant execution is capable of protecting the NOELV system against consequences of hardware faults, including those affecting simultaneously several replicated processes (common-cause faults). This requires a proper tuning of inter-process staggering delay. In particular, our simulation-based FI experiments have shown that the confident error detection (protection against silent data corruption) in the NOELV platform is achieved under staggering delays above 200 clock cycles.
By means of FPGA fault injection experiments it has been shown that the software-based SRE mechanism implemented in the SafeSoftDR library allows detection and signalling of data corruption errors and timeouts of replicas. At the same time, some additional protection mechanisms might be required to deal with hangs and crashes of Linux OS itself.