



*AI-augmented automation supporting modelling, coding,
testing, monitoring and continuous development in
Cyber-Physical Systems*

D5.7 “Use cases evaluation report – 1”

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, Spain. This document reflects only the author’s view and that the Commission is not responsible for any use that may be made of the information it contains.



Project Number:	101007350
Project Acronym:	AIDOaRt
Project Title:	AI-augmented automation supporting modelling, coding, testing, monitoring and continuous development in Cyber-Physical Systems
Deliverable Number:	D5.7
Deliverable Name:	Use cases evaluation report - 1
Deliverable Type:	R (Report)
Dissemination Level:	PU (Public)
Due Date:	M31 (October 2023)
Editor:	Lukas Marsik (CAMEA)
Contributors:	ABI, ACO, AST, AVL, BT, CAMEA, CSY, DT, HIB, INT, ITI, PRO, ROTTECH, TEK, UCAN, UNIVAQ, UOC, VCE, WMO
Reviewers:	WMO BT IMTA INT
Version	1.0
Date:	31 October 2023
Status:	Final
Abstract:	This document reports the intermediate results of evaluation of the AIDOaRt case studies after the first iteration of development (in terms of stage of development and usage of the tools) is done. The document will reflect the analysis of the progress towards the roadmap, requirements coverage and the improvements to be planned. As a conclusion, various numbers such as count of KPI or average KPI achievements are presented and discussed.

Executive summary

The AIDOaRt project aims at improving the entire design and life cycle of software systems, through a suite of tools that leverage Artificial Intelligence techniques. The *solutions provider* partners that participate in the project develop the tools, which are experimented by the *case studies providers* partners that use them to develop several software systems, which are also referred to as *demonstrators*. All partners collaborated in the work package WP1 “Use Cases Analysis and AIDOaRt Solution Architecture” to define the requirements of the tools and continue to collaborate in the *case studies*.

A case study consists of different types of activities: **(1)** to integrate the tools in the development processes of the case studies provider, **(2)** to develop the demonstrator, and **(3)** to evaluate the AIDOaRt outcomes by assessing the achievements of the tools as well as of the demonstrator.

The task T5.2 “Use Cases development” consists of the activities of types **(1)** and **(2)**, **T5.3** “Use Cases execution and evaluation” consists of those of type **(3)**. Both tasks belong to WP5 “Integration and Use Case Evaluation”. T5.2 and T5.3 are organised in two iterations whose results are the preliminary version and then the final version of the demonstrators and of the AIDOaRt evaluation.

This deliverable D5.7 (one of the results of task T5.3) follows the preliminary results of T5.2 reported in D5.6 [10], including the specified KPIs that were defined in T5.3. In this document, the first iteration by reporting on the preliminary evaluation will be concluded. The achievements of the second and final iteration will be reported in D5.8 [10] (task T5.2) and D5.9 [11] (task T5.3).

The case study providers proposed the systems to be developed on the basis of their specific industrial and technical interests. These systems span over different domains of application and require solutions for several challenges that are summarised below:

- *Requirements* — this challenge is mostly about automatic verification of requirement consistency with respect to a reference guideline. Automatic classification of the requirements and estimation of the development effort.
- *Models* — Automatic generation of models. One system, in the railway domain, uses the models to prove formally that the implementation complies with the specification. In the automotive domain, models of the driver behaviour are needed for the tests in real driving conditions (e.g. emissions or battery range and life-time). Moreover, in the same domain, the ambition is to arrive at Artificial Intelligence models that support the Project Manager by predicting the evolution of the development.
- *Verification* — Most case study providers share the ambition to automate test activities: selections of the critical parameters and of the test cases, generation and executions of the latter. The verification of the models and co-simulation techniques are included.
- *Self-adaptivity* — This challenge covers various topics such as sensor configuration, auto-calibration, and processing improvement according to the working conditions.

- *DevOps* — Artificial Intelligence techniques to support DevOps: identifications of links between code changes and test results; deployment adaptability to new projects; run time monitoring, to assess the correctness of new deployments and, during the operating life, for anomaly detection; automatic system recovery.

Finally, we would like to note that the above list is an attempt to summarise the key topics of the challenges tackled in the case studies of AIDOaRt, but it is not exhaustive.

Table of contents

Document revision log	3
Executive summary	4
Key Terminology Abbreviations	11
1. Introduction	12
1.1. Scope	12
1.2. Project overview	12
1.3. Case Studies	13
1.4. Project level and Case Studies KPIs.....	14
1.5. Document overview	16
2. ABI_CS01 case study “Safety critical systems in the automotive domain using disruption technology”	18
2.1. Case Study description	18
2.2. Use case scenario ABI_UCS1 — Functionality Verification	19
2.2.1. Summary of preliminary results.....	20
2.3. Use case scenario ABI_UCS2 — Test Definition.....	30
2.3.1. Summary of preliminary results.....	30
2.4. Use case scenario ABI_UCS3 — Compliance Verification	30
2.4.1. Summary of preliminary results.....	30
2.5. Implementation activities transversal to ABI use case scenarios	31
2.5.1. Summary of preliminary results.....	31
2.6. Evaluation of results considering requirements coverage	33
2.7. Evaluation of results considering KPIs	34
2.8. Planned improvements	35
2.9. Planned demonstration	36
3. AVL_CS02 case study “AI supported Digital Twin Synthesis supporting secure vehicle development and testing for novel propulsion systems”	38
3.1. Case Study description AVL_RDE - Real-Driver Emissions	38
3.1.1. Use case scenario AVL_RDE_UCS3 — Vehicle dynamics isolated speed profile data from real world measurements	40
3.1.2. Planned improvements	44
3.1.3. Planned demonstration	44
3.2. Case Study description AVL_TCV - Test Case Verification	45
3.2.1. Use case scenario AVL_TCV_UCS1 — SCENIUS Test Case Selection Validator.....	46
3.2.2. Use case scenario AVL_TCV_UCS2 — SCENIUS parameter recommender	50
3.2.3. Planned improvements.....	54

3.2.4.	Planned demonstration	54
3.3.	Case Study description AVL_SEC - Learning-Based Security Testing	54
3.3.1.	Use case scenario AVL_SEC_UCS1 — Learning-based model checking for security testing 55	
3.3.2.	Planned improvements	59
3.3.3.	Planned demonstration	59
3.4.	Case Study description AVL_ODP - Optimization of Development Processes	59
3.4.1.	Use case scenario AVL_ODP_UCS2 — ML-based KPI prediction	60
3.4.2.	Implementation activities transversal to AVL_ODP_UCS2 use case scenario	63
3.4.3.	Planned improvements	64
3.4.4.	Planned demonstration	64
4.	ALSTOM_CS03 (BT_CS03) case study “DevOps for Railway Propulsion System Design”	66
4.1.	Case Study description	66
4.2.	Use case scenario BT_UCS1	67
4.2.1.	Summary of preliminary results	67
4.2.2.	Evaluation of results considering requirements coverage	69
4.2.3.	Evaluation of results considering KPIs	69
4.3.	Use case scenario BT_UCS2 — Automated Model Parameterisation	70
4.3.1.	Summary of preliminary results: Solutions proposed by AVL	72
4.3.2.	Summary of preliminary results: Solution Proposed by MDU	73
4.3.3.	Evaluation of results considering requirements coverage	77
4.3.4.	Evaluation of results considering requirements coverage	78
4.4.	Planned improvements	80
4.5.	Planned demonstration	80
4.5.1.	BT_UCS_1	80
4.5.2.	BT_UCS_2	80
5.	CAM_CS04 case study “AI for Traffic Monitoring Systems”	81
5.1.	Case Study description	81
5.2.	Use case scenario CAMEA_UCS3 — Enable Low-power Device Configuration	82
5.2.1.	Summary of preliminary results	83
5.2.2.	Evaluation of results considering requirements coverage	84
5.2.3.	Evaluation of results considering KPIs	84
5.3.	Planned improvements	85
5.4.	Planned demonstration	85
6.	CSY_CS05 case study “Machine learning in interactive proving”	87
6.1.	Case Study description	87
6.2.	Use case scenario CSY_UCS1 — PO Classification	89
6.2.1.	Summary of preliminary results	89
6.2.2.	Evaluation of results considering KPIs	92

6.3.	Planned improvements	94
6.4.	Planned demonstration	94
7.	HIB_CS06 case study “AI DevOps in the restaurants business”	95
7.1.	Case Study description	95
7.2.	Use case scenario HIB_UCS1 — Automated log analysis.....	97
7.2.1.	Summary of preliminary results.....	98
7.2.2.	Evaluation of results considering requirements coverage	101
7.2.3.	Evaluation of results considering KPIs	101
7.3.	Use case scenario HIB_UCS2 — AI Requirements Management.....	101
7.3.1.	Summary of preliminary results.....	102
7.3.2.	Evaluation of results considering requirements coverage	105
7.3.3.	Evaluation of results considering KPIs	105
7.4.	Use case scenario HIB_UCS3 – AI-enabled new versions of assets analysis	106
7.4.1.	Summary of preliminary results.....	107
7.4.2.	Evaluation of results considering requirements coverage	107
7.4.3.	Evaluation of results considering KPIs	107
7.5.	Use case scenario HIB_UCS4 – Deployment and analysis of new HW/SW versions	107
7.5.1.	Summary of preliminary results.....	108
7.5.2.	Evaluation of results considering requirements coverage	108
7.5.3.	Evaluation of results considering KPIs	108
7.6.	Planned improvements	108
7.7.	Planned demonstration	109
8.	PRO_CS07 case study “Smart Port Platform Monitoring (SPPM)”	110
8.1.	Case Study description	110
8.2.	Use case scenario PRO_UCS1 — Infrastructure as code	111
8.2.1.	Summary of preliminary results.....	112
8.2.2.	Evaluation of results considering requirements coverage	112
8.2.3.	Evaluation of results considering KPIs	113
8.3.	Use case scenario PRO_UCS2 - Automatic validation of the platform provisioning	113
8.3.1.	Summary of preliminary results.....	113
8.3.2.	Evaluation of results considering requirements coverage	114
8.3.3.	Evaluation of results considering KPIs	114
8.4.	Use case scenario PRO_UCS3 - Detection of anomalies of the platform performance	114
8.4.1.	Summary of preliminary results.....	115
8.4.2.	Evaluation of results considering requirements coverage	115
8.4.3.	Evaluation of results considering KPIs	116
8.5.	Use case scenario PRO_UCS4 – Detection and correction of service interruptions	119
8.5.1.	Summary of preliminary results.....	119

8.5.2.	Evaluation of results considering requirements coverage	119
8.5.3.	Evaluation of results considering KPIs	119
8.6.	Use case scenario PRO_UCS5 – Resizing of resources based on current workload	121
8.6.1.	Summary of preliminary results.....	121
8.6.2.	Evaluation of results considering requirements coverage	126
8.6.3.	Evaluation of results considering KPIs	126
8.7.	Planned improvements.....	128
8.8.	Planned demonstration	129
9.	TEK_CS08 case study “Agile process and Electric/Electronic Architecture of a vehicle for professional applications”	131
9.1.	Use case scenario TEK_UCS_01 — Design choices verification	132
9.1.1.	Summary of preliminary results.....	134
9.1.2.	Evaluation of results considering requirements coverage	135
9.1.3.	Evaluation of results considering KPIs	137
9.2.	Use case scenario TEK_UCS_02 — Run-time verification.....	138
9.2.1.	Summary of preliminary results.....	138
9.2.2.	Evaluation of results considering requirements coverage	139
9.2.3.	Evaluation of results considering KPIs	139
9.3.	Use case scenario TEK_UCS3 — Operating life monitoring.....	140
9.3.1.	Summary of preliminary results.....	140
9.3.2.	Evaluation of results considering requirements coverage	141
9.3.3.	Evaluation of results considering KPIs	142
9.4.	Planned improvements.....	142
9.5.	Planned demonstration	143
10.	VCE_CS09 case study “Data modelling to support product development cost and efficiency” 144	
10.1.	Case Study description.....	144
10.2.	Use case scenario VCE_UCS_01 — Modelling system, software, data architectures	148
10.2.1.	Summary of preliminary results.....	148
10.2.2.	Evaluation of results considering requirements coverage	149
10.2.3.	Evaluation of results considering KPIs	150
10.3.	Use case scenario VCE_UCS_02 — Validation and verification of architecture models	151
10.3.1.	Summary of preliminary results.....	152
10.3.2.	Evaluation of results considering requirements coverage	152
10.3.3.	Evaluation of results considering KPIs	153
10.4.	Use case scenario VCE_UCS_03 — AI augmented DevOps workflow and data analytics... ..	154
10.4.1.	Summary of preliminary results.....	154
10.4.2.	Evaluation of results considering requirements coverage	156
10.4.3.	Evaluation of results considering KPIs	156

10.5. Planned improvements	157
10.6. Planned demonstration	158
11. WESTMO_CS10 case study “Automated continuous decision making in testing of robust and industrial-grade network equipment”	159
11.1. Case Study description	159
11.2. W_UCS_1 — AI-Augmented DevOps development process	161
11.2.1. Summary of preliminary results.....	161
11.3. W_UCS_2 — AI-powered root cause analysis w.r.t. functional issues	164
11.3.1. Summary of preliminary results.....	165
11.4. W_UCS_3 — AI-powered root cause analysis w.r.t. non-functional quality	167
11.4.1. Summary of preliminary results.....	169
11.5. W_UCS_4 — AI-powered log analysis/root cause analysis	171
11.5.1. Summary of preliminary results.....	172
11.6. Requirements Coverage.....	172
11.7. KPI Evaluation.....	174
11.8. Planned improvements	176
11.9. Planned demonstration	177
12. Conclusions	179
13. Referenced documents	181

Key Terminology Abbreviations

Abbreviations	Terminology
AI	Artificial Intelligence
AIOps	Artificial Intelligence for IT Operations
ANN	Artificial Neural Network
APFD	Average Percentage of Fault Detected
CI/CD	Continuous Integration and Continuous Delivery
CPS	Cyber-Physical Systems
DevOps	Development Operations
DL	Deep Learning
DSL	Domain System Language
FMU	Functional Mock-Up Unit
FPGA	Field Programmable Gate Arrays
HiL	Hardware-in-the-Loop
HL	High Level
HW	Hardware
IaC	Infrastructure as a Code
KPI	Key Performance Indicator
LPTN	Lumped Parameter Thermal Networks
LTL	Linear Temporal Logic
MBE	Model-Based Engineering
MDE	Model-Driven Engineering
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
ODD	Operational Design Domain
ONNX	Open Neural Network Exchange
PHM	Prognostics and Health Management
PO	Proof of Obligations
PSP	Property Specification Patterns
RNN	Recurrent Neural Network
SW	Software
TOS	Terminal Operating System
UC	Use Case
V&V	Verification and Validation
VNN-COMP	Verification of Neural Networks Competition
WP	Work Package

1. Introduction

1.1. Scope

This document, deliverable D5.7 “Use cases evaluation report – 1”, describes the preliminary activities and results of task T5.3 “Use Cases Evaluation” of work package WP5 “Integration and Use Case Evaluation” of the AIDOaRt project.

The document is about the evaluation of the development that individual partners have made by project month M30 (September 2023).

1.2. Project overview

The AIDOaRt project aims at providing a suite of tools that improves the entire design and life cycle of software systems, from the requirement analysis to the operating phase. The keywords, that are in the extended name of the project, are “Artificial Intelligence augmented automation”. More automation means less effort, fewer errors, definitively lower costs. For this purpose, AIDOaRt leverages Artificial Intelligence techniques. The tools are developed during the project, or improved starting from previous versions, by the AIDOaRt partners that are grouped under the name “*solution providers*”.

Another group of partners participates in the project, the “*case study providers*”, that can be considered as end users of the tools (in some cases some partners play both roles). These partners proposed, on the basis of their specific industrial and technical interests, the systems that they want to develop using the tools. Of course, the interaction between the two groups has iterative and incremental aspects, because during the project the tools are used as they are developed.

The development of the systems constitutes the “*case study*”, a term borrowed from the social sciences that means semi-empirical verification and validation. In this project, we strive for a quantitative evaluation of verification and validation of the tools (as an AIDOaRt outcomes).

It is worth noting that the improvements aimed by AIDOaRt are not limited to the development process and to the quality of systems, but in some cases extend to the functionalities: the boundary between tools and components blurs, because both enable advanced system features.

In order to better understand the case studies and describe the potential tools and solutions, we iteratively proceeded in a model-based approach for requirements elicitation, tools functional description and a common framework architecture specification. This was accomplished in the various work packages as illustrated in Figure 1.1.

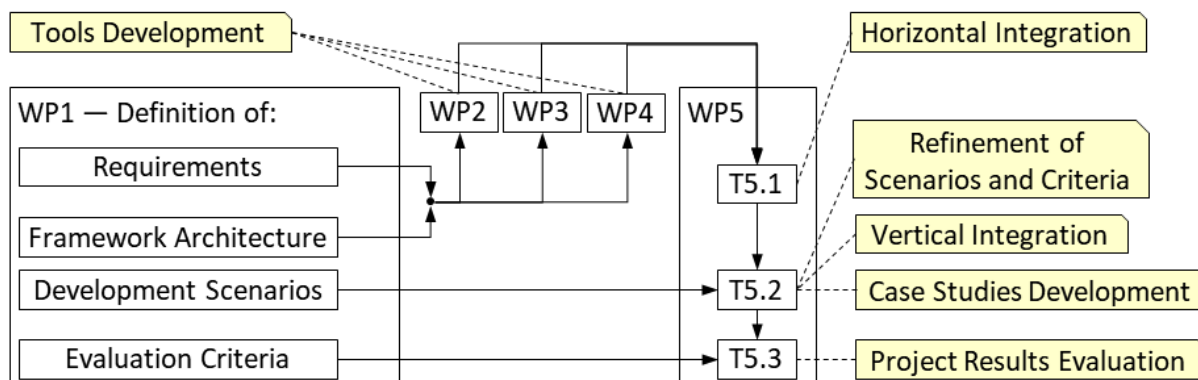


Figure 1.1 AIDOaRt workflow

In WP1, the case study providers, as end users, cooperated with the solution providers to define the requirements of the AIDOaRt tools—deliverable D1.1 [1]. The former specified the case studies, each of which was divided in *use case scenarios* (referred also as *development scenario*)—deliverable D1.3 [2]. The latter defined the architecture of the tools structured in a framework—deliverables D1.2 [3] and D1.4 [4].

The solution providers have continued their work in the *technical* work packages WP2, WP3, and WP4 where they develop the tools that integrate horizontally in the task T5.1 of WP5—deliverable D5.2 [7].

In task T5.2 of WP5, whose first results were presented in D5.6 [10] deliverable, the case study providers collaborated with the solution providers, for the vertical integration of the tools in their development processes, and develop their systems.

The collaboration extended to the T5.3 of WP5 where, on the basis of the use of the tools in the case studies, the AIDOaRt outcomes are evaluated.

This D5.7 deliverable reports about the first phase of evaluation of the case study development, specifically evaluating results considering requirement coverage and KPIs for individual UC scenarios. This report also contains planned improvements for the next development phase and plans for demonstration at the end of the project.

This document will be followed by D5.8 [10] for the final report of the case study development in M34, and D5.9 [11][11] for the final evaluation in M36.

1.3. Case Studies

Table 1.1 outlines the AIDOaRt Case Studies.

Table 1.1 Case studies of AIDOaRt

ID	Partner	Case Study Name	Domain
CS01	ABI (Abinsula SRL)	Safety critical systems in the automotive domain using disruption technology	Automotive
CS02	AVL (AVL List GmbH)	AI supported Digital Twin Synthesis supporting secure vehicle development and testing for novel propulsion systems	Automotive
CS03	BT (Bombardier Transportation) ¹	DevOps for Railway Propulsion System Design	Railway
CS04	CAMEA (CAMEA, spol. s r.o.)	AI for Traffic Monitoring Systems	Traffic management
CS05	CSY (CLEARSY SAS)	Machine learning in interactive proving	Development tools
CS06	HIB (HI Iberia Ingeniería y Proyectos S.L.)	AI DevOps in the restaurants business	Catering
CS07	PRO (Prodevelop SL)	Prodevelop - Smart Port Platform monitoring	Maritime
CS08	TEK (TEKNE SRL)	Agile process and Electric/Electronic Architecture of a vehicle for professional applications	Automotive
CS09	VCE (Volvo Construction Equipment AB)	Data modelling to support product development cost and efficiency	Construction Equipment
CS10	WMO (Westermo Network Technologies AB)	Automated continuous decision making in testing of robust and industrial-grade network equipment	Development tools

1.4. Project level and Case Studies KPIs

This section explains the common structure and usage of the AIDOaRt KPIs (Key Performance Indicators), whose measurements give a quantitative dimension to the evaluation of the project results.

The **project level KPIs** are reported in Table 1.2.

Table 1.2 Project level KPIs of AIDOaRt

Identifier	Project KPI definition	Target
KPI_1.1	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	25%
KPI_1.2	Improvement of the early detection of system deviations.	30%
KPI_2.1	Reduction of the time/effort required for managing and handling all the involved DevOps models.	30%

¹ Bombardier Transportation (BT) was acquired by Alstom. The identifier of the case study is “ALSTOM_CS03”. The identifiers of the other elements (e.g. use case scenarios) of the case study have “BT” as initial letters.

Identifier	Project KPI definition	Target
KPI_2.2	Increase in the number of available data sources to be actually managed and handled in existing engineering practices.	25%
KPI_2.3	Reduction of time/effort required for integrating new practices and services into a DevOps pipeline.	20%
KPI_3.1	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	30%
KPI_3.2	Increase the coverage and quality of actionable feedback for the next DevOps iteration.	25%
KPI_4.1	Increase in the percentage of parts of the DevOps process covered in the Use Cases with productivity improvement.	30%
KPI_4.2	Reduction of deviations from the specifications to improve predictability, conformance to specifications and proposal of system design refinements.	30%
KPI_5.1_1	Number of external manufacturers to which AIDOaRt will be presented.	20
KPI_5.1_2	Number of external manufacturers to which AIDOaRt will be presented and will interact and try our solution.	5
KPI_5.2	Number of presentations of the AIDOaRt technologies in the most important international open source forums.	5
KPI_6.1	Growth in systems sales for commercial partners in 3 years.	15%
KPI_6.2	Number of organised public workshops, hackathons and dissemination events to raise awareness on the opportunities of AIOps for European companies.	3

Note (1): The *project level KPIs* are considered *abstract classes* from which to derive the *case study KPIs* which are the ones actually measured. The reason for this specialisation is the wide difference among the case studies: the features of the system that is developed, the subset of the AIDOaRt tools suite that is involved in this development, and the industrial interests and processes of the partner that carries out the case study.

Note (2): Only the KPIs from KPI_1.1 to KPI_4.2 are directly related to the case study and consequently treated in this document.

Table 1.3 is an example of the specification of a **case study KPI**.

Table 1.3 Example of case study KPI specification

KPI Identifier: <identifier>		Scenario Identifier: <identifier >		
KPI Description:	Improve the detection of problems in the design phase.			
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the early detection of system deviations.		
	<i>Identifier:</i>	KPI_1.2	<i>Target</i>	≥ 30 %
KPI Measure:	Number of defects that derive from design problems as percentage of the total number of defects.			
KPI Baseline:	<i>Source:</i>	Project management documents.		
	<i>Value: k₀ =</i>	10 %		

KPI Identifier: <identifier>			Scenario Identifier: <identifier >		
Target:	<i>Decrease:</i>		$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	\geq	30 %
D5.6 Measure:	<i>Value: $k_6 =$</i>	8 %	$100 \cdot \Delta_6 / k_0 = 100 \cdot (k_0 - k_6) / k_0$	=	20 %
D5.7 Measure:	<i>Value: $k_7 =$</i>	6 %	$100 \cdot \Delta_7 / k_0 = 100 \cdot (k_0 - k_7) / k_0$	=	40 %

In what follows, we provide the description of the fields of Table 1.3:

- **KPI Description** — Generic description of the case study KPI. The name of this field is “KPI Instantiation and Definition” in D5.5 [8].
- **Refined AIDOaRt KPI** — Project level KPI (see Table 1.2) from which the case study KPI is derived. The sub-fields “Description”, “Identifier” and “Target” are reported from Table 1.2 so that the case study KPI specification becomes self-contained
- **KPI Measure** — How the case study KPI is measured.
- **KPI Baseline** — k_0 is the value of the case study KPI before AIDOaRt. The sub-field “Source” describes how k_0 is evaluated (e.g. form documents, dashboards, interviews with experts).
- **Target** — k is the value of the KPI measured in the case study. If the improvement is a “Decrease” (e.g. of effort, of number of errors, etc.) as in Table 1.2, then the difference is $\Delta = (k_0 - k)$. If the improvement is an “Increase” (e.g. capacity), then the difference is $\Delta = (k - k_0)$. At the right end of the row *Target* there is the case study KPI improvement, expressed as percentage, that is planned to be achieved at the end of AIDOaRt.
- **D5.6 Measure** — Where possible the row “D5.6 Measure” gives some insights about the intermediate evaluation that was carried out in D5.6, otherwise it is omitted. At the right end of the row there is the improvement computed from k_6 that is the measured value.
- **D5.7 Measure** — Where possible the row “D5.7 Measure” gives some insights about the intermediate evaluation that is reported here in D5.7, otherwise it is omitted. At the right end of the row there is the improvement computed from k_7 that is the measured value.

Note (3): The differences among the KPI specifications in this document and those in the past deliverable D5.5 [8] are properly flagged and explained.

1.5. Document overview

After this Introduction Section, there is **one Chapter for each Case Study**. It reports the status, at the project month M30, of the system whose development constitutes the Case Study. Each Chapter has the following sections:

1. The Section “Case Study Description” summarises the case study and the motivations the partner had—and still has—for proposing it in the AIDOaRt project. It gives the synopsis of the case study: (1) organisation in Use Case scenarios, (2) collaboration with the Solution Providers in terms of Case Study requirements and used tools, and (3) the KPIs considered in the Case Study.
2. There is one Section for each development scenario of the Case Study. It describes: (1) summary of the preliminary results reached so far, (2) evaluation of results considering

requirements coverage, and (3) evaluation of results considering KPIs. Mainly Solution Providers contributed to this description and, when appropriate, they illustrated the progress of their work.

3. The section “KPIs” specifies the metrics, baselines, and target values of the key performance indicators that will be used in the evaluation of the Case Studies.
4. The Section “Planned improvements” summarises suggested ideas (of UC and also Solution Providers) that will be used in the next development phase to fulfil the defined KPIs.
5. The section “Planned demonstration” (also collaboration of UC and Solution Providers) suggests ideas for final demonstration of the project results.

Chapter “Conclusions” closes the document.

2. ABI_CS01 case study “Safety critical systems in the automotive domain using disruption technology”

2.1. Case Study description

The Abinsula (ABI) case study presents a virtual rear-view mirror scenario in which multiple cooperative cameras are used to capture the context outside the vehicle, by means of AI-based technology (see deliverable D5.6 for a detailed description). In this case study, we consider two main macro challenges:

- 1) **ABI-CH1** — To guarantee the predictability of AI-based systems, using formal verification with respect to given specifications and guidelines is necessary. This also includes the need for formally verifying any Neural Network (NN) adopted in the system. To complete these features, a suite of tests should be defined to check whether a reactive system complies with a set of requirements. From ABI-CH1, the following sub-challenges can be derived:
 - a) **ABI-CH1.a** — To guarantee formal verification with respect to given specifications.
 - b) **ABI-CH1.b** — To guarantee formal verification with respect to given guidelines.
 - c) **ABI-CH1.c** — To guarantee formal verification of the NNs adopted in the system.
 - d) **ABI-CH1.d** — To test the compliance of the system with respect to requirements.
- 2) **ABI-CH2**. The system is expected to autonomously react according to the external stimuli and internal needs. Therefore, it should be able to adapt itself and continue operation with a reduced number of cameras, as well as it should be able to use AI and ML techniques for image processing to detect and to signal possible hazards. In this regard, there is also a need for measuring the reliability of the AI and ML algorithms in a humanly interpretable way, in a safety-critical context. From ABI-CH2, the following sub-challenges can be derived:
 - a) **ABI-CH2.a** — Realise an adaptive multi-camera system.
 - b) **ABI-CH2.b** — Implement video elaboration based on AI/ML techniques.
 - c) **ABI-CH2.c** — Measuring the reliability of the AI and ML algorithms in a humanly interpretable way.

In AIDOaRt, Abinsula is collaborating with the University of Sassari (UNISS) and Intecs Solutions (INT) to study the adoption of formal methods in the automotive domain, to support the predictability of AI based systems and loosen the current technological limitations and enable the possibility of freely playing with all the available technology in the development of future cars. The synopsis of the case study is in Table 2.1.

Table 2.1 Synopsis of the case study ABI_CS01

Use case scenario ABI_UCS1 — Functionality Verification	
Description:	The Functionality Verification refers to the automatic verification of requirement and model consistency against properties.
Requirements:	<ul style="list-style-type: none"> • Directly addresses: ABI_R01, ABI_R02.
Tools:	UNISS_SOL_01, UNISS_SOL_02, UNISS_SOL_04 (UNISS)
Use case scenario ABI_UCS2 — Test Definition	
Description:	The Test Definition refers to the automatic generation of a suite of tests for the system.
Requirements:	<ul style="list-style-type: none"> • Directly addresses: ABI_R03.
Tools:	UNISS_SOL_03 (UNISS).
Use case scenario ABI_UCS3 — Compliance Verification	
Description:	The Compliance Verification refers to the automatic verification of requirement consistency with respect to a reference guideline.
Requirements:	<ul style="list-style-type: none"> • Directly addresses: ABI_R05.
Tools:	UNISS_SOL_01, UNISS_SOL_04 (UNISS).
Implementation activities transversal to ABI use case scenario	
Description:	Implementation activities, transversal to the use case scenarios, that mainly involve the development and integration of AI-based applications for image processing.
Requirements:	ABI_R04, ABI_R06, ABI_R07, ABI_R08, ABI_R09, ABI_R010, ABI_R11, ABI_R12
Tools:	INT-DET, INT-DEPTH, INT-XAI (INT)
Evaluation	
KPI:	The KPI defined for ABI_CS01 is related to the case study in general and not only to a specific use case scenario. Details are reported in Section 2.7.

Please, notice that, with respect to deliverable D5.6, the solution UNISS_SOL_05 (UNISS) has been discarded. This does not affect the case study as its functionalities are absorbed by UNISS_SOL_01 and UNISS_SOL_04. In particular, the functionality related to the consistency verification of technical specifications with respect to a guideline will be covered by UNISS_SOL_01 through the use of formal methods combined with NLP techniques.

2.2. Use case scenario ABI_UCS1 — Functionality Verification

This section provides a brief description of the ABI_UCS1, and a summary of preliminary results. For details refer to deliverable D5.6 [9].

The UCS1 corresponds to the Functionality Verification step of this case study and involve mainly:

- Automated verification of the system with respect to given requirements and properties (addresses **ABI-CH1.a**).

- The formal verification of Neural Networks, to guarantee that NNs satisfy stated input-output relations (addresses **ABI-CH1.c**).

The solutions involved in ABI_UCS1 that mainly address **ABI-CH1.a** are UNISS_SOL_01 and UNISS_SOL_04, while UNISS_SOL_02 mainly addresses **ABI-CH1.c**. UNISS_SOL_01 provides automated consistency verification of technical specifications of safety-critical systems and UNISS_SOL_04 provides automated formal consistency verification of a system design expressed language. UNISS_SOL_02 is aimed at providing automated verification of NNs with the goal of striking a balance between maintaining accuracy and robustness while making the resulting networks amenable to formal analysis (refer to deliverable D4.1 [6] for details).

2.2.1. Summary of preliminary results

So far, two main investigations have been carried out in ABI_UCS1, *Modelling properties and requirements* and *Formal verification of NNs*.

Modelling properties and requirements

UNISS is working on the development of two different tools aimed at verifying the consistency of requirements (UNISS_SOL_01) and system properties (UNISS_SOL_04) automatically.

UNISS proposed the use of formal languages to model requirements and system properties since they provide a rigorous framework for specifying and analysing system behaviours, ensuring clarity, precision, consistency and correctness in the design process. In the context of UNISS_SOL_01, continuing the work presented in D5.6, we involved leveraging Property Specification Patterns (PSPs) to translate a set of Abinsula requirements expressed in natural language into formal specifications. PSPs are reusable templates/structures (see Figure 2.1) that capture common properties or constraints in system requirements. Information in the patterns is based on classifying the patterns in terms of system behaviours' types they describe, considering the simple occurrence of a behaviour (Absence, Universality, Existence, Bounded Existence) or the relation of a behaviour with respect to another (Precedence, Response, Chain Response, Chain Precedence).

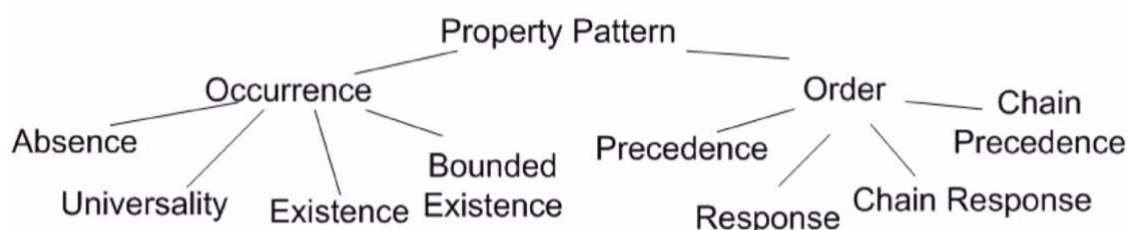


Figure 2.1. PSP: Classification of the patterns, according to the kinds of system behaviours they describe.

These patterns provide a systematic way to express and analyse properties in a formal specification language, and each pattern has a scope, which is the extent of the program execution over which the pattern must hold, e.g. global, before, after, between, after until.

The considered set of Abinsula requirements is composed of 46 atomic requirements coming from specific guidelines (such as the ISO 16505:2019) and from customers' needs. The first step of this work

was to bridge the gap between controlled natural language requirements and the formal specifications necessary for rigorous system development. PSPs are able to capture common structures and properties that frequently occur in system requirements, providing a reusable and systematic approach to transform natural language expressions into formal representations.

The process begins by analysing natural language requirements and identifying recurring patterns or structures within them. These patterns may involve properties like safety, synchronisation or timing constraints. By recognizing these patterns, we developed a catalogue of 42 property specification patterns that align with the specific domain. Next, we applied the identified patterns to the natural language requirements. They map the linguistic elements and structures in the requirements to the corresponding elements in the formal specification language. This step involved understanding the semantics of the patterns and their relationships to the desired system properties. Once the patterns are applied, a formal specification is obtained. This specification can be in the form of a mathematical logic formula, a temporal logic expression, or any other formal language suitable for system analysis and verification. The formal specification captured the essential properties and constraints originally expressed in natural language, but in a precise and unambiguous manner. This approach provided a systematic and consistent way to convert requirements into formal specifications, enabling rigorous analysis and verification of system behaviours.

Example: From requirement to PSP

Let's consider one of the Abinsula requirements, coming from the ISO 16505:2019.

Field of view: The field of view of the Camera Monitoring System shall cover the field of view at least that is required by the national body for conventional mirrors of the same class, both in horizontal and vertical direction.

- Firstly, this requirement has been partitioned into three atomic requirements, expressed in controlled natural language.
- Then, each one has been analysed using the PSP templates, to identify its scope and pattern.
- Finally, each requirement has been conferred into a formal specification.

Atomic requirement	Scope	Pattern	Requirement in PSP
IF Class EQUAL 1 AND IF default view THEN vertical vision distance MUST be EQUAL or GREATERTHAN 60 m behind driver	Global	Occurrence - Universality	Globally, it is always the case that if class = 1 and default_view holds, then vertical_vision_distance >= 60 holds as well
IF Class EQUAL 1 AND IF default view AND vertical longitudinal median plane EQUAL or GREATERTHAN 60 m THEN horizontal vision MUST be EQUAL or GREATERTHAN 20 m	Global	Occurrence - Universality	Globally, it is always the case that if class = 1 and default_view and vertical_longitudinal_median_plane >= 60 holds, then horizontal_vision >= 20 eventually holds.

IF Class EQUAL 3 AND IF default view THEN vertical vision distance MUST be EQUAL or GREATER THAN 20 m behind driver/passenger	Global	Occurrence - Universality	Globally, it is always the case that if class = 3 and default_view holds, then vertical_vision_distance >= 20 holds as well.
IF Class EQUAL 3 AND IF default view AND vertical longitudinal median plane EQUAL or GREATER THAN 4 m THEN vertical vision MUST be EQUAL or GREATER THAN 1 m	Global	Occurrence - Universality	Globally, it is always the case that if class = 3 and default_view and vertical_longitudinal_median_plane >= 4 holds, then vertical_vision >= 1 holds as well.
IF Class EQUAL 3 AND IF default view AND vertical longitudinal median plane EQUAL 20 m THEN horizontal vision MUST be EQUAL or GREATER THAN 4 m	Global	Occurrence - Universality	Globally, it is always the case that if class = 3 and default_view and vertical_longitudinal_median_plane >= 20 holds, then horizontal_vision >= 4 holds as well.

While the use of PSP offers significant benefits in translating requirements from natural language to formal specifications, there can be challenges and difficulties associated with their use. In fact, choosing the appropriate PSP for the given set of requirements can be challenging, since it requires a deep understanding of the domain, the system being developed, and the available patterns. Selecting the wrong pattern or failing to identify the relevant patterns can lead to inaccurate or incomplete formal specifications. Moreover, sometimes PSP may not cover all possible types of requirements or properties, necessitating the creation of new patterns or modifications to existing ones. Finally, interpreting and applying PSPs can be difficult, especially when dealing with complex or ambiguous natural language requirements. The mapping process from natural language to formal language requires careful analysis and understanding of the patterns' semantics, which may be subject to interpretation.

Experimental analysis. The main focus of the experimental analysis was to automatically formally verify the consistency of the 42 requirements expressed in PSP by means of ReqV², for formal consistency checking of requirements. Automatising the requirements validation process is important to streamline the verification phase and reduce potential human errors. ReqV is an open source tool for the formal consistency checking of requirements. The main goal of the tool is to provide an easy-to-use environment to enable users with no background knowledge of formal methods and logic languages to write and verify requirements, expressed as a list of properties specification patterns. It provides an intuitive interface (see Figure 2.2), accessible within a web browser, for writing requirements in PSP and automatically translate them in Linear Temporal Logic (LTL) in order to check

² ReqV is available for download at this link: https://qbflib.org/VMPROSSIMO/Prossimo_Sage.ova

their inner consistency by means of a model checker (i.e. NuSMV). In case of inconsistency, ReqV can also extract a minimal set of conflicting requirements to help designers in correcting the specification.

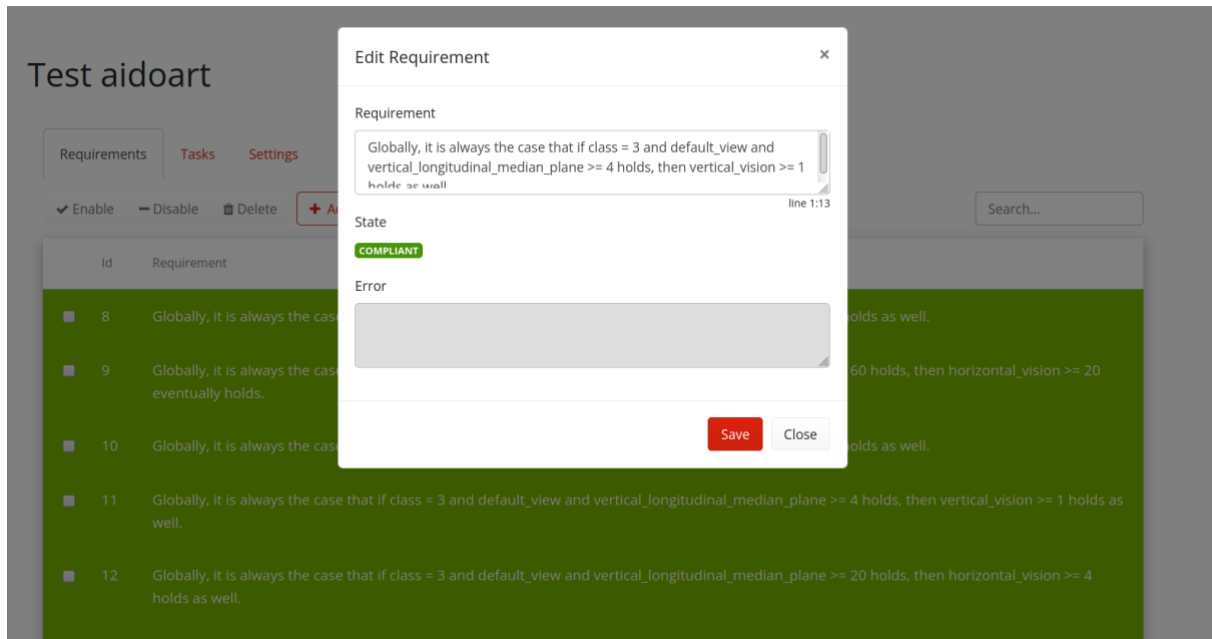


Figure 2.2. List of requirements inserted by the user in a project. Requirements coloured in green are syntactically correct, the ones in red are not. A grey requirement indicates it has been disabled and therefore not considered during the analysis.

The set of requirements involved in our analysis were initially 42 and it includes 25 boolean signals and 26 numerical ones. Unfortunately, we had to exclude from the analysis 8 requirements containing ratio and percentage type signals, as they are not supported by the tool, by reducing the set of requirements for the analysis to 34. Initially, we attempted to process all 42 requirements, disabling the 8 requirements with unsupported signals, however this resulted in excessively long processing times. Therefore, we partitioned the requirements into three groups to expedite the processing, which, in this case, proved to be extremely swift. The first group, consisting of 10 requirements, was processed in about 3 CPU seconds. The second group of 7 requirements was processed in 3 CPU seconds, while the third group consisting of 17 requirements was processed in 11 CPU seconds. All the tested requirements turned out to be formally consistent. When it terminated, ReqV returned the results, as shown in Figure 2.3. In Table 2.2 we present a synopsis of the requirements, to give an idea of the kind of patterns used in the specification experimental analysis.

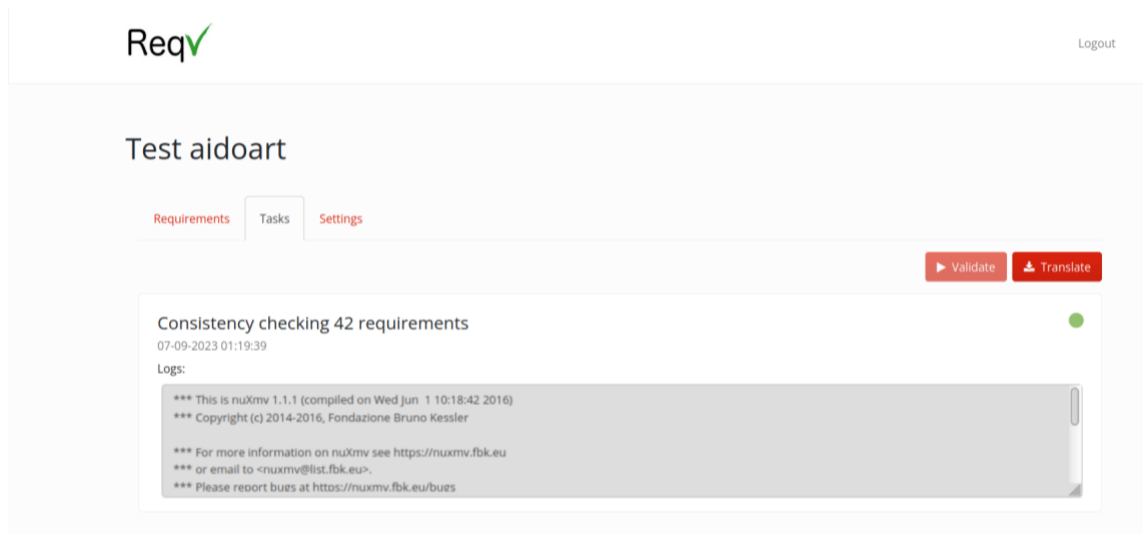


Figure 2.3. Report of the consistency checking task in ReqV.

Table 2.2. Case study requirements synopsis. The table is organised as follows: the first column reports the name of the patterns and it is followed by one group of two columns denoted with the scope type. Each cell in these two columns reports the number of requirements grouped by pattern and by scope type.

Pattern	Specification	
	Globally	After
Universality	14	
Existence	5	
Response	1	13
Absence	1	

Concerning the part of the automatic verification of system properties, which relates to the development of UNISS_SOL_04, we are currently experimenting with an ontology-based approach to support formal verification of system design. After the definition of the main components and properties of the Smart Camera Monitoring System (in Figure 2.4, we report a simplified graphical representation of the main ABI system model), we implemented it by means of ontology. Ontologies are well-suited for capturing domain knowledge data, deriving requirements, providing analysis, and developing applications. They provide representations of domain knowledge by defining concepts and their relationships and can be used both alone or with other ontologies since they offer interoperability solutions to data heterogeneity problems, providing a factorization benefit to knowledge so it can be reused in different projects. Specifically, using an ontology in software engineering offers two key benefits. Firstly, it establishes a shared vocabulary within specific domains, allowing software developers working on diverse applications to communicate effectively. Secondly, by representing the software model and user requirements as an ontology, one can employ an inference engine to automatically verify requirement satisfaction. There are several languages for representing ontologies, including the Web Ontology Language (OWL), which is the one used in this context. The tool used for

the creation of the ontology was Protégé and in Figure 2.5 we show the main structure of the ontology. Tools such as Protégé, with intuitive interfaces and the use of user-friendly constructs like Manchester syntax, provide easy access to a presumably complex and involved formalism. This allows designers, even without specialised knowledge in ontology development, to easily represent the system with its constraints and validate it through the use of integrated reasoners within the tool.

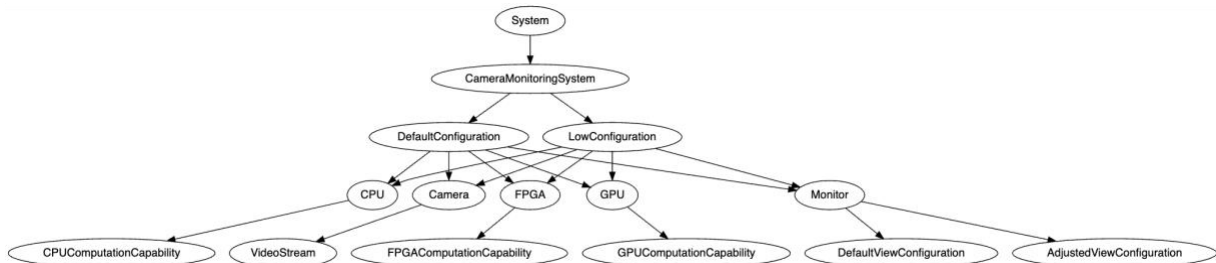


Figure 2.4. Representation of a simplified version of the Smart Camera Monitoring System

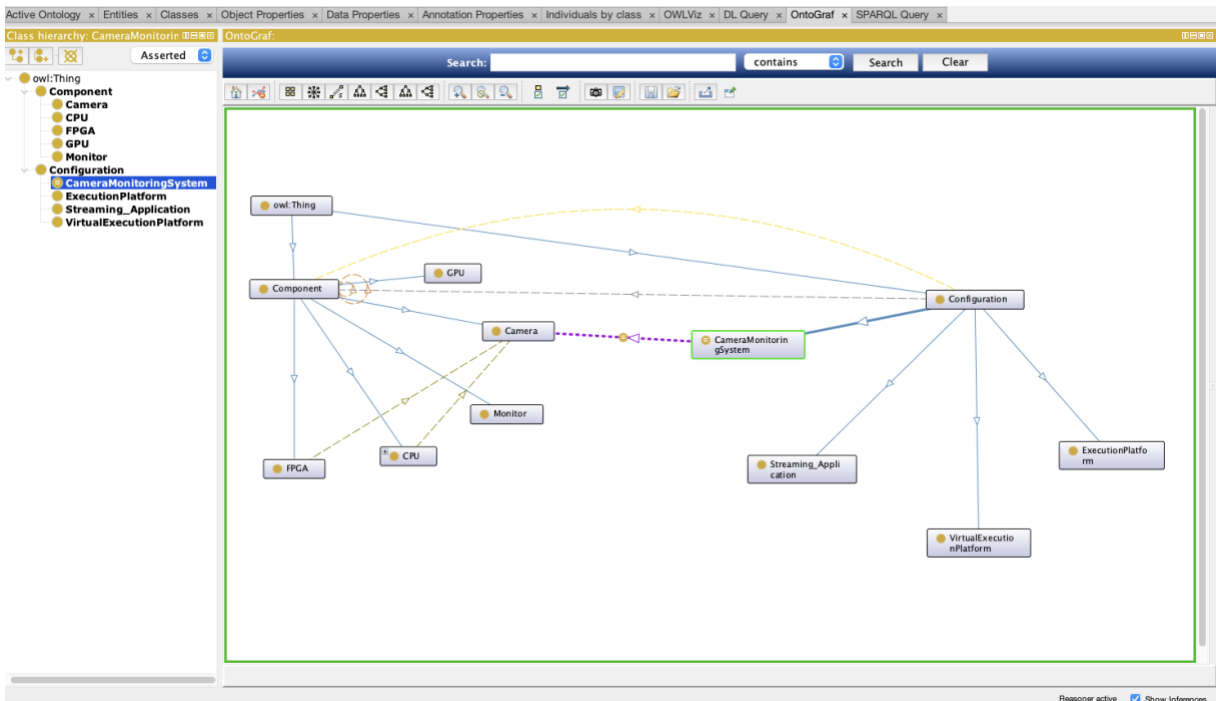


Figure 2.5. Main structure of the ontology in Protégé.

We report the metrics of the ontology in the following Table 2.3.

Table 2.3. Ontology metrics.

Metric name	Property
DL expressivity	<i>ALCHIQ(D)</i>

Class count	11
Object property count	18
Data property count	4
Axiom	134

Formal verification of NNs

UNISS investigated how the recent advancements in Satisfiability Modulo Theory (SMT) technologies may be leveraged to allow for the verification of NNs with non-linear activation functions. In particular, UNISS focused on a computer vision task that, while less complex than the one considered in the ABINSULA case study, is still able to give us insight into the feasibility of the considered approach while keeping the computational resources needed for the training of the networks to a bearable level.

SMT is a field of automated reasoning that focuses on determining whether a logical formula is satisfiable using a combination of decision procedures for different theories. It has been used as a key component in the development of powerful tools such as theorem provers, model checkers, and symbolic execution engines.

While leveraging SMT solvers for verifying neural networks is not a completely new idea (the first paper regarding this kind of application dates back to 2010), the innovations in the domain of SMT solvers occurred in the last ten years justify the need for a more recent experimental evaluation.

In this context, UNISS developed a set of benchmarks based on different network architectures and properties of interest and we used them to test the performances of four different SMT solvers chosen between the winner and runner-ups of the quantifier-free non-linear arithmetic division in the single query track of the international SMT competition (SMTCOMP 2022).

In particular, UNISS selected the CVC, MathSAT, Z3 and Yices solvers: the first two support both piecewise linear and transcendental functions, whereas the others support only piecewise linear ones. Therefore, it was possible to test on all four solvers only the benchmarks presenting ReLU activation functions: the remaining ones were tested only on CVC and MathSAT.

We chose to focus on convolutional neural networks since they are best suited for computer vision tasks: in particular, we considered networks whose feature extractors present the same architecture, whereas their classifiers differ in the choice of activation functions, layers size, and layers number. The activation functions considered were the Rectified Linear Unit (ReLU), the hyperbolic tangent and the logistic function. We focused on the verification of the classifier instead of the whole network since the feature extractor component usually has high complexity and is often regarded as a black box: it could be, for example, easily replaced with a non-neural feature extraction algorithm like SIFT.

The property we were interested in verifying is the robustness of the classifier to local perturbations of its input vector, known in the literature as adversarial perturbations, which may cause the misclassification of the same. In practice, we tried to prove that there exists at least an image whose

Chebyshev distance from the original one is less than a constant value (Epsilon) and which is incorrectly classified by the neural network of interest. If such property is proven to be unsatisfiable then it means that the network is robust to the local perturbation of interest.

In Table 2.4 a schematic representation of our benchmarks with the related parameters of interest is shown. As can be seen in the column **Accuracy** and the related sub-columns, all the networks presented satisfactory performances on the task considered, that is, were able to correctly classify more than 96% of the images composing the test set.

Table 2.4. Relevant data for the verification benchmarks. The column Architecture reports the number of neurons for each intermediate fully-connected layer of the classifier, the column Accuracy reports the percentage of test set images classified correctly by the networks, the column Epsilon reports the magnitude of the perturbation considered in the benchmark, and the column Benchmark ID represent the identifier assigned to the corresponding benchmark. The subcolumns ReLU, Logi and Tanh indicate that the reported values are related to the network architecture using the corresponding activation functions.

Architecture	Accuracy			Epsilon	Benchmark ID		
	ReLU	Logi	Tanh		ReLU	Logi	Tanh
[16]	97.66%	97.10%	97.44%	0.001	B_000	B_015	B_030
				0.01	B_001	B_016	B_031
				0.1	B_002	B_017	B_032
[32]	97.10%	97.22%	97.14%	0.001	B_003	B_018	B_033
				0.01	B_004	B_019	B_034
				0.1	B_005	B_020	B_035
[16-8]	97.00%	96.25%	96.73%	0.001	B_006	B_021	B_036
				0.01	B_007	B_022	B_037
				0.1	B_008	B_023	B_038
[32-16]	97.14%	96.56%	97.01%	0.001	B_009	B_024	B_039
				0.01	B_010	B_025	B_040
				0.1	B_011	B_026	B_041
[64]	97.58%	97.49%	97.50%	0.001	B_012	B_027	B_042
				0.01	B_013	B_028	B_043
				0.1	B_014	B_029	B_044

In Table 2.5 we report the results of our experimental evaluation. Column **Benchmark ID** reports the identifier of the benchmark tested, while column **Times** reports the time needed, in seconds, to solve the benchmarks by the solvers of subcolumns *MathSAT*, *CVC5*, *Z3*, *Yices2*. Column **Results** report the response of the solvers regarding the query of interest. The symbol “-” means that the time limit of one hour was reached while trying to solve the query, whereas “N.S.” indicates that the solver did not

support the activation functions used in the query. We do not report in the table the results for the benchmarks from B_016 to B_044 because Z3 and Yices do not support transcendent activation function and both CVC and MathSAT were unable to solve any benchmark before the timeout. Finally, in Figure 2.6, we try to clarify the results presented in Table 2.3 using a graphical representation.

Table 2.5.: Experimental results obtained by testing our benchmarks on the SMT solvers of interest.

Benchmark ID	Times				Results			
	<i>MathSAT</i>	<i>CVC5</i>	<i>Z3</i>	<i>Yices2</i>	<i>MathSAT</i>	<i>CVC5</i>	<i>Z3</i>	<i>Yices2</i>
B_000	0.117	16.275	0.256	3.866	unsat	unsat	unsat	unsat
B_001	21.795	9.977	0.167	0.166	sat	sat	sat	sat
B_002	111.489	591.434	0.088	0.034	sat	sat	sat	sat
B_003	0.156	88.571	0.412	-	unsat	unsat	unsat	-
B_004	26.423	240.545	-	-	unsat	unsat	-	-
B_005	-	283.813	-	-	-	sat	-	-
B_006	0.254	79.782	0.632	1.807	unsat	unsat	unsat	unsat
B_007	631.505	1132.074	871.195	-	unsat	unsat	unknown	-
B_008	-	-	-	-	-	-	-	-
B_009	403.500	1164.594	1192.746	-	unsat	unsat	unknown	-
B_010	-	-	-	-	-	-	-	-
B_011	-	-	-	-	-	-	-	-
B_012	19.161	373.030	2.215	-	unsat	unsat	unsat	-
B_013	1311.708	181.763	3397.059	-	sat	sat	unknown	-
B_014	-	268.974	0.284	0.133	-	sat	sat	sat
B_015	61.028	-	N.S.	N.S.	unsat	-	N.S.	N.S.

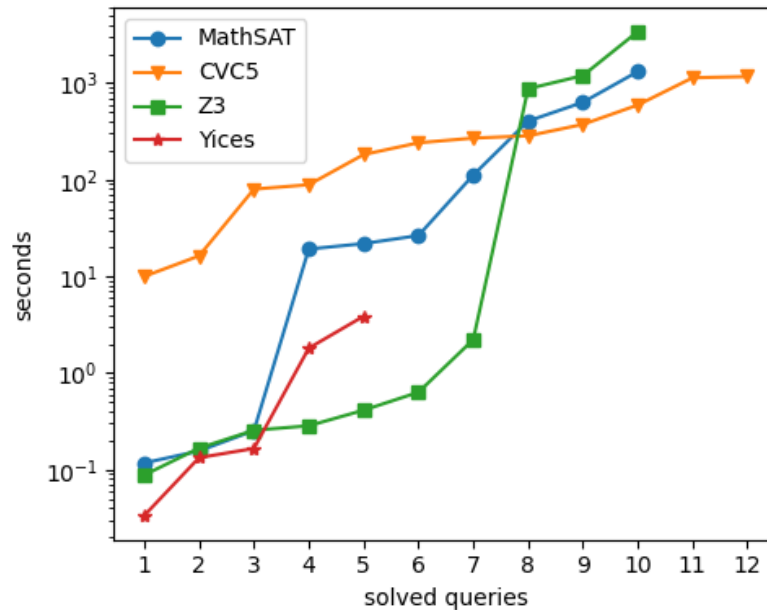


Figure 2.6. Graphical representation, on a logarithmic scale, of the times needed by the solvers to complete the queries of interest. The x-axis represents the number of solved queries, whereas the y-axis represents the time needed by the solvers to answer each query. The times for the queries which were not solved before the timeout are not reported.

From Tables 2.4 and 2.5 it appears clear that the “size” of the input variable space, controlled by *Epsilon*, is an important discriminating factor for the success of the verification process: indeed, all the queries presenting a higher value of *Epsilon* were, in general, much harder to verify for all the solvers and in many cases were impossible to verify before the timeout. It also appears clear that, as expected, the size of the network is another important factor. Furthermore, it appears that the number of layers in the network is much more important than the total number of neurons: B_006 to B_008 and B_009 to B_011, corresponding to the networks with two layers of 16 and 8 and 32 and 16 neurons respectively, seems to be harder to solve than B_012 to B_014, corresponding to the network with a single layer of 64 neurons before its output layer. Finally, it is undeniable that, at least at this time, the complexity introduced by transcendent activation functions is too high for the capabilities of the existing solvers: indeed, only in the case of the smallest input variable space (*Epsilon* ≤ 0.01) and network architecture (a single layer of 16 neurons) MathSAT was able to solve a query before the timeout.

Regarding the performances of the different solvers, CVC managed to successfully solve the greatest number of benchmarks, followed by MathSAT and Z3, with Yices managing to solve only 5 benchmarks out of the 15 whose activation functions it supported. However, it should be noted that Z3 did not manage to correctly determine the satisfiability of benchmarks B_007, B_009, and B_013 even if it appears to have finished their analysis before the timeout. All in all, from our results it seems that CVC and MathSAT are the best suited for the task of verification of neural networks. While this result is not particularly surprising for the case of CVC, as the winner of the QF_NRA division in the Single Query

Track of SMTCOMP 2022, MathSAT behaved quite better than expected, given its modest positioning in the same competition.

2.3. Use case scenario ABI_UCS2 — Test Definition

This section provides a brief description of the ABI_UCS2, and a summary of preliminary results. For details refer to deliverable D5.6.

The UCS2 corresponds to the Test Definition of this case study and involves mainly the problem of automating the process of generating sets of test cases given a particular testing goal, such as structural, functional, non-functional and state-based properties (addresses **ABI-CH1.d**).

The AIDoArt solution more directly involved in this use case scenario is UNISS_SOL_03, which aims at providing automatic test suite generation in order to check whether a reactive system complies with a set of requirements. For details, please refer to deliverable D4.1.

2.3.1. Summary of preliminary results

The first step at the base of all use case scenarios involves the requirement definition and the system modelling. Therefore, we prioritised the work related to the rules to define the use case requirement, as well as the studies to identify the best modelling language and to verify the NNs. Therefore, no preliminary results are available yet in ABI_UCS2.

2.4. Use case scenario ABI_UCS3 — Compliance Verification

This section provides a brief description of the ABI_UCS3, and a summary of preliminary results. For details refer to deliverable D5.6.

The UCS3 corresponds to the Compliance Verification of the methodological part of this case study and involves mainly the automated verification of technical requirements with respect to a reference guideline (addresses **ABI-CH1.a** and **ABI-CH1.b**). Originally, the AIDoArt solution more directly involved in this use case scenario was UNISS_SOL_05, which was aimed at providing consistency verification of technical specifications with respect to a guideline (e.g. ISO standard guidelines). This solution has been discarded, and its functionalities have been absorbed by UNISS_SOL_01, through the use of formal methods combined with NLP techniques.

Being **ABI-CH1.a** and **ABI-CH1.b** so strongly related, the work presented in ABI_UCS01 addresses both of them. In particular, the set of requirements we analysed include the Custom Requirements and the Specifications from ISO 16505:2019.

2.4.1. Summary of preliminary results

The first step at the base of all use case scenarios involves the requirement definition and the system modelling. Therefore, we prioritised the work, described in ABI_UCS1, related to the rules to define

the use case requirement, as well as the studies to identify the best modelling language and to verify the NNs. Please, refer to ABI_UCS1 for these results.

2.5. Implementation activities transversal to ABI use case scenarios

As in deliverable D5.6, this section is meant to report the activities that are transversal across the case study and cannot be described under a specific use case scenario. In particular, they involve mainly:

- the adaptivity of the system that needs to work with a variable number of cameras (addressing **ABI-CH2.a**);
- implementing video elaboration based on AI/ML techniques (addressing **ABI-CH2.b**);
- measure reliability of AI/ML in a humanly interpretable way (addressing **ABI-CH2.c**).

So far, the work carried out in this regard involved the tools INT-DEPTH, INT-DET and INT-XAI developed by Intecs. The INT-DEPTH and INT-DET solutions respond to the need of predicting new scenarios that might be considered as safety critical by identifying and giving a depth estimation of objects (vehicles, pedestrians, etc.) present in the surroundings of the vehicle. These solutions address **ABI-CH2.b** and are described in detail in deliverable D4.1. The INT-XAI solution brings the concept of explainability to the deep learning modelling world, giving an additional validation method. This solution addresses **ABI-CH2.c** and is described in detail in deliverable D3.4.

2.5.1. Summary of preliminary results

In order to develop a fast and high-performance object detector, INT-DET, a tailored version of YOLOv3 [14] was selected. This architecture falls under the single-stage object detector category in which the entire image is processed in a single pass and the class probabilities and the bounding box coordinates are learned all at once. Another key feature is that YOLOv3 divides the input image into multiple grids and makes predictions at three different scales: this is crucial in helping to recognise objects of interest (vehicles, pedestrians, etc.) even when they are far away. YOLOv3 is also optimised for real-time object detection on various hardware platforms, making it suitable for safety-critical applications such as assisting the driver.

This architecture was trained and tested on KITTI dataset [15], a widely used computer vision dataset for benchmarking and evaluating algorithms related to autonomous driving and robotics. It provides a diverse collection of sensor data collected from a moving vehicle in real-world urban traffic scenarios.

In Figure 2.7 and Figure 2.8 an overview of the performance achieved is shown. In addition to achieving high precision and recall values, it is important to note the trend of the Mean Average Precision (mAP) metric: it represents a precision and recall trade-off and is obtained by averaging the Average Precision (AP) values across all categories, calculated in turn by taking the area under the precision-recall curve, to provide an overall assessment of the model's performance.

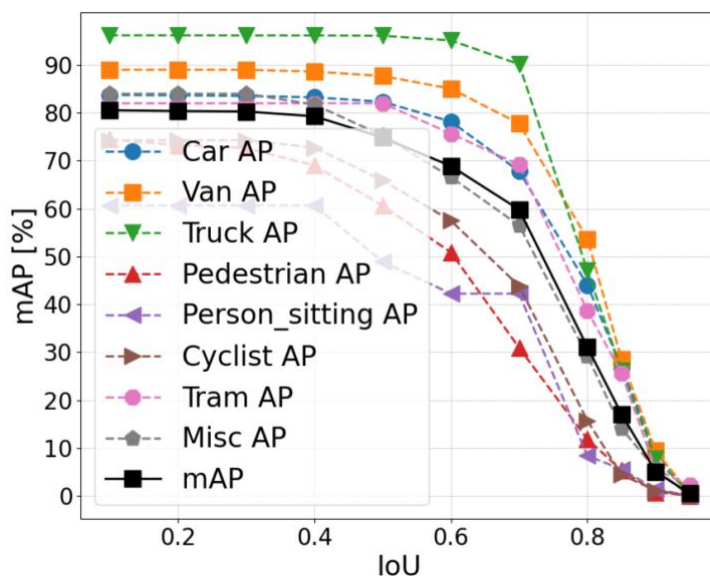


Figure 2.7. Model performance on the test set in terms of AP and mAP values; mAP@0.5 is 74.83%.

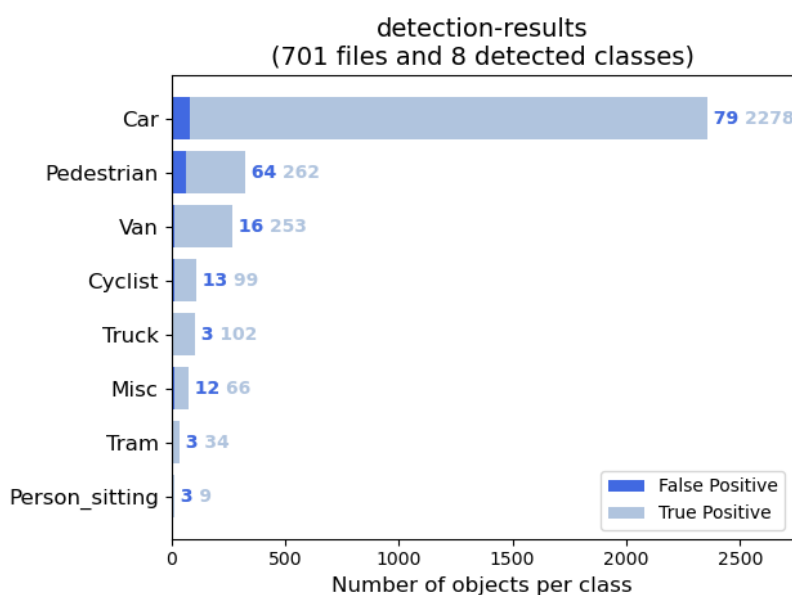


Figure 2.8. Model performance on the test set in terms of True Positive and False Positive; Precision is 94.14%, Recall is 80.81% and F1-score is 86.97%.

The INT-XAI tool is in an enhancement phase. The strand of providing the interpretation of an image containing only one object, thus solving the problem of explaining a classification model, is well-developed in literature. The context of detection models, on the other hand, are much more complex as there are potentially many objects in the scene and also belonging to many classes simultaneously. Different strategies to solve this problem are being finally compared.

The development of the INT-DEPTH tool is still ongoing. The training of the deep neural network underlying the algorithm that will provide an estimate of the depth of the objects in the scene will also

be based on the KITTI dataset [15], which provides a precise depth map as ground-truth obtained from LiDaR scans and is therefore an excellent benchmark for evaluating the final predictions.

2.6. Evaluation of results considering requirements coverage

In ABI case study, 12 requirements have been defined that are here summarised to make this chapter as comprehensive as possible:

- ABI_R01: Use automated reasoning and ML techniques for verification of specifications and high-level models.
- ABI_R02: Use automated reasoning for verification of Deep Neural Network models.
- ABI_R03: Use automated reasoning and ML techniques for generation of optimal test suites.
- ABI_R04: Use ML for video elaboration.
- ABI_R05: Use of automatic tools for compliance verification.
- ABI_R06: The system shall work with 4 cameras.
- ABI_R07: The videos shall be visualised on the car dashboard.
- ABI_R08: The system should continue operation with a reduced number of cameras (2).
- ABI_R09: The system shall provide soiling detection to recognize if the camera is dirty.
- ABI_R10: The system shall adapt when driving into/out of a tunnel.
- ABI_R11: The system shall detect relevant objects.
- ABI_R12: The system shall estimate the vehicles approaching speed.

The mapping of ABI requirements to solutions, and in turn to results, is not one-to-one. Table 2.6 is meant to give an overview of the requirements coverage, grouping them according to their relation to the ABI macro challenges described at the beginning of Section 2.1, and mapping them to the work carried out in this case study.

In particular, requirements ABI_R1, ABI_R2, ABI_R3 and ABI_R5 are directly related to ABI-CH1 and to the usage of formal verification techniques in automotive. Therefore, these requirements (that are expected to be achieved by the end of the project) are completely covered by the ongoing and future work in the context of UNISS solutions.

Requirements ABI_R4, ABI_R6, ABI_R7, ABI_R8, ABI_R9, ABI_R10, ABI_R11, ABI_R12 (that are expected to be achieved by the end of the project) concern the system functioning and the challenge defined in ABI-CH2. A subset of these requirements is directly related to implementation of the demonstration; in particular:

- ABI_R6, ABI_R7 and ABI_R8 are technical requirements necessary for the system working, and are derived directly from the customer specifications. These requirements are directly covered by ABI technical work.
- ABI_R4, ABI_R11 and ABI_R12 are related to the use of ML for video elaboration, and are completely covered by the ongoing and future work in the context of INT solutions.

- ABI_R09 and ABI_R10 are currently in standby, and the ABI technical activities to fulfil them are under evaluation.
- All these requirements related to ABI-CH2 (marked with “o” in the Table 2.6), are part of an extended set of requirements that is the input for UNISS tools .

Table 2.6 Case study requirements coverage

		UNISS				INT			ABI
		SOL_01	SOL_02	SOL_03	SOL_04	INT-DET	INT-DEPTH	INT-XAI	
ABI_R1	ABI-CH1	x			x				
ABI_R2			x						
ABI_R3				x					
ABI_R5		x			x				
ABI_R4	ABI-CH2	o			o	x	x	x	x
ABI_R6		o			o				x
ABI_R7		o			o				x
ABI_R8		o			o				x
ABI_R9		o	o	o	o				x
ABI_R10		o	o	o	o				x
ABI_R11		o	o	o	o	x		x	
ABI_R12		o	o	o	o	x	x		

2.7. Evaluation of results considering KPIs

Table 2.7 depicts the KPI specified for the ABI_CS01 case study: KPI ABI_GEN_3.1_1, which is related to the introduction of automated methods and procedures in Abinsula development processes.

Table 2.7 Case study KPI ABI_GEN_3.1_1

KPI Identifier: ABI_GEN_3.1_1		Scenario Identifier: Independent			
KPI Description:	Automation of processes improves efficiency in performing the specification and consistency verification.				
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).			
	<i>Identifier:</i>	KPI_3.1	<i>Target</i>	≥	30%
KPI Measure:	k = n/N - Number of automated procedures (n) versus total number of procedures (N).				
KPI Baseline:	<i>Source:</i>	No procedures were automated at the beginning of the project.			
	<i>Value: k₀ =</i>	0			
Target:	<i>Increase:</i>	100-k	≥	30%	

This case study has been developed from scratch in AIDOaRt, as well as the tools and solutions described in this chapter, that are still under development. Therefore, they are not mature enough yet to be integrated in Abinsula processes. Nevertheless, the work is proceeding well and first experiments with automated tools for requirement analysis and check have been done.

2.8. Planned improvements

MODELING PROPERTIES AND REQUIREMENTS (UNISS_SOL_01)

To address the difficulties presented in Section 2.2.1 “Modelling properties and requirements”, we intend to focus on the development of automatic mechanisms that can significantly assist the users in the translation of (controlled) natural language requirements to PSPs. These mechanisms will offer guidance and support throughout the translation process, making it more efficient and accessible. In particular, it would enable users with no background knowledge in formal methods and logical languages to write requirements in PSP forms and check their consistency, by providing them guidance in pattern selection, automate the mapping process, and support the analysis and validation of the resulting formal specifications.

FORMAL VERIFICATION OF NEURAL NETWORKS (UNISS_SOL_02)

We plan to expand our experimental evaluation by incorporating additional datasets regarding object detection tasks. This will allow us to explore a wider range of network architectures and move closer to the architectures needed for the ABI case study. Furthermore, we are actively researching ways to enhance existing verification methodologies based on abstract interpretation. Our aim is to extend these techniques to accommodate the neural network architectures of interest.

- **0-step vehicle detection**

As part of our ongoing efforts to enhance the experimental evaluation of our research, we are embarking on a significant expansion by introducing a preliminary step in the object detection process, which we refer to as "step zero" object detection. In this context, our primary objective is to perform image classification to determine whether an image contains a vehicle or not. This step is particularly pivotal due to the inherent complexity associated with neural networks commonly employed for object detection. These complexities often render them challenging to validate and verify using existing state-of-the-art methodologies.

The relevance of this task to ABI case study cannot be overstated, as neural networks that have undergone formal certification for robustness have the potential to serve as a highly dependable emergency detection system. Such a system would excel in identifying the presence of vehicles in close proximity to our own, thereby enhancing safety and security.

Considering that safety is paramount in the automotive domain and emergency response systems, accurate image classification plays a vital role in ensuring safety by enabling timely and precise responses to potential hazards on the road. This task contributes to reducing accidents, collisions, and potentially life-threatening situations, since a rapid and accurate detection of vehicles in proximity is essential for making informed decisions.

Our specific approach entails addressing the intricacies posed by the Vehicle Detection Image Set³. This meticulously curated dataset comprises a substantial collection of 17,760 images neatly categorised into two distinct classes: those containing vehicles and those devoid of them. Our overarching plan involves leveraging this dataset to train standard convolutional neural networks (CNNs) with the aim of achieving proficiency in image classification. Subsequently, we will rigorously assess the local robustness of these CNNs using cutting-edge verification tools and methodologies. This meticulous process is integral to ensuring that our "step zero" object detection system can reliably and accurately identify the presence or absence of vehicles in images, paving the way for more advanced object detection stages and contributing to the overall effectiveness of our research in the domain of image-based vehicle detection and emergency response systems.

GENERATION OF TEST (UNISS_SOL_03)

We plan to provide service for automatic test suite generation by using formal methods and automated reasoning since they play an important role in increasing the quality, reliability of safety-critical systems, by helping in the early detection of errors and failures that will reduce the cost and effort involved in testing.

CONSISTENCY VERIFICATION OF SYSTEM MODEL (UNISS_SOL_04)

In future work, building upon the ontological representation of system properties, we aim to conduct a series of experiments employing knowledge representation techniques and formal methods. The goal is to check the consistency of the system's model at design-time, in order to formally ensure their correctness and satisfaction by avoiding manual review which is time-consuming and error-prone. In our approach, we will address a diverse array of reasoning tasks on models, encompassing satisfiability, consistency, and axiom implication. These tasks include checking if certain conditions are possible, ensuring everything is logically sound, and seeing if certain statements can be inferred from the initial assumptions. By leveraging these techniques, we intend to unlock deeper insights into the system's capabilities and limitations, paving the way for more robust and reliable system designs in the considered domain.

2.9. Planned demonstration

Most of the solutions exploited in this use case are meant to be used by the developers, therefore the results of the collaborations with solutions providers are going to be exploited in the implementation of the Virtual Rear Mirror scenario.

The proof of concept that will be shown will consist of three main parts:

- Image acquisition: a camera that captures 1920x1080 images (HD 1080P). Originally, the data related to the camera included a up to 60 FPS stream. Due to the change of the elaboration platform (see details in deliverable D5.6) we switched to a different camera with a lower stream, but further options will be evaluated. Please, note that for demonstration purposes,

³ <https://www.kaggle.com/datasets/brsdincer/vehicle-detection-image-set>

it will be considered the possibility of simulating the acquisition from a camera using videos from a database.

- Elaboration: the INT-DET and INT-DEPTH solutions will run on the processing platform (currently studies to exploit a Xilinx Kria KV260 Vision AI Starter Kit are ongoing).
- Visualisation: currently, we are working with a 7 inch touchscreen monitor, with 1024 x 600 resolution. However, we will evaluate the adoption of different screens.

As already said, most of the solutions exploited in this use case are meant to be used by the software developers, therefore it will be considered the possibility of preparing a video to complement this PoC and present the work behind it. For INT-XAI, which is meant to work on the results of INT-DET during the development phases, it will be possible to set up a parallel PoC: INT-XAI will run on a development processing platform and will process some of the detections returned by INT-DET.

3. AVL_CS02 case study “AI supported Digital Twin Synthesis supporting secure vehicle development and testing for novel propulsion systems”

AVL subdivides its general case study into several sub case studies. In the first version of the use case evaluation report, we are focussing on the following sub case studies:

1. AVL_RDE - Real-Driver Emissions
2. AVL_TCV - Test Case Verification
3. AVL_SEC - Learning-based security testing
4. AVL_ODP - Optimization of Development Processes

3.1. Case Study description AVL_RDE - Real-Driver Emissions

Case Study Overview: The current worldwide regulation for car homologation is requiring strict limits for emissions, battery range and also battery life-time. The regulation is prescribing tests with real driving conditions. This requires car developers to perform real driving tests in different environments (simulation, Hardware-in-the-Loop (HiL), testbeds) which also make use of a driver model. The driver model needs to reproduce the behaviour of a human driver as accurately as possible. This accuracy is critical for getting comparable vehicle behaviours either with a human or simulated driver. Currently used model is a simple rule-based parametric model, whose accuracy is not fully sufficient and needs to be improved. The deviations between a human driver and a rule-based parametric model mostly come from the fact that the human driver has a more complex behaviour that is hard to model with a simple algorithm. Each driver has its own specific behaviour that influences the Real Driver Experience Key Performance Indicator (RDX KPI), electrical range, mechanical, electrical and thermal durability thermal behaviour, battery ageing, emissions and fuel consumption. The differences between different drivers should then also be captured as they are relevant factors for the test results. Therefore, a data-driven model of the driver’s behaviour is needed. A data-driven model is needed to be applicable to simulate human-like driving on any arbitrary test route. A data-driven model is also considered as a better way to pinpoint differences between different human driver behaviours.

Industrial interests: The goal of the use case is to estimate important KPIs such as emissions and energy consumption of a vehicle driving along an arbitrary route under realistic driving conditions. The problem can be split into two components:

- Estimating the velocity profile of the vehicle along the track.
- Computing the energy consumption/ emissions from velocity and environmental factors.

As a simulation expert in the automotive domain, AVL is highly proficient in the latter, leaving only the former challenge to handle.

Table 3.1 Synopsis of the case study AVL_RDE

Use case scenario AVL_RDE_UCS1 — Vehicle dynamics isolated speed profile data from real world measurements	
Description:	We are developing a method for the generation of a speed driver profile for an arbitrary selected driving route by the end-user. Therefore, a fidelity of the automatically generated speed profiles of a driver is needed.
Requirements:	AVL_RDE_R01, AVL_RDE_R02
Tools:	AALpy (TUG)
KPI:	AVL_RDE_UCS1_KPI_3.1_1
Use case scenario AVL_RDE_UCS2 — Basic driver behaviour identification	
Description:	Generation of a driver behaviour model based on the environment events (i.e. traffic signs, traffic signalization, traffic conditions, ...). Thus, the method should automate the generation of driver behaviour.
Requirements:	AVL_RDE_R05, AVL_RDE_R02
Tools:	AALpy (TUG)
KPI:	AVL_RDE_UCS2_KPI_3.1_1
Use case scenario AVL_RDE_UCS3 — Modelling driver behaviour pattern	
Description:	Modelling of driver's behaviour patterns in constant speed limit motorway area in terms of speed profile is needed. The model should increase the fidelity of the automatically generated speed profiles of a driver on a motorway.
Requirements:	AVL_RDE_R03
Tools:	AALpy (TUG)
KPI:	AVL_RDE_UCS3_KPI_3.2_1
Use case scenario AVL_RDE_UCS4 — Worst Case Emissions Scenario	
Description:	Generated driver behaviour speed profile and driver behaviour model will be used to calculate the worst-case emission and compared with the worst-case real engine testbed experiments as currently done at the AVL (i.e. without using the AI/ML methodology). The model needs to show an increase in quality and accuracy of the critical emission estimation generated by the engine testbed experiments.
Requirements:	AVL_RDE_R04
Tools:	AALpy (TUG)
KPI:	AVL_RDE_UCS4_KPI_3.2_1

Table 3.1 is summarising all discussed and potential use case scenarios for the AVL_RDE use case. In this section, we will report about AVL_RDE_UCS3, i.e. modelling human driver behaviour on the highway. In the demonstrator dedicated to the AVL_RDE_UCS3, we will show a possible driver profile for an arbitrary selected driving route at the highway. The integrated traffic conditions, we will select from historical data provided by HERE Maps⁴.

⁴ <https://www.here.com/>

3.1.1. Use case scenario AVL_RDE_UCS3 — Vehicle dynamics isolated speed profile data from real world measurements

We are developing a method for the generation of a speed driver profile for an arbitrary selected driving route by the end-user. Therefore, a fidelity of the automatically generated speed profiles of a driver is needed.

Challenges

Modelling the velocity profile of a human driver is not a trivial task. The very fact that the task is not of a discriminative nature due to hidden variables (e.g. traffic conditions, exact weather conditions, traffic signalization, ...) makes this generative ML problem challenging. Moreover, there are a number of dataset challenges that must be taken into account when modelling a vehicle's speed profile. Namely, recording only one driving cycle is very expensive, time-consuming, and repeatable only to a limited extent. Therefore, data is sparse and very often contains missing information.

To model a signal similar to a velocity profile of human driving along an arbitrarily selected route and under different traffic conditions, the following challenges are to be expected:

- Limited data quantity due to the cost of performing test drives.
- Limited data quality: not all relevant data features are recorded during test drives.
- Trade-offs between expressive power and generalisation capabilities of the learned models.
- Assessing model quality, which previously required an expert.

Tools

The driving recordings (i.e. test drives) from different routes are stored in the Smart Mobile Solutions⁵ (SMS) tool developed by AVL. Additionally to the real driver recordings, this tool can generate GPS tracks augmented by static environment data and even some statistical dynamic data (such as average velocity of traffic participants for a given day). The plan is to improve the existing velocity profile generation methods of SMS using tools based on AALpy, the automata learning library of TUG.

3.1.1.1. Summary of preliminary results

TUG developed a prototype for learning probabilistic behavioural models of human driver behaviour from a set of recorded driving cycles (i.e. test drives) enriched by static environment data such as the current speed limit and curvature. The tool can also use a learned model to artificially generate new behaviour data for new routes (sampling). An example of such generated behaviours for a given route is shown in Figure 3.1 along with real measurement data for comparison.

⁵ <https://www.avl.com/it/-/avl-smart-mobile-solutions>

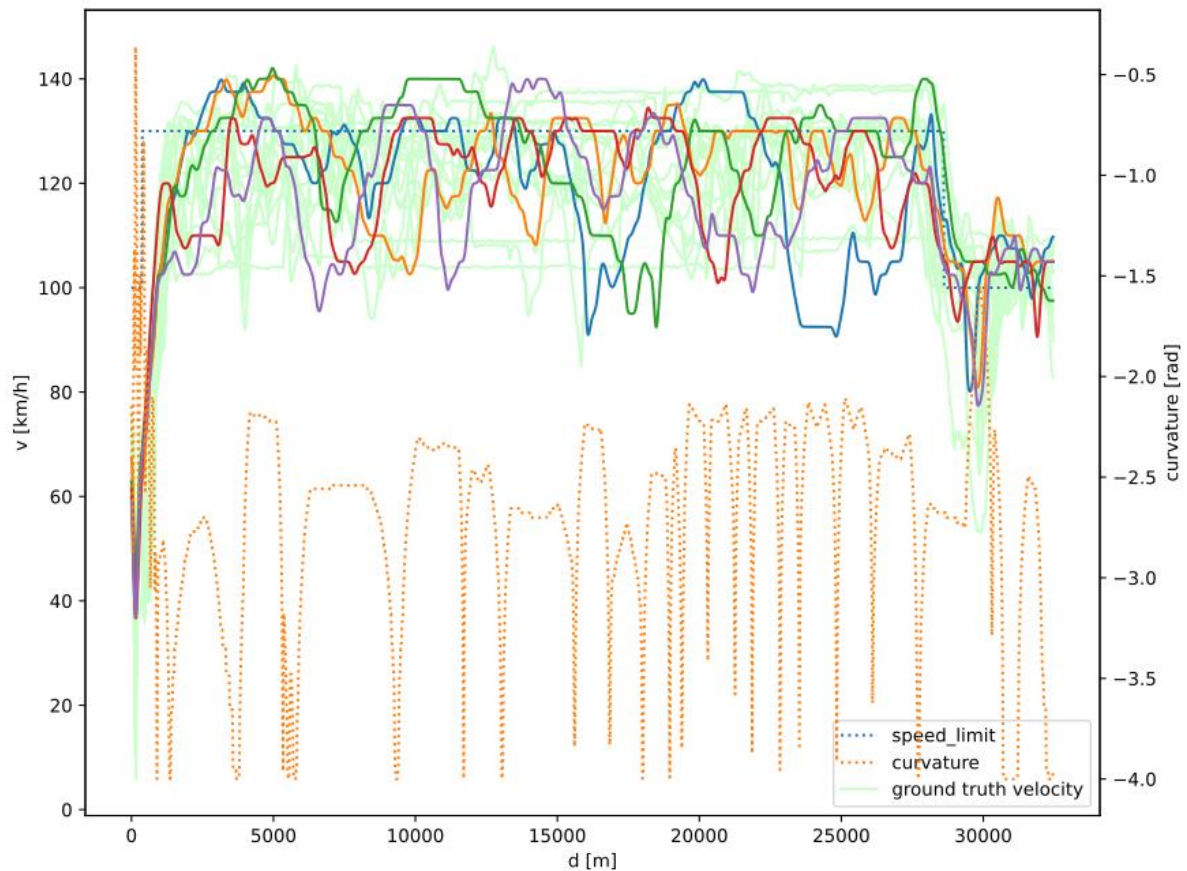


Figure 3.1 Measured behaviour (light green), behaviour generated from a learned model (solid lines) and road features (dashed) for a given route

The tool models driver behaviour as a probabilistic agent acting on known environment stimuli. When generating new velocity data, this model interacts with a track model that is generated from augmented GPS data of the target route. The outputs of the driver model serve as inputs for the environment and vice versa.

The driver behaviour is modelled as a Markov Decision Process (MDP), which can be defined by a set of states that emit output symbols when active and probabilistic state transitions between states which are triggered by inputs. Further, the sets of inputs, outputs and states are finite. For this reason, it is necessary to have a discrete representation of the continuous real-world data, as well as a mapping between those two spaces. This mapping can be seen as an abstraction method for the real-world data. This methodology is illustrated in Figure 3.2.

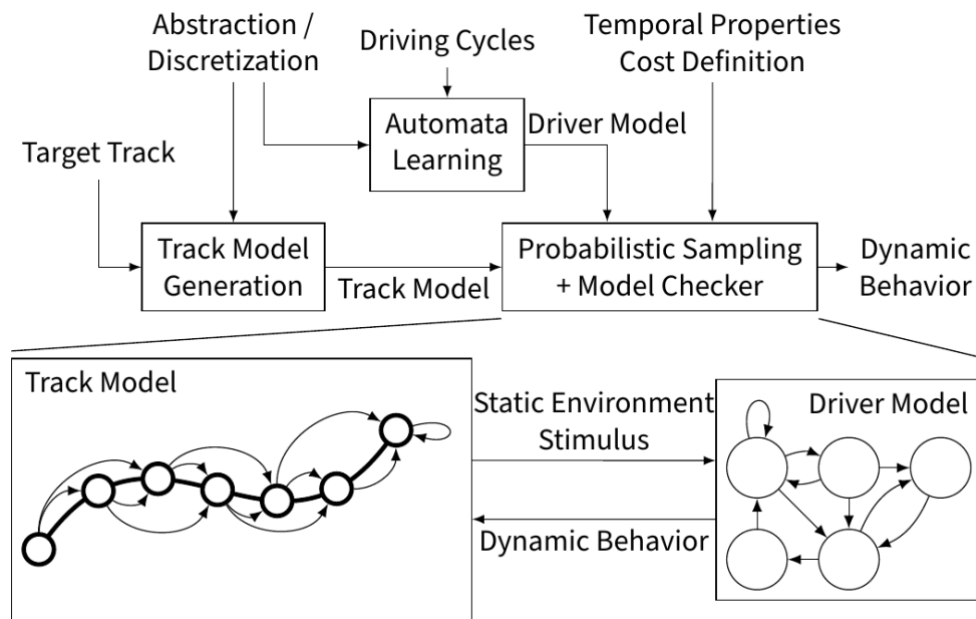


Figure 3.2 Methodology overview

A main focus was exploring the space of suitable abstractions which involves the selection of relevant data fields and number of discretization steps for those fields. Shifted versions of the data can also be important features since driving behaviour strongly depends on environment features from future positions. This also requires cleaning and preprocessing the provided data, for example to resample the data which is given as time series to use a fixed distance step instead.

The approach developed by TUG uses domain knowledge to aid the learning process. This is done by providing a segmentation criterion which is used to split each input trace into labelled segments. The labels correspond to different high-level actions such as accelerating or braking. Instead of learning a behavioural model of the whole behaviour directly, the approach learns models for each of the actions and how they interact with each other. Those models are then combined to get a full driver model.

The model used to produce the samples shown in Figure 3.1 was trained on 21 test drives on three different routes from which we obtained 39 different highway segments with a total of 314 traces with a total length of about 1150 kilometres. The inputs are the road type, future and previous speed limit and curvature and produces velocity profiles with a resolution of 2.5 km/h and a spatial resolution of 5 metres. It uses two high-level actions for acceleration and deceleration and has about 100 states in total.

3.1.1.2. Evaluation of results considering requirements coverage

This use case scenario relates to AVL_RDE_R03⁶. The extraction of traffic conditions as described in AVL_RDE_R02⁷ has not been addressed so far. The methodology developed by TUG partly addresses AVL_RDE_R01⁸ as it provides a model that can be used to provide human-like driving given an arbitrary target route, however only focussing on the highway. We concentrate only on the highway as it is the most challenging part of the route for the current available AVL solution. Since the extraction of traffic information from the training data is currently not available, the resulting models do not accept corresponding inputs. However, this should not be an issue once AVL_RDE_R02 is addressed. The approach also does not model the influence of vehicle characteristics on the velocity profile. This was not prioritised since the influence of vehicle characteristics on driving behaviour is estimated to be small compared to other influences such as driving environment and driving style.

3.1.1.3. Evaluation of results considering KPIs

AVL_RDE_UCS3_KPI_3.2_1

Table 3.2 Case study KPI “AVL_RDE_UCS3_KPI_3.2_1”

KPI Identifier: AVL_RDE_UCS3_KPI_3.2_1		Scenario Identifier: AVL_RDE_UCS3	
KPI Description:	Increase the coverage and quality of the generated measurements of drivers behaviour.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase the coverage and quality of actionable feedback for the next DevOps iteration.	
	<i>Identifier:</i>	KPI_3.2	<i>Target</i> ≥ 25%
KPI Measure:	k = the generated driver profile can be used for different driving routes		
KPI Baseline:	<i>Source:</i>	AVL SMS Route studio profile of a human driver for different routes	
	<i>Value: k₀ =</i>	Probability of the speed profile generated by the current SMS Route studio solution to belong to the distribution of the real driver speed profile.	
Target:	Increase:	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 25%

⁶ AVL_RDE_R03: Automated multi-source data analysis of the real driving test data such that the relevant features of the driver behaviour can be clustered (e.g. highway driving, low speed driving, cornering, braking, acceleration,...). To be used for understanding the driving conditions.

⁷ AVL_RDE_R02: AI method that will provide better statistics of the environment based on the statistics of the real driving recording and data from digital map service

⁸ AVL_RDE_R01: Based on the real driving recordings (time based data on vehicle speed, throttle/brake pedals, curvature, road gradient, GPS coordinates...) the ML model is trained to simulate human-like driving given a target route, vehicle and traffic conditions. During the training of the ML model vehicle characteristics are known. Traffic conditions have to be extracted from the recorded data based on the speed profile. Additionally, traffic conditions can be estimated based on traffic data provided by AVL partners (digital map service). Thus, driver behaviour in constant speed limit areas can be simulated by augmenting the AI based model on top of a dynamics simulator.

AVL experts have evaluated the results of the developed method, and this visual inspection has shown the following:

1. The driving profile generated by the proposed method significantly better models the driver speed profile on the motorway compared to the method currently in use by the SMS Ruoth Studio.
2. The general driving trend generated by the proposed method is hard to distinguish from the general driver trend of a human driving on the motorway.
3. Due to the granularity of the automata approach, the local speed profile generated by the proposed method does not show the variations seen in the speed profile of a human driver.

3.1.2. Planned improvements

We plan to further explore different possibilities regarding the abstraction of the data in close collaboration with domain experts to improve the quality of the learned driver models.

Currently, the approach developed by TUG for learning driver models is applied only to highway driving. In future work, this approach could be extended to driving behaviour in general. The main lines of work in this direction are:

- Extending the concept of high-level actions to a multi-level hierarchy. An example for this would be a two-level hierarchy with rural driving vs high-way driving at the top level and acceleration and deceleration actions on the second level.
- Combining our approach with ML algorithms such as imitation learning.

We also plan to investigate methods that use domain knowledge to cope with different traffic conditions. This could be done by influencing the process of sampling new traces from a learned driver model such that the samples adhere to properties that are characteristic of certain traffic conditions.

3.1.3. Planned demonstration

At the plenary meeting in December, we will present a AVL_RDE_UCS3 demonstrator capable of generating a speed driver profile for an arbitrary selected driving route on a highway by the end-user. We will demonstrate that generated speed profiles closely correspond to human driving profiles. We will show that the generated driving profiles differ among each other, however, each corresponds to possible human driving behaviour. We will show how changing the traffic condition affects the generated speed profile on the selected route.

3.2. Case Study description AVL_TCV - Test Case Verification

Overview: The AVL SCENIUS Test Case Generator⁹ is an easy means to generate Advance Driver-Assistance Systems (ADAS) test cases from abstract scenarios. By enhancing one given scenario - like an overtake manoeuvre on a highway - by concrete values for vehicle speeds, distance, etc., thousands of test cases can be generated. Since HIL or Vehicle test execution is very expensive, it is crucial to select a subset of tests, sufficient to cover all critical situations. SCENIUS provides means to perform such a selection. However, we need to be able to test whether the selection performed by SCENIUS covers all critical cases while excluding non-critical cases.

Industrial interests: The validator must determine, with a given accuracy, if a given test case selection is adequate without itself requiring executions of all possible test cases. Namely, the number of possible scenarios and the range of possible initial conditions for each scenario is growing exponentially, so it cannot be exhaustively estimated. The test generator developed at AVL that needs to be validated is ML based. Furthermore, the verdict given by the validator must be understandable/explainable by humans. Finally, the validator should provide the parameter values that lead to critical situations if they are not covered by the test generator.

Table 3.3 Synopsis of the case study AVL_TCV

Use case scenario AVL_TCV_UCS1 — SCENIUS Test Case Selection Validator	
Description:	We are developing a methodology that can determine whether the case sampled by SCENIUS is in the physically meaningful range and provide the SCENIUS user an explanation of the decision. By analysing instances of a logical scenario, an analytical/statistical range of parameters limited to physically meaningful ones is compared to the sampled value of the parameters by SCENIUS. The probability/possibility of sampled values being in the physically meaningful range is reported to the user.
Requirements:	AVL_TCV_R01, AVL_TCV_R02
Tools:	DTsynth (AIT)
KPI:	AVL_TCV_UCS1_KPI_3.1_1
Use case scenario AVL_TCV_UCS2 — SCENIUS parameter recommender	
Description:	We are developing methodology that can provide the SCENIUS user with a set of parameters that lead to yet uncovered critical solutions. By analysing instances of a logical scenario, we are developing an analytical solution that limits the range of parameters to physically meaningful ones. The uncovered critical solution is then determined by a novel sampling strategy of the parameters values inside determined parameters range.
Requirements:	AVL_TCV_R03
Tools:	DTsynt (AIT)
KPI:	AVL_TCV_UCS2_KPI_3.1_1

⁹ <https://www.avl.com/-/scenius>

Table 3.3 is summarising all discussed and potential use case scenarios for the AVL_TCV use case. In this section, we will report about both use case scenarios.

3.2.1. Use case scenario AVL_TCV_UCS1 — SCENIUS Test Case Selection Validator

We are developing a methodology that can determine whether the case sampled by SCENIUS is in the physically meaningful range and provide the SCENIUS user an explanation of the decision.

By analysing instances of a logical scenario, an analytical/statistical range of parameters limited to physically meaningful ones is compared to the sampled value of the parameters by SCENIUS. The probability/possibility of sampled values being in the physically meaningful range is reported to the user.

The purpose is to develop a set of tools and rules that would allow generating critical scenario parameters for autonomous vehicle testing. Concretely, autonomous vehicle functions are tested in simulation, in virtual traffic scenarios, which are parametrized by dozens of variables. The objective is to develop a sound methodology to reduce the parameter space volume significantly, to do more time efficient testing. This is done by first identifying where and when the critical situations could occur within the simulation, and then using dynamics equations to discard the variables that would not result in the actors being at the critical time and space.

Challenges

The main challenge is balancing efficiency by reducing the parameter space, with certainty that the discarded regions of the parameter space could not have led to a critical or interesting scenario. The biggest problem is that the behaviour of the autonomous vehicle function is not known a priori and could even be seen as a black box, so the developed methodology needs to be able to work for all functions. Next, simulation capabilities are limited in comparison to the space that needs to be exploited while testing AD vehicles. Thus, an intelligent selection of testing samples is necessary for reliable testing.

Tools

The simulation environment used consists of OpenScenario 1.1 scenario files, which are played on the esMINI open-source simulator. The SCENIUS tool from AVL supports the creation of scenario files, while AIT is developing a Python interface to connect to AVL's toolchain for the actual parameter reduction.

Approach: The proposed solution involves physical constraints of all actors involved in the scenario in order to be certain that no critical scenarios are accidentally discarded. The input given is an OpenScenario 1.1 file, which is a traffic scenario description file, that serves as a template for the individual test cases. The scenario file is used to extract all the relevant high-level information: the number of vehicles and Vulnerable Road Users (VRUs), their behaviour (including their trajectories), and the trajectory that the Vehicle Under Test (VUT) will undergo.

Given the desired trajectories of all actors (including the VUT), critical regions in the map are identified by calculating the physical behaviour of scenario participants. For safety-related ADAS functionalities (e.g., Automatic Emergency Braking), a critical region would be a part of the map where the vehicle could collide with another vehicle or a VRU. Since the behaviour of all other participants is known and controlled by the simulator, it is possible to calculate a set of critical times a priori, when other actors are within the identified critical areas. We also utilise the notion of Post Encroachment Time (PET) denoting the time span when the last VRU leaves the critical area until the VUT enters this critical area. Thus, PET defines a measure of criticality and can be used to identify initial values that lead to a low PET and, therefore, to a critical situation.

After identifying all critical areas and times, the Operational Design Domain (ODD) constraints are used to calculate parameter ranges which would lead to critical scenarios between the VUT and other participants. This approach discards all scenarios which are outside of the ODD of the ADAS/AD functionality, which are irrelevant for testing.

The output will be an intelligent sampling strategy which outputs OpenScenario files fed to the simulator. For coverage purposes, a uniform sampling strategy will be employed to explore every region of the scenario parameter space. Additionally, active sampling strategies that receive feedback on the performance of the ADAS function after each simulation, will be employed for optimization purposes trying to find the set of parameters which is most critical. This information is then fed back to the teams developing the ADAS functions, or used to create more sophisticated HIL testing.

3.2.1.1. Summary of preliminary results

The above approach has been used on a family of scenarios meant to test an Automatic Emergency Braking ADAS function. The scenarios considered typically involve the VUT driving, and then a pedestrian crossing the road. As a first step, the solution provided already reduces the parameter space of the scenario to just 25% of what was originally considered (i.e., 4x improvement).

However, as is natural during any solution development, further improvements have been identified and are underway. Crucially, if the parameter reduction is done considering the full ODD, then there are still many sampled scenarios that do not test the ADAS function in a meaningful way. We started to investigate the PET as a measure of meaningfulness of possible parameter sets. Based on the PET and the given trajectories of road users and the VUT we were able to come up with a closed form solution for the initial position of the VUT. Thus, effectively reducing the parameter space based on the “criticality” of the situation at hand for a single scenario.

The following Figure 3.3 depicts the result of inverting the mathematical model if we impose, i.e., a PET between 0s and 1s. The goal was to derive the starting positions of the VUT which lead to a certain PET range.

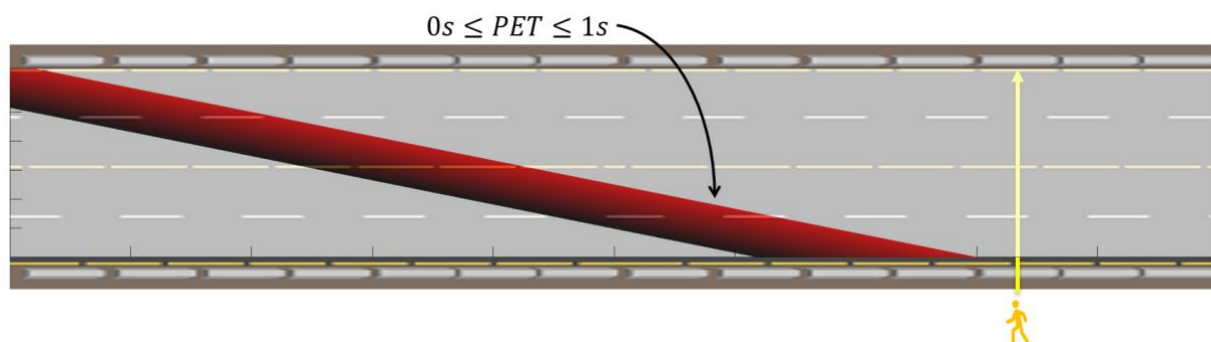


Figure 3.3 Straight street scenario, where a pedestrian is crossing the street.

In this use case scenario we are able to decrease the parameter space before any sampling by at least 70% but still fulfilling the requirement of the PET being in the range [0, 1].

3.2.1.2. Evaluation of results considering requirements coverage

We will evaluate the analytical approach by comparing the number of actually sampled scenarios using the different sampling strategies without out pre-screening the parameter space that fulfil specific criticality measures, i.e. PET in the range of [0, 1] seconds, to the same sampling strategy but with the analytical parameter space reduction. This evaluation addresses AVL_TCV_R01¹⁰ and AVL_TCV_R02¹¹. We introduce the Average Requirement Coverage Score (ARCS) which is then the average of the aforementioned relation over N samples. ARCS is directly related to AVL_TVC_UCS_KPI_3.1_1 but also taking into account that even when the parameter space is analytically reduced according to the requirements, physical parameters, i.e., acceleration, are random variables.

Recently, we started looking into the possibility to assess the criticality of concrete scenarios a-priori, based on the specific parameter values. We will explain this technique by example of the pedestrian-crossing scenario which is used to test the obstacle avoidance capabilities of automated vehicles: in this scenario, the Ego vehicle approaches a point on the road with a defined speed and initial distance. At the same time, a pedestrian approaches the same point from a lateral direction with their own speed and initial distance. If these speed and distance values are specifically tuned, they will result in a collision between the Ego vehicle and the pedestrian. Otherwise, however, the two traffic

¹⁰ AVL_TCV_R01: The SCENIUS Test Case Selection Validator must determine with a given accuracy, if a given test case selection is adequate.

¹¹ AVL_TCV_R02: The verdict given by the SCENIUS Test Case Selection Validator must be understandable/explainable by humans

participants might never come into close proximity to each other. Under the assumption of linear motion of both the Ego vehicle and the pedestrian, we create closed-form solutions of different KPIs like Post-Encroachment Time or the Distance of Closest Approach (i.e., the closest distance at which the Ego vehicle and the pedestrian pass each other by), and we evaluate them based on their capability to identify concrete scenarios of low criticality and therefore low relevance.

3.2.1.3. Evaluation of results considering KPIs

AVL_TCV_UCS1_KPI_3.1_1

Table 3.4 Case study KPI “AVL_TCV_UCS1_KPI_3.1_1”

KPI Identifier: AVL_TCV_UCS1_KPI_3.1_1		Scenario Identifier: AVL_TCV_UCS1	
KPI Description:	Reduction of the no-critical test cases using automated examination processes and while providing explainability.		
Refined AIDoArt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual.	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = the number of not critical test cases is reduced based on the pre-calculation of the physically meaningful restriction.		
KPI Baseline:	<i>Source:</i>	AVL SCENIUS sampling strategy	
	<i>Value: k₀ =</i>	the number of noncritical test cases generated by the uniform sampling strategy utilised in the AVL SCENIUS. <i>Not yet calculated.</i>	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 30%

We are currently in the process of validating the a-priori scenario criticality estimation against previously generated “full-factorial” test suites which cover the full parameter space. The individual concrete scenarios of this test set were created by SCENIUS and the simulations were performed by Model.CONNECT and esMINI, using the CARLA autonomous driving stack.

Regarding AVL_TCV_UCS1, once our a-priori estimation has been shown to provide reliable results, we can use it to validate the quality of different test case generation methods. In a further step, we aim to integrate it into our Active DoE method to save on simulation time by pre-filtering irrelevant concrete scenarios. Nevertheless, our preliminary results show that around 30% of generated test cases can be recognised as not critical.

3.2.2. Use case scenario AVL_TCV_UCS2 — SCENIUS parameter recommender

We are developing a methodology that can provide the SCENIUS user with a set of parameters that lead to yet uncovered critical solutions.

By analysing instances of a logical scenario, we are developing an analytical solution that limits the range of parameters to physically meaningful ones. The uncovered critical solution is then determined by a novel sampling strategy of the parameters values inside determined parameters range.

3.2.2.1. Summary of preliminary results

We present a novel generative machine learning approach for the automatic generation of critical test cases within a given traffic scenario. This involves producing parameter values that initialise the traffic scenario for input into a simulator. A test case is deemed critical when the resulting simulation involves high-risk situations, such as the ego vehicle approaching dangerously close to a pedestrian.

Preliminary results show that our Generative based AI approach is capable of generating test samples that are only in the physically constrained range of values.

3.2.2.2. Evaluation of results considering requirements coverage

We evaluated the approach introduced in the previous section that addresses AVL_TCV_R03¹². The evaluation is directly related to AVL_TVC_UCS_KPI_3.1_1, on a specific traffic scenario where the Ego vehicle approached a crosswalk with a pedestrian crossing the street (refer to Figure 3.4). The Ego vehicle moved parallel to the roadside at a constant distance h and velocity v_e , starting l metres away

¹² AVL_TCV_R03: The new parameter values given by the SCENIUS parameter recommender must lead to critical situations which are not covered by the generated Tests from the SCENIUS test case generator.

from the crosswalk. Simultaneously, the pedestrian moved from the roadside with a constant velocity v_p . The parameters l , h , v_e , and v_p defined this scenario (denoted as $X = \{ l, h, v_e, v_p \}$).

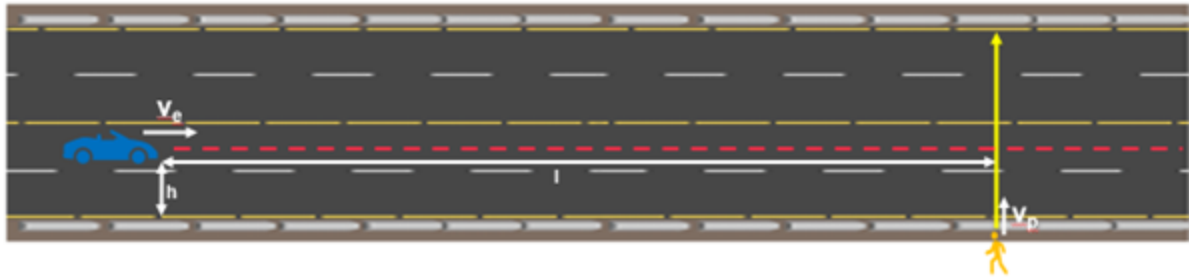


Figure 3.4 - Pedestrian Traffic Scenario

A critical simulation occurred if the Ego vehicle eventually entered a predefined pedestrian safety area, defined as a rectangle with specific coordinates in the Cartesian plane (refer to Figure 3.5).

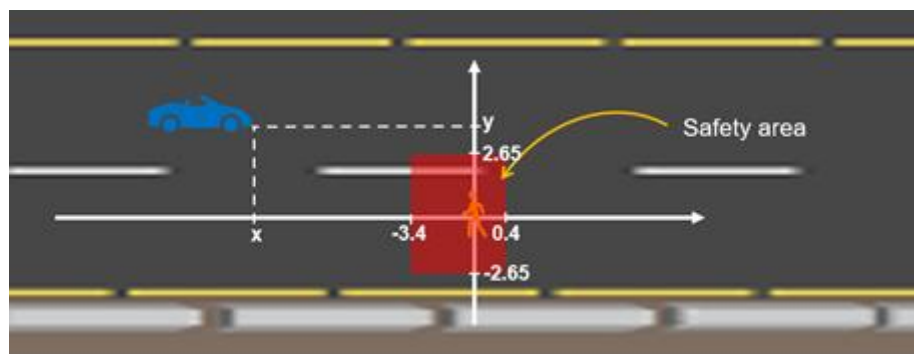


Figure 3.5 - Cartesian Plane & Safety Area

The validity domains for the parameters were defined as follows:

- i. $l \in [3.4, 100]$ m
- ii. $h \in [1, 5]$ m
- iii. $v_e \in [20, 60]$ km/h
- iv. $v_p \in [1, 10]$ km/h

To assess criticality, the Ego vehicle's position (x,y) in relation to the pedestrian's safety area was crucial. A test case was considered critical if, at some point T during the simulation, the Ego vehicle was in the pedestrian's safety area, i.e.:

- a) $x \in [-3.4, 0.4]$ m, and $y \in [-2.65, 2.65]$ m.

If the Ego vehicle was within the safety area at time T , it must have entered the area at some earlier time T' ($T' < T$). Since the Ego vehicle moved from left to right and the pedestrian from bottom to top, there were two possible scenarios for T' :

- (1) $x = -3.4$ m and $y \in [-2.65, 2.65]$ m, or
- (2) $x \in [-3.4, 0.4]$ m, and $y = 2.65$ m.

Thus, we simplified the criticality constraints to either scenario (1) or (2). To handle this, two test case generators could be trained: one for scenario (1) and one for scenario (2).

The test case generator for scenario (1) was trained to sample 100,000 data points whose values are constrained by the physic meaningful ones defined above.

The System Under Test (SUT) used the motion equations of the Ego and pedestrian to compute the Ego vehicle's y -coordinate when its x -coordinate became -3.4 metres. If the resulting y -coordinate fell within the interval $[-2.65, 2.65]$ metres, the simulation was considered critical; otherwise, it was not. Thus, the SUT outputted the (x, y) -coordinates of the Ego vehicle in the Cartesian plane with the pedestrian as origin, where x was fixed and equal to -3.4 m and y was computed as:

$$y = h - (l - 3.4) \cdot v_p / v_e .$$

To optimise this scenario, the loss function was defined considering the constraints i. – iv. on l , h , v_e , v_p , and the simplified criticality scenario (1) with the constraint on y . The loss function consisted of mean squared errors calculated based on these constraints.

3.2.2.3. Evaluation of results considering KPIs

AVL_TCV_UCS2_KPI_3.1_1

Table 3.5 Case study KPI “AVL_TCV_UCS2_KPI_3.1_1”

KPI Identifier: AVL_TCV_UCS2_KPI_3.1_1		Scenario Identifier: AVL_TCV_UCS2	
KPI Description:	Reduction of the number of manually evaluated test cases by automating the processes of identifying whether the selected test cases are critical or non-critical.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual.	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> \geq 30%
KPI Measure:	k = The percentage of test cases that should be manually evaluated to determine whether the cases are critical or not.		
KPI Baseline:	<i>Source:</i>	The test case evaluation done in the AVL SCENIUS.	
	<i>Value: k_0 =</i>	100% (i.e. in the current strategy each test case has to be evaluated)	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	\geq 20%

We used the generator trained as described in the previous section to generate 1000 test cases for the pedestrian traffic scenario. Figure 3.6 illustrates these generated test cases, indicating their corresponding y-coordinates and time values when the x-coordinate equals -3.4 m. The visualisation utilises the parallel coordinates technique, enabling a high-dimensional representation where each y-axis corresponds to a specific dimension. Each test case appears as a set of connected lines, depicting its values on the axes. The top and bottom lines serve as boundary markers for variable validity and are not indicative of generated test cases.

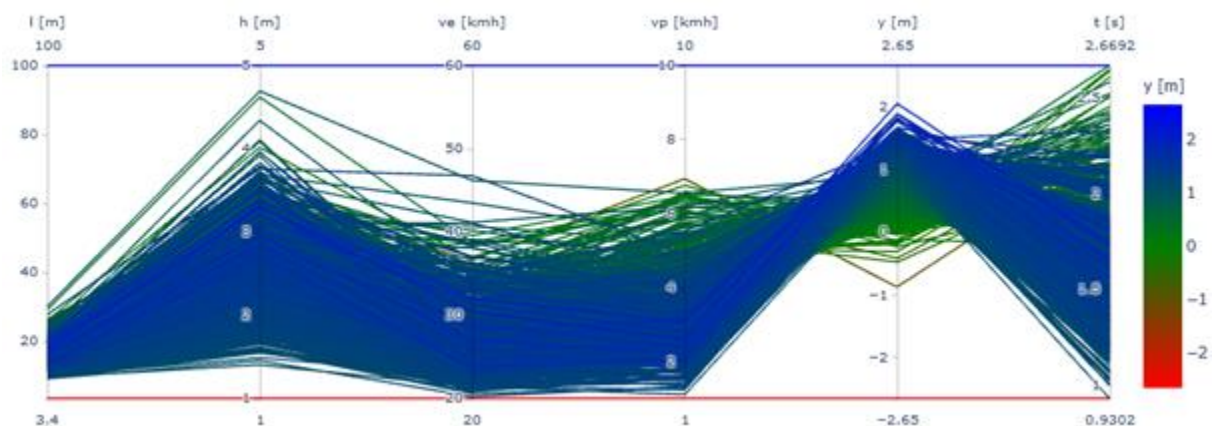


Figure 3.6 - Generated Test Cases

Figure 3.6 illustrates that all generated test cases fell within the defined validity ranges (i. – iv.) outlined in the previous section. Additionally, all simulations were critical, leading to y-values within the pedestrian’s safety area.

However, it is evident that the generated test cases did not cover the complete validity ranges of all scenario initialization variables. Among these variables, the lateral distance (h) of the Ego vehicle from the roadside, and to some extent, the velocities (v_e and v_p) of the Ego and pedestrian, showed better coverage within their validity ranges. On the other hand, the initial distance (l) of the Ego vehicle from the crosswalk had relatively poor coverage. This limitation can be attributed partially to the quality of the test case generator and potentially to the absence of solutions in certain regions of the input search space. This shortfall might arise due to the validity range limits being set without considering dynamic relationships between the variables.

To enhance the quality of the trained test case generator concerning validity range coverage, it might be beneficial to encourage the generator during training to produce output values that not only fall within the defined ranges but also exhibit maximum variability, if necessary.

3.2.3. Planned improvements

Both approaches, however, are highly specific to the individual logical scenarios and a KPI that works for one scenario might not be applicable to the other. To illustrate this, let's examine the Target-Deceleration scenario which is used to validate distance-keeping: here, a Target vehicle travels in front of the Ego vehicle. At some point, the Target vehicle starts decelerating for a certain amount of time, reducing its speed to a defined final speed. Assuming linear motion for both cars, the Distance of Closest Approach KPI is not applicable in the same form as before, but splits into three distinct cases: if the speed of the Ego vehicle is lower than the final speed of the Target vehicle, both vehicles will drift apart over time. If the speed of the Ego vehicle is equal to the final speed of the Target vehicle, they will maintain a fixed distance. If the speed of the Ego vehicle is higher than the final speed of the Target vehicle, both vehicles would eventually crash if linear motion was maintained. Only in this third case, the Ego vehicle would have to intervene in order to prevent a crash.

Generalisation of the method is planned for future research.

3.2.4. Planned demonstration

At the Dec 23 AIDOaRt plenary meeting, we will present a demonstrator capable of removing non-critical test cases (AVL_TCV_UCS1), as well as a demonstrator capable of generating only test cases that may lead to a critical situation within physical constraints (AVL_TCV_UCS2).

3.3. Case Study description AVL_SEC - Learning-Based Security Testing

Overview: With the increasing importance of software and the trend of connected vehicles's security issues become more and more in the focus of vehicle tests. In AIDOaRt, therefore, AI-supported methods for vehicle security tests should be developed for both vehicle life time phases development and usage.

Industrial interests: A vehicle consists of many heterogeneous components that communicate with each other via communication protocols, which can exhibit a sizable amount of vulnerabilities. In practice, models prove to be an efficient tool for stateful testing. Creating these models manually and keeping them up to date is tedious. To overcome this challenge, automata learning techniques should be used to automatically infer such behavioural models. However, applying these techniques to learn real communication protocol implementations requires an interface that ensures reliable testing. Learning wireless protocols is complicated by the fact that the connection is not reliable, e.g. due to packet loss or delayed transmissions. Later, we will also explore wired communication protocols, e.g. via the CAN interface, as well as specialised V2X (Vehicle-to-X) communication protocols. The goal is to streamline the learner in a manner that one learning framework can use different adapters and therefore the same interface can be used to learn a multitude of different protocols.

Another challenge is to develop model-based security testing techniques that can explore unexpected behaviour but also verify the absence of security issues. This also includes the challenge of creating a

sufficient security specification to successfully apply model checking. To allow for efficient zero-input testing, we aim for using the learnt model to guide a model-based fuzzing framework.

Table 3.6 Synopsis of the case study AVL_SEC

Use case scenario AVL_SEC_UCS1 — Learning-based model checking for security testing	
Description:	We develop an automata learning (and poss. ANN-plausibility checked)-test case generator that uses fuzzing and model checking to use traces of specification violations.
Requirements:	AVL_SEC_R01, AVL_SEC_R02, AVL_SEC_R03, AVL_SEC_R04
Tools:	AALpy (TUG)
KPI:	AVL_SEC_USC1_KPI_1.1_1, AVL_SEC_USC1_KPI_1.2_1
Use case scenario AVL_SEC_UCS2 — Machine-learning based anomaly detection for automotive systems	
Description:	We develop a feedbacked fuzzer based on an anomaly detection-enhanced test oracle. We train an ANN to learn the normal behaviour of a CAN network and try to find anomalies that occur when fuzz-testing the same.
Requirements:	AVL_SEC_R05, AVL_SEC_R06, AVL_SEC_R07
Tools:	DT_synth (AIT)
KPI:	AVL_SEC_USC2_KPI_3.2_1, AVL_SEC_USC2_KPI_4.2_1
Use case scenario AVL_SEC_UCS3 — Machine-learning based anomaly detection for automotive systems	
Description:	We develop a remote, live-data fuzzer based on AVL_Sec_UCS2.
Requirements:	AVL_SEC_R08, AVL_SEC_R09
Tools:	Device.Connect (AVL)
KPI:	AVL_SEC_USC3_KPI_3.1_1

Table 3.6 is summarising all discussed and potential use case scenarios for the AVL_SEC use case. In this section, we are reporting about AVL_SEC_UCS1.

3.3.1. Use case scenario AVL_SEC_UCS1 — Learning-based model checking for security testing

We develop a test case generator based on automata learning (including ANN-plausibility checks) that uses fuzzing and model checking to use traces of specification violations.

The goal of this use case scenario is to develop a learning-based testing toolkit that (1) learns behavioural models of communication protocols used for in-vehicle communication, and (2) includes a model-based testing and verification framework.

Challenges

First, we need to learn behavioural models of communication protocols. In-vehicle communication comprises different protocols. As a first step, we will focus on wireless protocols used for car access,

e.g. Bluetooth Low Energy (BLE) or the NFC protocol. Later, we will also explore wired communication protocols, e.g. via the CAN interface and specific V2X protocol(s).

Creating a testing and verification framework for these protocols introduces challenges at a different level. The testing framework should create test cases for security-critical scenarios. For this, we develop a model-based framework that allows for using the same interface for different protocols (using different adapters) and that is able to automatically hand over the model to a model checker (e.g. NuSMV) or a stateful fuzzing framework to enable black-box testing.

3.3.1.1. Summary of preliminary results

We developed a stateful black-box fuzzing technique for testing BLE devices. We introduce learning-based fuzzing as a novel security testing technique that combines automata learning and fuzz testing. Fuzzing is a security testing technique where (pseudo-) random inputs are executed on the system under testing in order to reveal unexpected behaviour such as crashes. Figure 3.7 describes our learning-based fuzzing framework for a System-Under-Test (SUT) running a BLE stack in a two steps procedure. In the first step, we learned the behavioural models of BLE stack implementations. For the learning procedure, we created a BLE interface that included another BLE device that allowed us to communicate with the SUT by sending customised BLE packets and receiving the corresponding responses. The developed learning framework uses active automata learning algorithms provided by the Python library AALpy to learn behavioural models. In addition, the learning framework implemented handling procedures for lost and delayed packets. The learnt behavioural model then built the basis for the second step: model-based fuzzing. In model-based fuzzing, we tested the conformance between the learned behavioural model and the BLE device using a fuzzing test suite that provides state-coverage. To generate fuzzed inputs, we modified BLE packets such that they include invalid and boundary values.

We evaluated our learning-based fuzzing technique on six different BLE devices. The selected BLE devices are standard BLE devices from different manufacturers, including also different boards from the same manufacturer. A first result of our learning-based fuzzing technique was that all learned behavioural models of the BLE stack implementations were different. Hence, automata learning can be used to fingerprint black-box devices. Model-based fuzzing of these investigated BLE devices uncovered reliability issues in four out of six devices where BLE packets could be sent causing the tested devices to crash. Furthermore, new states were discovered through fuzzing.

As a next step, we evaluated whether the learning-based fuzzing approach can be also applied to already existing software systems in the automotive industry. In another case study, we learned the BLE devices implemented in a Tesla Model 3. The Tesla Model 3 uses BLE as one method for the car access. Therefore, also the corresponding key fob uses BLE. We learned the BLE device installed in the car as well as in the key fob. The learned models have eleven states and were behaviourally equivalent. The learned model distinguishes again from all other learned BLE devices of the previous case study. Learning of the BLE devices implemented in the Tesla Model 3 took approximately 87 minutes.

Learning-Based Fuzzing

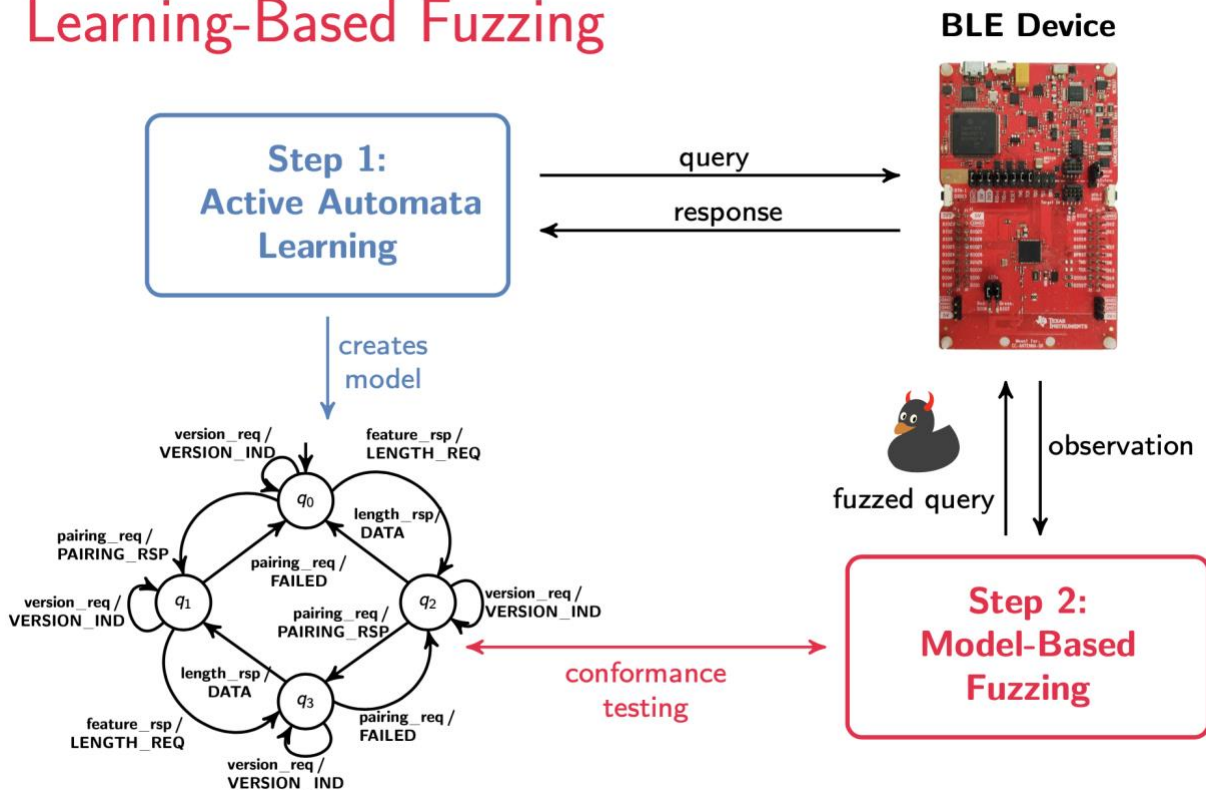


Figure 3.7 Two-step learning-based BLE Fuzzing Framework

3.3.1.2. Evaluation of results considering requirements coverage

The developed solution fulfils AVL_SEC_R01¹³ by using automata learning algorithms to derive state models of protocols on SUTs (to be precise at time of writing BLE communication protocols on Bluetooth interfaces of IoT devices). For AVL_SEC_R02¹⁴, AVL created a suitable dataset for training an ANN. It contains traces of CAN Bus data from a known vehicle and can be labelled with a so-called Can DataBase Container file and contains multiple scenarios of typical driving manoeuvres. This Dataset will then be subsequently used to fulfil AVL_SEC_R03¹⁵ to create a test oracle that is usable for cybersecurity testing and is able to detect anomalies in the CAN Traffic while a test is executing. The

¹³ Use automata learning and ML techniques to derive SUT models

¹⁴ Use an ANN to perform plausibility checks on models

¹⁵ Train ANN on SUT topology discovery using test observation

model checking component (AVL_SEC_R04¹⁶) is future work and will be implemented in the last AIDoArT period.

3.3.1.3. Evaluation of results considering KPIs

For KPI_1.1 (Table 3.7), we use the learning for BLE protocols as a basis. Having successfully learned models, $k > 1$ the KPI has been met $100 \cdot (k - k_0) / k_0 = 100 \cdot (1 - 0.1) / 0.1 = 900$ (**Target ≥ 100**).

KPI_1.2 (Table 3.8) cannot be measured yet, because so far only the model derivation part has been implemented. The formal specification is yet to be done.

AVL_SEC_USC1_KPI_1.1_1

Table 3.7 Case study KPI “AVL_SEC_USC1_KPI_1.1_1”

KPI Identifier: AVL_SEC_USC1_KPI_1.1_1		Scenario Identifier: AVL_SEC_USC1	
KPI Description:	Increase the number of learned protocol implementation models.		
Refined AIDoArT KPI:	<i>Description:</i>	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> \geq 25%
KPI Measure:	k = Number of learned models		
KPI Baseline:	<i>Source:</i>	Current testing platform: number of learned models / number of total models	
	<i>Value: k₀ =</i>	0.1 (1 out of 10)	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	\geq 100%

AVL_SEC_USC1_KPI_1.2_1

Table 3.8 Case study KPI “AVL_SEC_USC1_KPI_1.2_1”

KPI Identifier: AVL_SEC_USC1_KPI_1.2_1		Scenario Identifier: AVL_SEC_USC1	
KPI Description:	Increase the number of found Counterexamples.		
Refined AIDoArT KPI:	<i>Description:</i>	Improvement of the early detection of system deviations.	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> \geq 30%
KPI Measure:	k = Number of deviations detected		
KPI Baseline:	<i>Source:</i>	Specification vs. implementation: number of detected deviations / number analysed models	
	<i>Value: k₀ =</i>	0.1 (1 out of 10)	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	\geq 30%

So far, examined systems under scrutiny are according to the given specifications. Still through the learned models, it is expected that the found deviations will rise during the project's course.

¹⁶ Use formal model checking methods to derive test cases out of a system model

3.3.2. Planned improvements

For the next cycle, we plan for additional model learned (e.g., for a V2X protocol) as well as a model-based automated checking method that enables the testing framework to check the system-under-test for (a) specified security properties and/or (b) deviations from a specification (e.g., modelled from a standard). Furthermore, we plan for integrating the learned models into a model-based fuzzer, but this is of lower priority than the model checking and therefore only conducted if the timely resources are available after the model checking implementation.

3.3.3. Planned demonstration

Until the end of the project, we plan a demonstrator for a standards-based specification checker for BLE that is able to compare a learned model with a specification model built after the BLE standard. In addition, a preliminary version of this demonstrator may be shown at upcoming AIDOaRt plenary meetings.

The demonstrator will consist of a working BLE learning setup that is able to infer a model in the form of a (Mealy) state machine of an actual implementation from a BLE device in an active, black-box manner and a Mealy model of the BLE specification modelled after the BLE standard. The demonstrator will automatically compare the behavioural equivalence of both models using bisimulation and/or trace equivalence. Therefore, the demonstrator will be able to automatically assert BLE devices for their compliance with the standard, as well as pointing out possible deviations in the form of traces of the input that lead to the deviation and the respective output in dissensus with the standard.

3.4. Case Study description AVL_ODP - Optimization of Development Processes

Overview: The need for and trend towards traceability among data artefacts in the development of complex CPS will in the foreseeable future lead to structured data being created during development, i.e., data that precisely captures how the product under development evolves along the development process. The two main questions addressed in this use case are: (1) How to synthetically create such structured data (in sufficient volume required for training state-of-the-art ML algorithms)? (2) How to train process models on that data for optimising development projects?

Industrial interests: The main driver for creating such process models is to utilise data generated in (past) development projects to make CPS development more effective (e.g. by detecting trends in a product development project that likely will not bring the product under development closer to product target fulfilment and thus should be avoided) and more efficient (e.g. by identifying tests that could be replaced by simulation while still providing the same information gain).

Table 3.9 Synopsis of the case study AVL_ODP focusing the scenario AVL_ODP_UCS2

Use case scenario AVL_ODP_UCS2 ML-based KPI prediction	
Description:	Learn a data-driven process model that predicts the evolution of product’s related KPIs (e.g. vehicle energy consumption) and parameters (e.g. vehicle weight) in an ongoing development project. The process model is trained with past finished projects and parameterised with the history of the current ongoing project. With this model, the project manager can check whether the project is or is not on track to reach all KPI targets and can take appropriate measures.
Requirements:	AVL_ODP_R02
Tools:	MOMoT (JKU), Modeling of processes and knowledge base (UCAN)
KPI:	AVL_ODP_UCS2_KPI_1.1_1

In this section, we will report about the AVL_ODP_UCS2 use case scenario as outlined in Table 3.9.

3.4.1. Use case scenario AVL_ODP_UCS2 — ML-based KPI prediction

In this use case scenario, we want to train a data-driven process model that predicts the evolution of product-related KPIs (e.g. vehicle energy consumption) and parameters (e.g. vehicle weight) in an ongoing development project (see Figure 3.8). The process model is trained with past finished projects and parameterised with the history of the current ongoing project. With this model, the project manager can check whether the project is or is not on track to reach all KPI targets and can take appropriate measures.

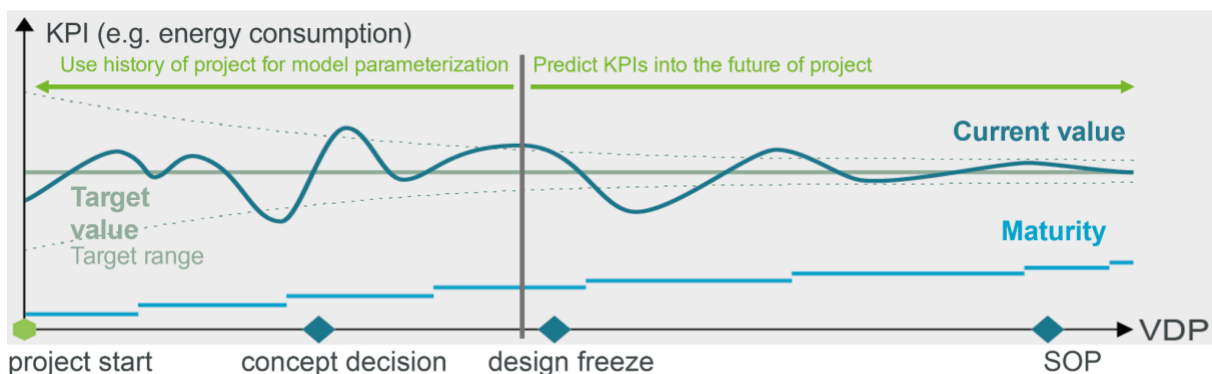


Figure 3.8 Evolution of product-related KPIs along the vehicle development process (VDP). The vertical grey bar indicates the present time in a project. History data from the project is used for parameterization of a model, which predicts the KPI evolution into the future of the project.

Challenges

The development of CPS products – due to their inherent complexity – typically contains inefficient development phases like dead ends (i.e., when teams at some point in time realise that their initial solution concept does not work, so that they have to start over again). Ideally, these phases should be identified as soon as possible and be omitted to make development more efficient.

This use case scenario addresses the following challenge: is it possible to emulate development projects containing these flaws/patterns in order to synthetically create realistic data (such as outlined in Figure 3.9) that can be used for training ML algorithms? Figure 3.9 schematically shows the evolution of one KPI along the vehicle development process, including typical patterns and flaws: KPI evolution in real-world projects shows an oscillation around and a convergence towards the target value associated with the KPI. Segments in which the KPI evolves away from the target can be considered as flaws, because development effort is spent without getting the product closer to target fulfilment.

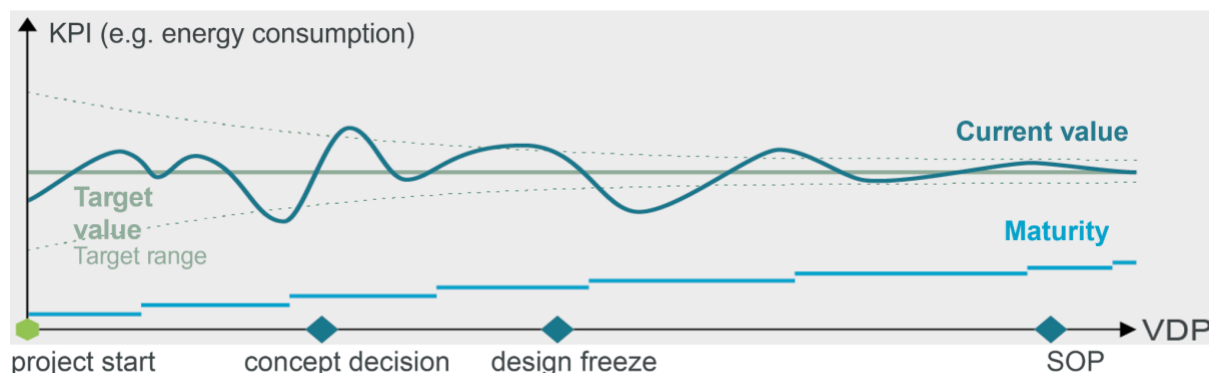


Figure 3.9 Possible evolution of a vehicle KPI (here: energy consumption) over various vehicle development process (VDP) phases

The solution approach for this challenge is to simplify the CPS development process to a high-dimensional optimisation problem, which means: product parameters, which encode product design, are iteratively optimised until all product KPIs can meet their individual target values. This renders optimisation solution components applicable to this challenge.

Tools

To set up the demonstrator for this case study, we want to use JKU's MOMoT solution component, specifically its model-based optimisation capabilities.

3.4.1.1. Summary of preliminary results

To be able to emulate CPS development projects – more precisely, vehicle/automotive development projects in this use case scenario – a transformation of product parameters (input) to product KPIs (output) is required. In real-world development projects this transformation is done by a multitude of different simulations and tests. In this use case scenario, to reduce effort, only one or a few simulation models in the form of simulation-tool-neutral FMU (Functional Mock-Up Unit)¹⁷ simulation models shall be used to compute KPIs out of parameters.

To emulate vehicle development projects based on FMU simulation models, we need to create a framework to manipulate the product parameters that parameterize FMU simulation models, execute the simulation model with these parameters and compute the corresponding KPI values.

¹⁷ [Functional Mock-up Interface \(fmi-standard.org\)](http://fmi-standard.org)

AVL is working on a suitable simulation model (i.e., simple enough to build a proof-of-concept and complex enough to emulate a non-trivial product development process) that shall represent the product under development.

JKU has started integrating (in MOMoT) an existing metamodel for FMU simulation models¹⁸ and especially FMU parameters, aiming at reusing the results from the AIDOaRt's parent project Megamart2¹⁹.

For executing the parameterized simulation models, a suitable FMU execution environment needs to be set up and connected with MOMoT, such that MOMoT provides the inputs (parameters) for a new simulation run, triggers the execution of the simulation run, and receives the results of the simulation run in order to compute the parameters for the next iteration of the simulation. Different FMU execution frameworks exist, from open source frameworks to proprietary ones such as CRUISE™ M or Model.CONNECT™ by AVL. JKU is assessing these alternatives in order to find a suitable framework for connecting with MOMoT.

3.4.1.2. Evaluation of results considering requirements coverage

The demonstrator currently under development has the target of emulating the evolution of product parameters and product KPIs, which shall ultimately provide the required training data for ML models. The demonstrator contributes to the requirement AVL_ODP_R02 in two ways:

1. For a given set of initial optimisation hyperparameters, it shall emulate a development project, i.e., forecast the evolution of both product parameters and product KPIs for the whole project. The associated metric for measuring the requirements coverage is: are the generated datasets/forecasts realistic, i.e., do they exhibit features and flaws also observed in real-world data? This metric can only be evaluated qualitatively by domain expert assessments based on expert experience.
2. It will provide the basis, i.e., the training datasets (emulated projects for different sets of hyperparameters), for creating a second set of models: data-driven process models, which shall be able to forecast the KPIs for the remainder (beginning at any point in time in the project) of unknown development projects, i.e., development projects with unknown hyperparameters. The associated metric for evaluating the requirements coverage is rather simple: how well is such a model capable of forecasting KPIs for the remainder of a development project which it has not seen during training. This metric can be evaluated by separating the datasets generated in (1) into a training and test dataset and measuring the deviations of KPI predictions for the test datasets.

3.4.1.3. Evaluation of results considering KPIs

The demonstrator currently under development is the basis for the development of the data-driven process models the performance of which are evaluated by KPI_1.1, in the sense that the current demonstrator will provide the training data required for creating data-driven process models.

¹⁸ org.eclipse.papyrus-moka/fmu-metamodel.ecore at 49b8b5549283a32ba3db2cf165fb4ee2dd18fa15 · megamart2/org.eclipse.papyrus-moka (github.com)

¹⁹ MegaMart2 - MegaModelling at Runtime (megamart2-ecsel.eu)

Thus, KPI_1.1 cannot be measured yet. The measurement of KPI_1.1 requires the creation of data-driven process models that can predict the parameter and KPI evolution for the remainder of a development project. The time how far into the future the model will be able to correctly predict the KPI evolution (compared to the test data) will be equal to the improvement of the time required for the identification of misleading trends, since the model will then also be able to predict flaws/inefficiencies in the future, which can then be at present identified by projects managers.

AVL_ODP_UCS2_KPI_1.1_1

Table 3.10 Case study KPI “AVL_ODP_UCS2_KPI_1.1_1”

KPI Identifier: AVL_ODP_UCS2_KPI_1.1_1		Scenario Identifier: AVL_ODP_UCS2	
KPI Description:	Improvement of the time required for the identification of misleading trends, thus enabling earlier decision making and saving development time and effort.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 25%
KPI Measure:	k = Time required for the automated identification of design problems based on KPI prediction.		
KPI Baseline:	<i>Source:</i>	Domain expert’s experience	
	<i>Value: k₀ =</i>	1-2 days	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 25%

3.4.2. Implementation activities transversal to AVL_ODP_UCS2 use case scenario

During the discussions at recent plenary meeting²⁰ obvious similarities were observed between this use case scenario and VCE_UCS_02, due to using the same underlying technologies and aiming to solve similar issues: both use case scenarios focus on the optimization of simulation model output by changing either the simulation model or the simulation model inputs in a systematic way. Particularly, the use of the tool MOMoT from JKU was identified as the main solution technology in both use cases. Therefore the decision was made to merge the different concrete hackathon challenges and work jointly towards the same problem with the same solution provider, so that value could be gained via common problems and common discussions, while letting the solution provider employ the tools more readily via commonly agreed activities instead of addressing separate problems with the same tool.

3.4.2.1. Summary of preliminary results

So far the collaboration has been defined and regular meetings are organised between VCE/AVL/MDU and JKU who provide the proposed solution in the form of MOMoT. The preliminary results are the merging of the use case problems and a preliminary plan and implementation of the MOMoT tool for FMU execution and eventual parameterization and optimization of models.

²⁰ So-called hackathons have been organised in the course of the plenary meetings, which significantly supported cross-use case discussions and potential horizontal use case integrations.

3.4.3. Planned improvements

Concerning the utilisation of optimization and MOMoT for emulating development projects, the planned improvements are mainly about refining the demonstrator to make it and its data output more realistic. For example, refining the Ecore metamodel that defines optimization targets, model transformation rules and transformation constraints. Another planned improvement is researching whether co-evolution approaches are applicable to the demonstrator, especially for emulating real-world simultaneous/concurrent development.

With the chosen approach of emulating product development by casting it as a mere optimization problem, many characteristics of product development processes are, however, sacrificed to problem simplification. For example, development processes define specific process roles, phases, development streams, milestones, quality gates etc. Thus, as a countermeasure, we want to create a formal process model. UCAN's modelling of the processes and knowledge base solution component shall contribute to this use case by capturing and modelling the process and involving KPIs, parameters, and simulation models. For creating formal development process models, the discussions among AVL and UCAN have shown that it is necessary to capture different aspects of a development process (e.g., available and consumed resources) for such a process model to be useful for process optimization. Available plain modelling languages for technical systems (e.g. UML or SysML) might not be able to capture these aspects, while dedicated process modelling languages might not be able to capture the required aspects of the technical system developed in the product development process. The combination of SysML/UML with the MARTE profile has been identified as a promising starting point to create a formal process model capturing different relevant aspects of a technical product development process. Once a process modelling language is chosen, JKU can define model transformation rules to automate the generation of optimised process models.

If there is enough time left in the remainder of this research project, the task of identifying process faults/flaws in the predicted KPI data could be automated. This requires manually annotating flaws/faults in the training data and training a classification model on that data.

3.4.4. Planned demonstration

Until the end of the project, a demonstrator shall be developed that is both capable of:

- generating product parameter and product KPI time series data that shows similar patterns and features as data observed in real-world development projects;
- forecasting product KPI evolution and particularly flaws in KPI evolution, for unseen time series data.

At the next plenary meeting, a PoC is planned to be shown demonstrating two aspects of this use case:

- the optimization of product design (as encoded by product parameters) starting from an initial random product design/parameter set such that the product KPIs can meet their targets;

- the different consecutive iterations/generations during the optimization process ordered and plotted over time, show an evolution of parameters and product KPIs with respect to time that is similar to patterns observed in real-world development projects.

4. ALSTOM_CS03 (BT_CS03) case study “DevOps for Railway Propulsion System Design”

Note: Bombardier Transportation (BT) has been acquired by Alstom. The name of the company and the use case have been changed accordingly, but the abbreviations have been preserved (BT_*) for consistency with previous deliverables).

4.1. Case Study description

Alstom use case scenarios are formulated with the goal to automate data processing and data transfer between different stages of development process, multi-physics modelling and test data correlation. The potential outcomes of such an endeavour are improved design and development chain resulting in more efficient processes in test facilities, reduction of the overall costs, energy efficiency, etc.

Within the AIDoRt project, the Alstom case study incorporates two specific use case scenarios: in use case scenario 1, Alstom aims to automate and improve the requirements engineering process using an AI/ML solution that would facilitate analysis of new requirements. In use case scenario 2, by using AI/ML algorithms Alstom aims to automate parametrization of thermal models in a propulsion control system. Today this is performed manually by adjusting parameter settings during system testing.

The synopsis of the case study ALSTOM_CS03 (BT_CS3) can be found in Table 4.1 below.

Table 4.1 Synopsis of the case study ALSTOM_CS03 (BT_CS3)

Use case scenario BT_UCS_1 — Requirement Engineering Recommender System	
Description:	Improved analysis of the customer specifications for critical requirements and use of AI to provide recommendations for suitable actions.
Requirements:	BT_R01
Tools:	Requirement Ambiguity Checker (MDU), VARA (RISE), Requirements Similarity Checker (SOFT)
KPI:	BT_UCS_1_KPI_3.1_1, BT_UCS_1_KPI_4.1_1, BT_UCS_1_KPI_2.2_1
Use case scenario BT_UCS_2 — Model Parametrisation	
Description:	This use case scenario corresponds to parametrization of the machine thermal model in the propulsion system drive during physical system test. Using AI/ML enabled algorithms, it is aimed that the manual tuning procedure steps and the man hour involved in the temperature rise test could be reduced.
Requirements:	BT_R02
Tools:	CAMEO (AVL), Tool from MDU
KPI:	BT_UCS_2_KPI_3.1_1, BT_UCS_2_KPI_4.1_1, BT_UCS_2_KPI_4.2_1

4.2. Use case scenario BT_UCS1

Railway traction equipment consists of complex hardware and software elements that in their aggregation constitute cyber-physical systems. The business is driven by a bidding process, typically in the form of public procurement. Customers, i.e., railway rolling stock owners and/or operators issue detailed specifications for the complete trains out of which some directly affect traction systems and others can result in derived requirements. Despite the diversity between customers, most specifications address the same features and design aspects. However, there is a great diversity in the way the requirements are formulated.

Today, as a consequence, vast numbers of customer requirements are manually analysed, allocated and further broken down. To be carried out effectively, this usually requires highly experienced bid and customer project engineers and it remains a very time-consuming task.

The goal is to provide appropriate recommendations to the bid and project engineers in an automated manner based on requirement datasets for standard products and past projects. This includes finding requirement defects (such as ambiguities, vagueness...), allocating requirements to different teams and responding to the requirements (can we comply with it or not?).

4.2.1. Summary of preliminary results

The work on this use case scenario focuses on several sub-tasks of requirement engineering in railway traction domain:

- Requirement identification: requirements are typically obtained from customers as large text documents in various formats. The first task is to split the text into individual potential requirements, which may encompass one or more sentences. Splitting into too small chunks of text may render each requirement meaningless, whereas splitting into too large chunks of text may result in multiple requirements lumped together. The second task is to distinguish between requirements and background information; requirements must be implementable and verifiable. Taken together, this process is known as requirement identification.
- Ambiguity checking: requirements that refer to past experiences, accepted practices, unspecified standards, etc. are ambiguous and unverifiable. It is important to flag such requirements early so that they can be clarified by the customer.
- Similarity checking: if an equivalence can be established between a new requirement and a requirement in a standard product or a past project, a recommendation can be made by the system that such requirements can be accepted without further analysis. Furthermore, requirement qualification (i.e. text describing how such a requirement can be fulfilled) and proof (i.e. a reference to a specific design document or a test report) can be suggested by the system. Note that the final decision will still lie with the requirement engineers, but making automatic suggestions (with various degrees of certainty) will greatly simplify and speed up their work.
- Team allocation: each requirement not immediately accepted by the requirement engineers needs to be forwarded to an expert in a specific area, such as traction control (software),

converter design, safety, etc. Automatic team allocation will greatly speed up requirement processing. Note that a certain level of erroneous allocations is acceptable since the experts can manually reassign the requirements if needed.

Within the AIDOaRt project, the focus is on investigating the problems and suggesting solutions (implemented as demonstrators) that would use state-of-the-art technologies to solve domain-specific problems. Thus, the development steps are:

- Identifying the tasks and available data for analysis.
- Developing, training, and validating ML models for each task.
- Solving the problems of integration between the suggested models and existing engineering practices and tools.

We have currently completed the first step together with the solution providers, the solutions providers have suggested ML models for each task and are in the process of refining them. Development and first evaluation of the tools & trained ML models have been done using real-life data sets provided by Alstom. The next steps include validation using broader data sets and integration in the form of demonstrators.

The following tools have been suggested by the solution providers:

Requirements Ambiguity Checker (by Mälardalen University, Sweden)

The tool identifies ambiguous requirements from textual documents using a set of ambiguous keywords and patterns and NLP & AI/ML techniques. It was developed using a small amount of manually labelled data. The achieved accuracy for the limited data set used for validation is around 80%.

VARA (by Research Institutes of Sweden)

The tool performs automated similarity analysis and feature reuse recommendation using Natural Language Processing (NLP): VARA enables automatic analysis of textual requirements for a new project and identifies components and artefacts that can be reused from a previous project for the implementation of the new requirements based on similarity analysis. Allocation Prediction has achieved 71% accuracy on datasets from multiple projects (data is labelled as part of manual requirement engineering process at Alstom).

Requirements Similarity Checker (by SoftTeam, France)

The tool supports identification, extraction, and classification of requirements from textual documents with NLP & AI/ML techniques. It correctly identifies over 60% of similarity pairs as close or related (Alstom projects used for analysis share a large number of similar requirements, over 70%).

4.2.2. Evaluation of results considering requirements coverage

The functional requirement BT_R01, defined in Deliverable 1.1 Use Cases Requirement Specification, is formulated as follows: “NLP contextual analysis of requirements and match against database of responses/solutions”. The proposed solutions directly address this requirement and we can state that the requirement coverage will be 100%. The success rate of requirement analysis (i.e., identification and ambiguity checking) and matching (i.e., similarity checking) will be determined during validation of prototypes.

4.2.3. Evaluation of results considering KPIs

The following KPIs have been defined earlier (see, for example, Deliverable 5.6 Use Cases Development Report). At the current stage of AIDoArt tool development, we are dealing with prototypes that have been developed and validated using small, manually labelled datasets, and are not yet prepared for integration into the Alstom requirement management process. Therefore, it's not yet possible to evaluate these KPIs numerically. The plan is to develop higher fidelity prototypes within the next few months and perform validation in the production environment (i.e., with validation will be performed by requirement engineers and using full-scale datasets).

The current state of tool development should be judged as adequate and corresponding to the existing work plan, with a good prospect that the tools developed in the AIDoArt project will reach maturity before the end of the project.

BT_UCS_1_KPI_3.1_1

Table 4.2 Case study KPI “BT_UCS1_KPI_3.1_1”

KPI Identifier: BT_UCS_1_KPI_3.1_1		Scenario Identifier: BT_UCS_1	
KPI Description:	Share of automated process steps in customer requirements analysis		
Refined AIDoArt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g., predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	Customer requirement analysis is an engineering process that follows specific guidelines with identifiable steps. The KPI measure is the number of automated steps divided by the total number of steps, multiplied by 100.		
KPI Baseline:	<i>Source:</i>	Step-by-step description of the requirement analysis process	
	<i>Value: k₀ =</i>	0%	
Target:	<i>Increase:</i>	k	≥ 30%

BT_UCS_1_KPI_3.1_1

Table 4.3 Case study KPI “BT_UCS1_KPI_3.1_1”

KPI Identifier: BT_UCS_1_KPI_4.1_1		Scenario Identifier: BT_UCS_1	
KPI Description:	Relative time consumed for customer requirements analysis		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of parts of the DevOps process covered in the Use Cases with productivity improvement.	
	<i>Identifier:</i>	KPI_4.1	<i>Target</i> ≥ 30%
KPI Measure:	Customer requirement analysis is a distinct step and it is possible to report time for it separately. The KPI measure is the decrease of time spent, normalised across projects (normalisation is needed since different projects have different complexity), expressed as a percentage.		
KPI Baseline:	<i>Source:</i>	Time reporting tool	
	<i>Value: k₀ =</i>	TBD	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

BT_UCS_1_KPI_2.2_1

Table 4.4 Case study KPI “BT_UCS_1_KPI_2.2_1”

KPI Identifier: BT_UCS_1_KPI_2.2_1		Scenario Identifier: BT_UCS_1	
KPI Description:	Share of data sources accessed and managed automatically for use in the customer requirements analysis process		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the number of available data sources to be actually managed and handled in existing engineering practices.	
	<i>Identifier:</i>	KPI_2.2	<i>Target</i> ≥ 25%
KPI Measure:	In manual requirement processing, the requirement data set for a new project is typically compared to the requirement data set of one past project from the same customer or another customer in the same market. In automated requirement processing, it can be compared to requirement data sets of multiple past projects. The KPI measure is the number of requirement data sets of past projects used for requirement analysis of a new project.		
KPI Baseline:	<i>Source:</i>	DOORS requirement database	
	<i>Value: k₀ =</i>	1	
Target:	Increase:	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 25%

4.3. Use case scenario BT_UCS2 — Automated Model Parameterisation

Accurate temperature estimation in critical parts of a traction machine is essential for maintaining the reliability and safety in real time operation. To deal with, on the test bench, the real time operation is replicated to have an accurate estimation of the temperatures. However, it is computationally expensive to include physics-based models pertaining to the drivetrain components for estimation of all the input parameters. A reduced-order thermal model (abstract lumped parameter thermal networks (LPTN) with 4 nodes) with analytical based approaches substitutes this need. However, such an approach needs resources of expert attention for feasible parameterization. Today this is done manually by iterating parameter settings against measured data during systems testing. With offline

parameter identification algorithms based on machine learning methods, accurate motor temperatures can be simulated in the test rig efficiently with less manual iteration.

A dark grey-box lumped parameter thermal network consisting of 4 nodes was considered in this use case, as shown in Figure 4.1. The thermal parameters (thermal conductance and capacitance) are linearly varying and quasi-state and need to be identified from the measurements.

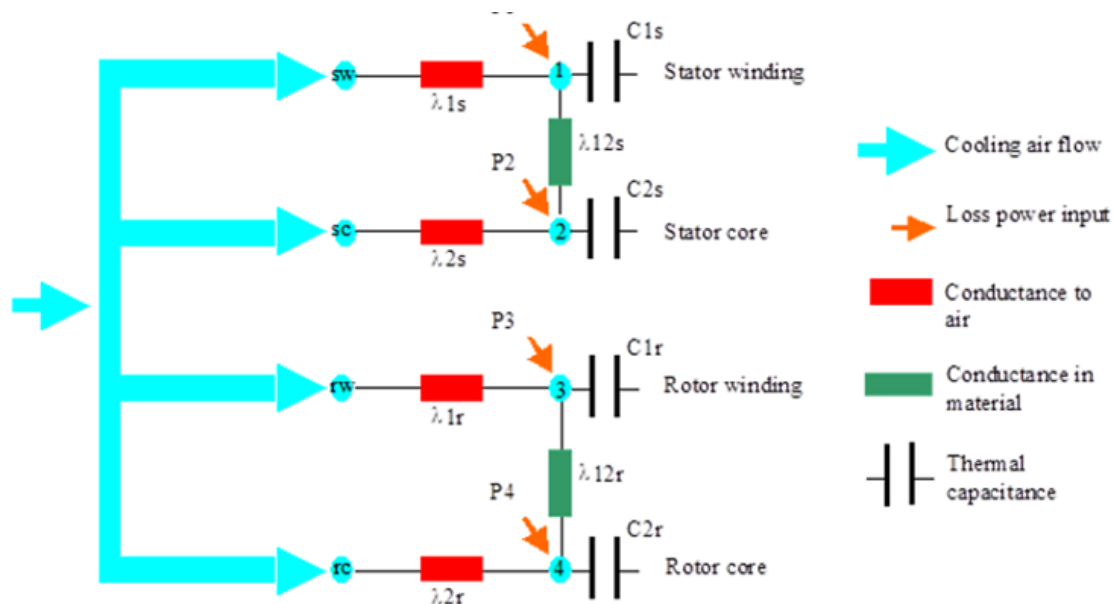


Figure 4.1 Four-Node Lumped Parameter Thermal Network Model

A computationally efficient and accurate means for offline temperature estimation is deemed necessary for obtaining the thermal model tuning parameters. To solve this problem, wide exploration of the state-of-the-art literature on the AI/ML techniques applicable to machine thermal analysis was conducted. After defining the challenges and requirements, subsequently a detailed level of deliberation with the solution providers during the hackathon challenges, a common understanding of the problem has been reached.

The main challenge includes identifying representative measured datasets that can accurately correlate the parameters of the reduced order model in the controller to the actual behaviour of the motor. The temperature model under investigation is a so-called grey-box model, whereby the structure of the input/output relationship is known to an extent, but some important parameters in the model are unknown and cannot be measured directly from the system under test. Moreover, some parameters change with respect to the operating point, which is given by the current, temperature, as well as the air flow. This means that the unknowns of the model are not scalar parameters but functions, and the challenge is to identify these sub-functions. A neural network class called “Symbolic Regression Models” was considered as a solution of the parameter identification task. Compared to standard regression model identification, which trains the parameters of a neural network with a given structure, symbolic regression models allow both the identification of the model structure as well as the corresponding regression parameters at the same time. This allows for a limited prior knowledge on the unknown sub-functions. Moreover, the implemented algorithm allows for the definition of a part of the model structure with unknown sub-functions, which makes this approach suitable to

identify grey-box type models without training completely black-box models which do not satisfy the required model structure.

4.3.1. Summary of preliminary results: Solutions proposed by AVL

In order to optimally parameterise the temperature model parameters, the predictive performance of each set of parameter combinations can be trained using AI/ML algorithms. Thereby, the model of the Unit Under Test (UUT) - e.g. a Simulink model - can be executed with different parameter combinations and the resulting temperature can be compared with the measurements from the original UUT on the test bed. The difference between the two signals can be learned by a machine learning model, see Figure 4.2. This so-called surrogate model can be used to find the optimal parameters, by using them as target functions in optimization algorithms.

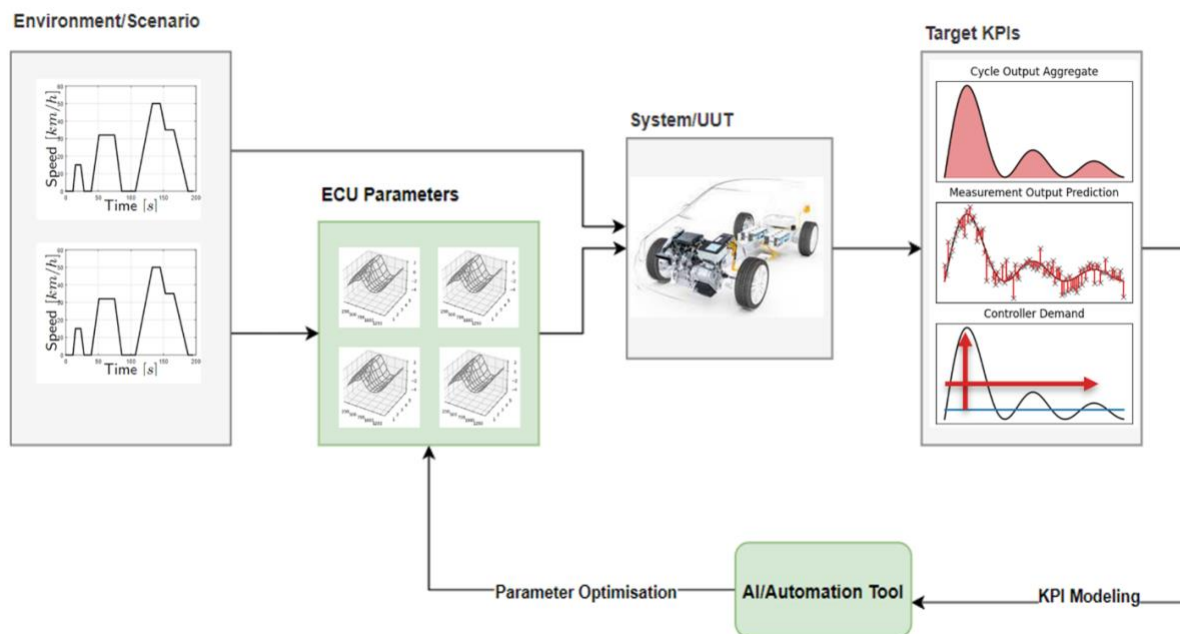


Figure 4.2 Surrogate modelling approach in AVL CAMEO: model feedback of the UUT, can be learned using machine learning techniques

In order to demonstrate the modelling approach, the data and the Simulink model provided by ALSTOM as well as other solution providers have been used in order to train initial error models, one for the rotor and one for the stator temperature, respectively. An intersection plot of these error models is shown in Figure 4.3. Here, two models are shown as a function of conductive and convective thermal parameters. It can be seen that those input parameters linked to the resistance/capacitance of the rotor have the main effect on the rotor temperature error and the same holds for the stator model parameters.

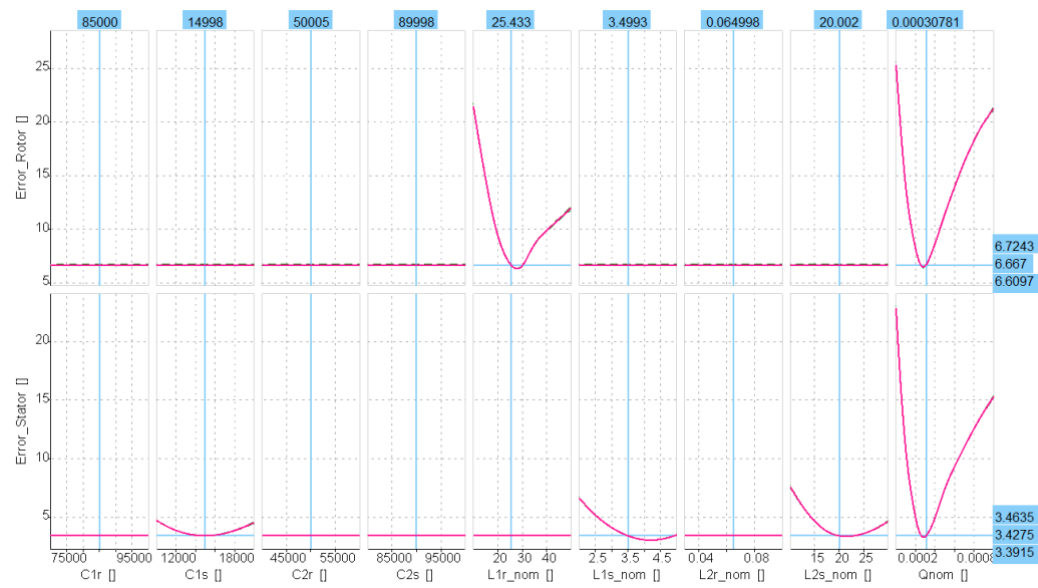


Figure 4.3 Black-box modelling approach in AVL CAMEO: model feedback, given by the model error of rotor and stator temperature can be modelled as a function of the model parameters

This model has been used in order to optimise the function for the optimal parameters. However, the parameters used so far as inputs are scalar and not functions of the operating space. We intend to generalise this approach in order to use surrogate modelling approaches to find the optimal parameters as functions of the UUT state, which is given by motor speed/torque, as well as the current temperature.

4.3.2. Summary of preliminary results: Solution Proposed by MDU

Data related to the operation of the traction motor, such as motor speed, torque, current, and voltage and temperatures, were collected during several driving cycles. The data has been preprocessed to synchronise the measurements. Figure 4.4 shows a sample of the preprocessed measurements.

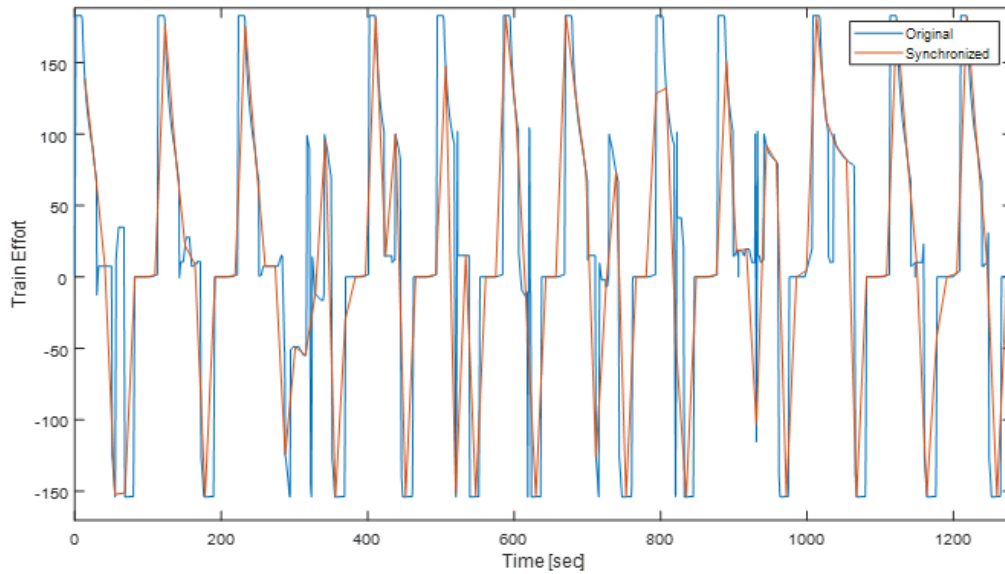


Figure 4.4 A sample of the preprocessed measurements

A data-driven reinforcement learning-based parametrization method is proposed to tune the parameters of the temperature model as shown in Figure 4.5. For each optimization cycle, the off-policy agent will find the parameters from one driving cycle and one model variant such that the final parameters will count for the different operating conditions and model uncertainty.

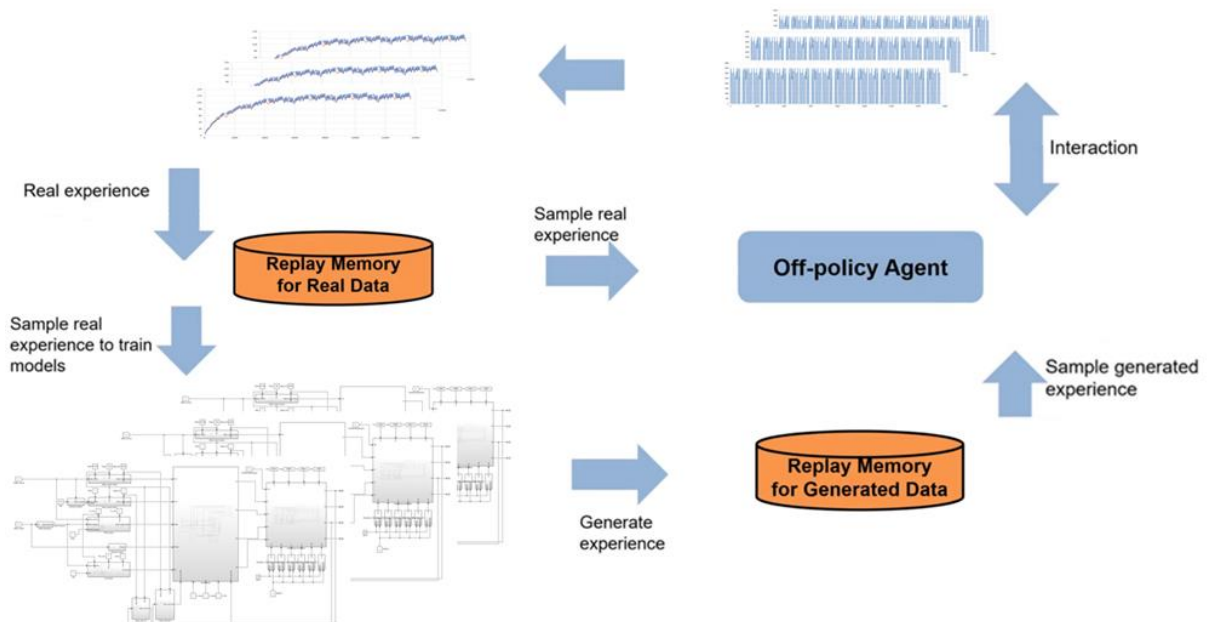


Figure 4.5 Reinforcement learning-based parametrization method.

A time series neural network has been initially trained to fit the temperature measurements that could be used as a black-box temperature model. Figure 4.6 shows the stator and rotor temperature measurements with the fitting curves.

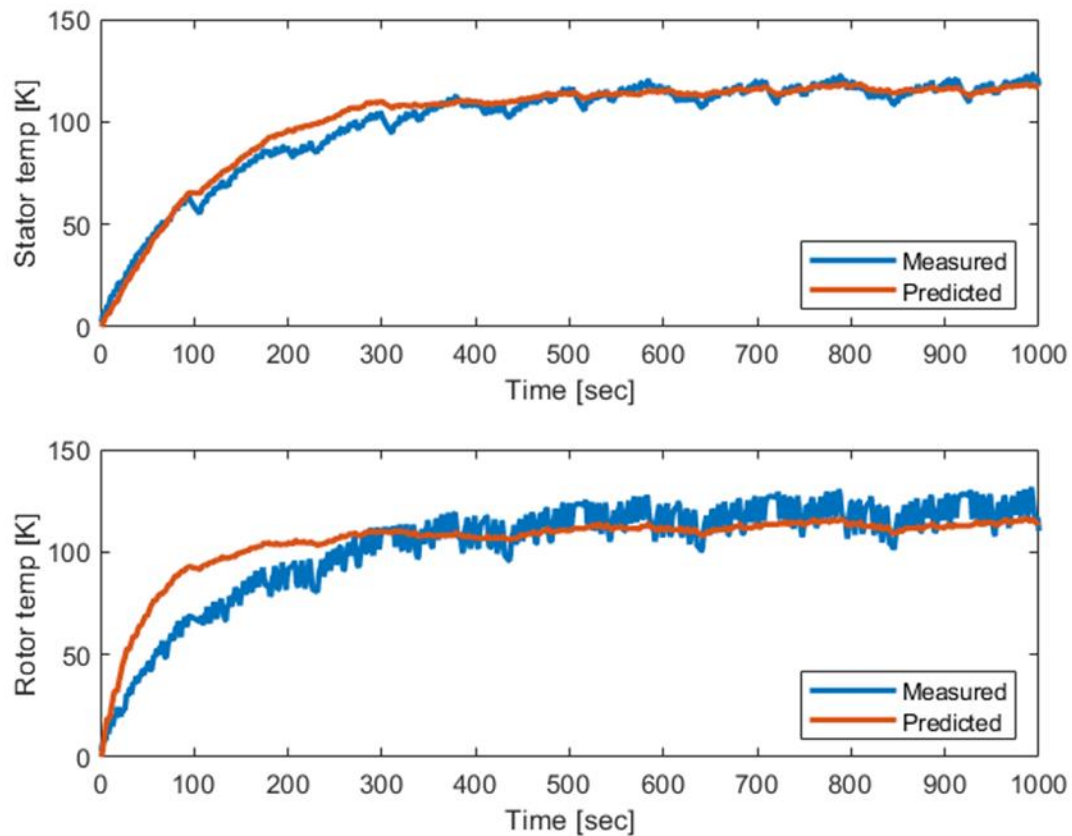


Figure 4.6 Fitting the temperature measurements

Off-policy reinforcement learning agents are employed to adjust the parameters of the temperature model. During each optimization iteration, the off-policy agent determines the parameters from a single driving cycle. Both Twin Delayed Twin Delayed Deep Deterministic Policy Gradient (TD3) and Twin Delayed Deep Deterministic Policy Gradients (DDPG) are utilised for optimising various driving cycles. These agents boast intricate neural network layers, enabling them to produce more experiences without directly interacting with the environment, leading to optimal parameter identification. They successfully optimised data from 6 driving cycles where the recorded stator and rotor temperatures closely matched the estimated temperatures by TD3 and DDPG, assisting in parameter determination. The prediction error or loss measured is minimal. Two generalised models have been created using these reinforcement learning agents, facilitating the discovery of optimal parameters under varying environmental conditions and accounting for model uncertainties. The outcomes of driving cycles using the temperature model and reinforcement learning agents are presented in Figure 4.7(a) & (b).

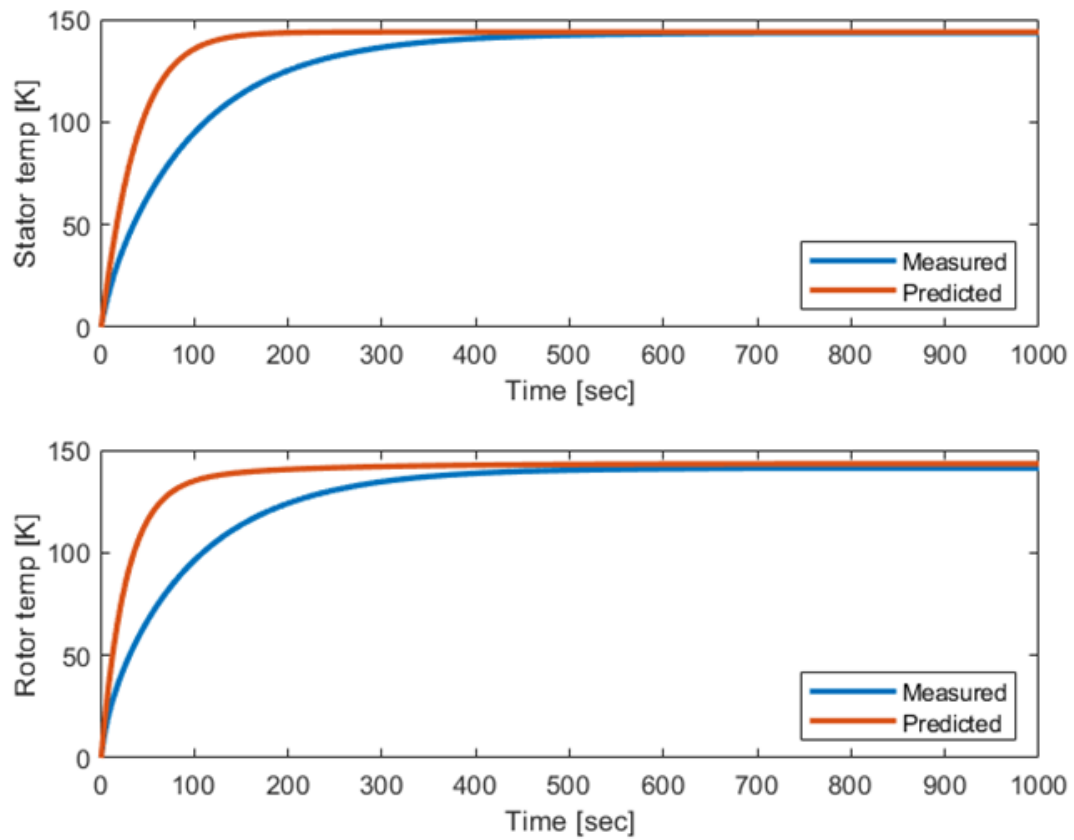


Figure 4.7 (a) Fitting the temperature measurements Drive-Cycle1

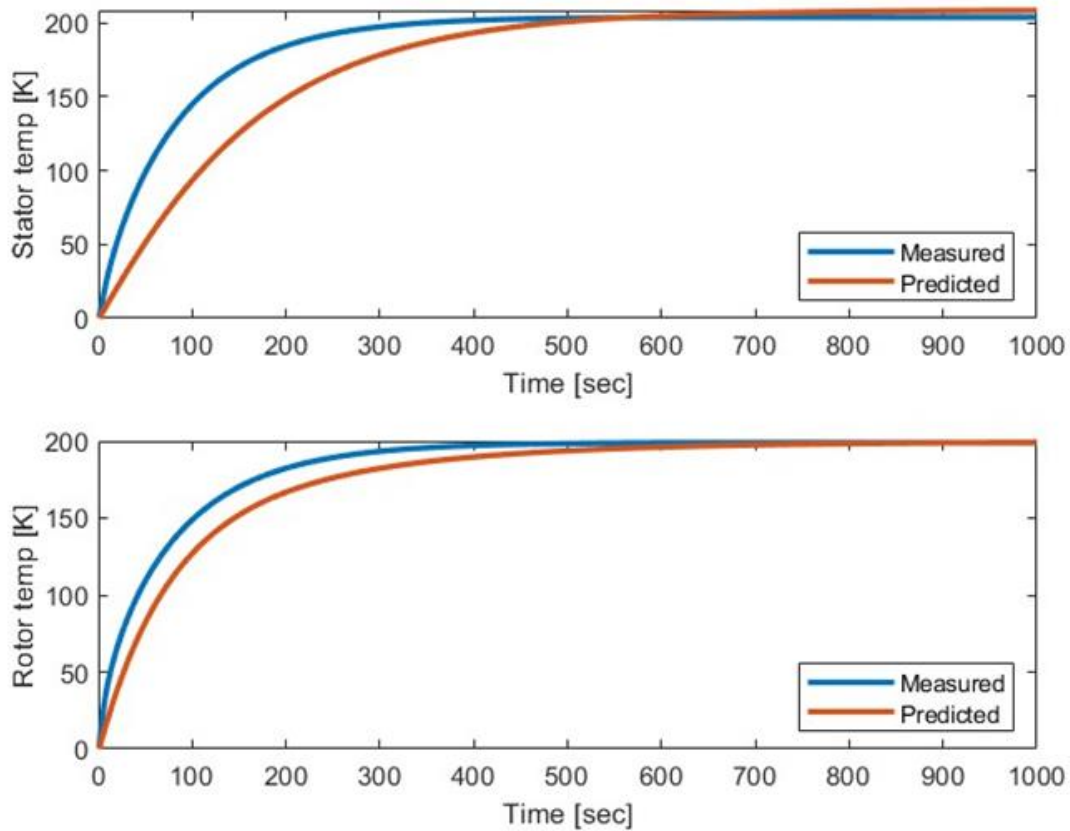


Figure 4.7 (b) Fitting the temperature measurements Drive-Cycle2

4.3.3. Evaluation of results considering requirements coverage

BT_R02 requirement “ML aided control model parameterization during propulsion system testing” is about the parameterization of the machine thermal model in the propulsion control system drive. The requirement contributes to the MDE and AI/ML AIDOaRT dimensions. It is aimed that the offline parameter estimation with aid of ML/AI based techniques would speed up and improve the time required in systems testing and in particular reduce the necessary manual effort by propulsion control experts in the test phase. In this context, reinforcement learning (RL) machine learning techniques (solution from MDU) and Active DOE with CAMEO tools (from AVL) are worked on to cover this requirement. The RL-based methods are promising data-driven techniques explored in the field of control of electric motor drives. RL methods enable learning in a trial-and-error manner and avoid supervision of each data sample. The algorithm requires a reward function to receive the reward signals throughout the learning process. Thus, the control policy could be improved on a continuous basis based on the measurement feedback. The RL type approach has been implemented and automated in AVL CAMEO Active DoE. This approach helps to reduce the number of necessary tests, e.g. compared to standard full-factorial test designs, by determining the necessary tests online, using

the machine learning models. It is planned that this online-modelling approach can be used in order to reduce the number of measurements required for parameter identification procedure of the temperature models.

4.3.4. Evaluation of results considering requirements coverage

BT_UCS_2_KPI_3.1_1

Table 4.5 Case study KPI “BT_UCS_2_KPI_3.1_1

KPI Identifier: BT_UCS_2_KPI_3.1_1		Scenario Identifier: BT_UCS_2	
KPI Description:	Share of automated process steps for control parameter tuning		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g., predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	Steps involved in temperature rise test in the test rig		
KPI Baseline:	<i>Source</i>	Experience from power lab engineers	
	<i>Value: k₀ =</i>	Nos. of steps followed in the present scenario	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

This KPI is planned to be assessed based on the number of steps followed on the real time test set up to validate the design concepts. In order to assess the performance, ALSTOM plans to use AI/ML estimated parameters as inputs to the controller to minimise the deviation against reduced order estimation methods. We hope to conduct these measurements included in the upcoming deliverables D5.8 and D5.9.

BT_UCS_2_KPI_4.1_1

Table 4.6 Case study KPI “BT_UCS_2_KPI_4.1_1”

KPI Identifier: BT_UCS_2_KPI_4.1_1		Scenario Identifier: BT_UCS_2	
KPI Description:	Relative reduction of time consumed for parameter tuning		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of parts of the DevOps process covered in the Use Cases with productivity improvement.	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = Hours used for temperature rise test in each driving cycle		
KPI Baseline:	<i>Source</i>	Actual recorded run time in the power lab	
	<i>Value: k₀ =</i>	Hours for each point temperature rise test*no. of points*no. of iterations + manual parameter tuning time* number of iterations* no. of points+ drive cycle test run	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

This KPI is planned to be assessed based on the number of steps followed on the real time test set up to validate the design concepts. In order to assess the performance, ALSTOM plans to use the solution as inputs to the controller to reduce the number of steps for design validation and thus hours spent in the test bench. These measurements shall be updated in the upcoming deliverables D5.8 and D5.9.

BT_UCS_2_KPI_4.2_1

Table 4.7 Case study KPI “BT_UCS_2_KPI_4.2_1”

KPI Identifier: BT_UCS_2_KPI_4.2_1		Scenario Identifier: BT_UCS_2	
KPI Description:	Model response deviation reduction with AI tuned parameters vs initial parameter set, i.e. improved accuracy in tuning parameter prediction with help of ML based algorithms.		
Refined AIDOaRt KPI:	<i>Description:</i>	Reduction of deviations from the specifications to improve predictability, conformance to specifications and proposal of system design refinements.	
	<i>Identifier:</i>	KPI_4.2	<i>Target</i> ≥ 30%
KPI Measure:	k = Percentage deviation from the actual		
KPI Baseline:	<i>Source</i>	Simulation hours for the implemented algorithm, physical test run time	
	<i>Value: k₀ =</i>	Deviation from the actual test run value from with ML algorithmbased estimation Vs educated guess of tuning parameters	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

This KPI is planned to be assessed based on the assessment of accuracy of ML/AI prediction against the real time test setup. These measurements shall be included in the upcoming deliverables D5.8 and D5.9.

4.4. Planned improvements

4.4.1. BT_UCS_1

The initial labelled datasets have been relatively small. The plan is to develop higher fidelity prototypes within the next few months and perform validation in the production environment using full-scale datasets, with validation performed by requirement engineers.

4.4.2. BT_UCS_2

The future planned improvements is to complete the development of the proposed lumped-parameter thermal model, which includes support of multiple nodes in the model, support of model inputs and efficient parameter training on cycle data. Testing the thermal model structure will be performed using the measurement data. There is also intention to validate the models based on reinforcement learning using more precise and realistic data compared to the previously used tool-generated data. This initiative will necessitate close coordination among the partners and evaluation of the defined KPIs. It is also essential to incorporate more intricate neural network layers into off-policy agents to offer optimal parameters grounded in diverse driving cycle data. Furthermore, adaptation of the AVL CAMEO Active DoE procedure specifically for the dynamic thermal model is planned for testing of the DoE procedure and evaluation of the KPIs.

4.5. Planned demonstration

4.5.1. BT_UCS_1

Demonstration will consist of the prototype tools deployed and evaluated in the Alstom production environment.

4.5.2. BT_UCS_2

A simulation-based temperature model will be developed that will be capable of handling data from various driving cycles. To enhance power control optimization, different agents of reinforcement learning-based optimization algorithms will be employed to work towards achieving the most optimal parameters for the controller, thus improving overall system performance.

5. CAM_CS04 case study “AI for Traffic Monitoring Systems”

5.1. Case Study description

Traffic monitoring system is usually a complex solution consisting of various sensors and components. CAMEA traffic monitoring use case includes systems that are mostly video-based or alternatively radar-based, and they can serve for applications as travel time estimation or vehicle detection and classification. Within the use case, we are investigating the possibility of enhancing current Traffic monitoring systems (within the Transport and Smart Mobility domain) using AI. Within the UC, we are mainly targeting low-power requirements using some of the following proposed techniques: suitable (embedded) platform, pre-processing of data, balancing between load and power consumption, and AI guided configuration and setup. Such systems can be then deployed to the field with possibility of autonomous operation with e.g. battery supply or solar power.

Previously, there was no AI employed within the use case. Using conventional methods for camera/radar data processing and classification, the system does not often meet defined accuracy requirements or with certain limitations and the system is not as reliable and of quality as demanded. There is often demand for low-power operation where the pre-processing of data and its balancing between load and power consumption is necessary. This is also done manually thanks to some expert knowledge. The process is very demanding and can differ for individual installations of the system. However, there could be many combinations missed that can lead to much better results.

CAMEA uses a so-called radar-on-chip platform that is a highly integrated solution with a radar signal part and processing cores embedded in silicon. The radar sensor needs to be configured during start-up which is specific for each application and often also location. Modern smart radars are very compact devices that can be mounted practically everywhere and operated, e.g. using battery or solar power. For this, power consumption of the device needs to be kept very low. Additionally, the radars are installed outdoors in many cases and need to resist various weather conditions. For this, radar is enclosed in a sealed box and thus it lacks any possibility for active cooling. Thus, heat dissipation of the platform must be considered as well.

The configuration of the radar sensor is very complex and some of its parameters have an influence on power consumption and heat dissipation of the radar chip. There are also some constraints on the configuration parameters that need to be fulfilled and some requirements pertaining to the environment sensing properties (e.g. maximum range, range resolution, angular resolution, and velocity resolution) that need to be met. With expertise, engineers can manually tune a radar configuration in a standard logic way to reduce e.g. the duty cycle of transmission.

The AI-based STGEM solution by ABO will be used for power-aware radar configuration and can possibly generate much more interesting combinations of configuration parameters that minimise power consumption while providing the same level of performance. STGEM will be connected to the CAMEA interface that accepts the radar configuration. Information about power consumption and core temperature will be collected periodically and subsequently analysed.

The *Case Study Synopsis* is in Table 5.1.

Table 5.1 Synopsis of the case study CAMEA

Use case scenario CAMEA_UCS3 — Enable Low-power Device Configuration	
Description:	As the configuration of radar sensor (and possibly also other sensors), there could be some combinations hidden for manual adjustment. Automatically generated/tuned configuration could thus bring lower power consumption and heat dissipation (critical for autonomous outdoor operation) keeping or improving defined sensor qualities. Specialized AI-based method can be used for this purpose.
Requirements:	CAM_R02
Tools:	STGEM (ABO)
KPI:	CAMEA_UCS3_KPI_3.1

5.2. Use case scenario CAMEA_UCS3 — Enable Low-power Device Configuration

The focus of this CAMEA scenario is low-power sensor configuration with possible processing improvement and auto-calibration enabling. Selecting individual configuration parameters with influence on power consumption can be quite challenging. As well, defining constraints for such parameters needs to be carefully done. We are aiming to measure both power consumption of the so-called RCA²¹ module and junction temperature of the radar chip (which is part of the RCA module itself). Power consumption and temperature with stock configuration will be measured and then we start with iterations of AI-tuned configurations. Reasonable reduction of 20% in power consumption and 5 degrees Celsius in temperature is expected to be measured.

Considering low-power operation, AI can play a role there as well - e.g. using suitable (embedded) platform, pre-processing of data, balancing between load and power consumption, and AI guided configuration and setup. When fulfilled (among others) by configuring a sensor device, tuning the configuration while keeping other qualities can be easily done. AI-based methods and defining some constraints for selected parameters and searching through generated configuration space can find the most suitable configuration for a given application keeping the lowest power consumption possible.

Main part of the planned testbed is the radar sensor itself. The radar sensor will be installed in both – lab and real conditions on a multi lane road with high and stable traffic density. This is connected to the PC (via Ethernet) that will be controlling the sensor. As has been said, the radar chip has its own capabilities for junction temperature measurement. As there are no power metres, the TI INA226 chip needs to be added into the box and it will be periodically read by the radar chip. At the beginning of each iteration, the sensor is restarted. Then, the selected configuration is sent to the sensor, and it starts with measuring. The measured values – current temperature, average and peak power consumption – will be sent to the PC with a defined period. Values will be collected and

²¹ The name RCA is coming from internal convention for naming devices in CAMEA.

evaluated for each iteration. Individual tuned configurations are generated using AI-based STGEM tools from project partner ABO.

5.2.1. Summary of preliminary results

Using our experimental radar sensor, we are measuring two values:

- Radar chip temperature (junction temp) – directly accessible via radar SDK so it can periodically read it and send it out.
- Radar module power consumption – cannot be measured with internal HW means. We will include a simple but accurate I2C power metre that will be directly read by the radar chip.

In case of temperature, stabilisation of the value should be considered. It is necessary to wait a minute or two to warm up a bit or cool down a bit. This is very necessary especially in case of the first start-up of the sensor. In case of outdoor operation, environment temperature should be also measured and used for compensation. In case of the power consumption, there is no need for stabilisation, and it can be measured immediately after configuration. However, this quantity will differ with load, radar should be fed with artificial/recorded data. Although we see it as a necessary step, this could be more problematic. Data needs to be corresponding with configuration, which could be problematic in case of various configurations and pre-recorded data. Also feeding a real radar device with such data is not very easy. Thus, after some testing of the approach on the table, we should move the radar outside. We will also use CAMEA infrastructure and make radar accessible via the internet. There we can get some real vehicle passes so the radar processing chain could be more loaded and realistic (and thus higher temperature and power consumption can be observed). As load can be dependent on many things – one of them is traffic density - we can get some additional information from the radar. This could be average load and traffic statistics or individual detections, so we compensate our observations based on that.

The data collected from the radar are stored in a given format and then analysed/evaluated with an initial metric that still needs to be adjusted to cover more aspects. Then, an iterative process is being executed to evaluate as many configurations as possible. Also, some guidance for configuration adjustment has already been defined and tested. A test generation algorithm called OGAN (in ABO's STGEM tool) was used to generate parameters that try to fulfil the use-case providers requirements of lowering the radar modules average power consumption, beyond that of what random search is capable of doing. Preliminary findings, as illustrated in Figure 5.1, indicate that OGAN exhibits the potential to optimise the radar system over time. However, it is important to note that the degree of optimization achieved does not entirely meet the desired objectives. This suboptimal performance may be attributed to two plausible factors: first, the radar system might have remained in a dormant or idle state during the testing phase, potentially impacting the optimization process. Second, it is conceivable that the search space encompassing the multitude of parameters subjected to optimization is excessively extensive, posing challenges in effectively navigating this vast parameter landscape.

To address these limitations and enhance the efficacy of future investigations, several avenues for further research emerge. One possible avenue involves the introduction of deterministic disturbances into the radar system during testing, thereby introducing more realistic operational conditions. Additionally, reducing the search space by focusing on a subset of parameters deemed most pertinent and meaningful for the optimization objectives could potentially yield more promising results.

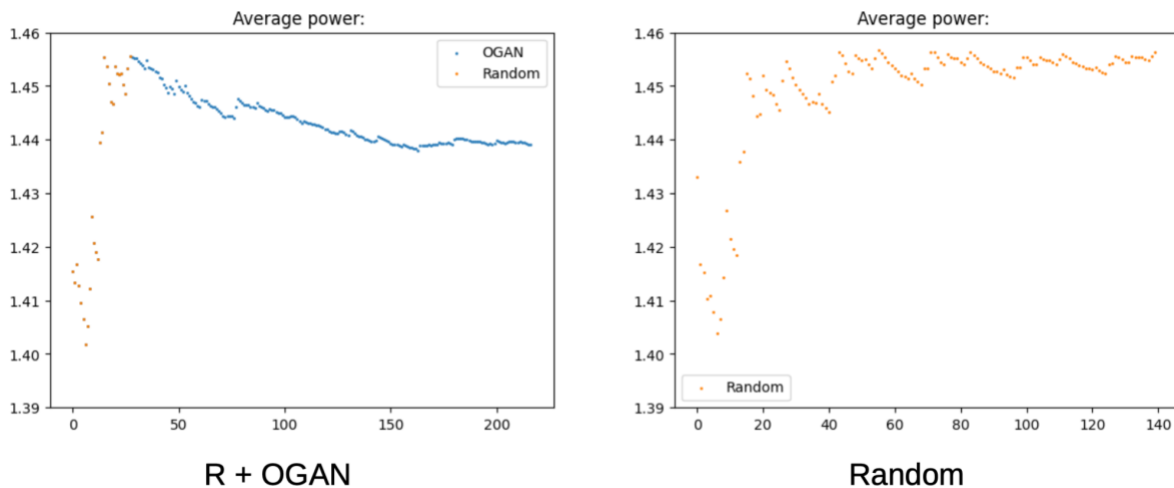


Figure 5.1 Visualisation of preliminary results with ABO’s STGEM tool AIDOaRt workflow

5.2.2. Evaluation of results considering requirements coverage

As the requirement CAM_R02 [1] needs AI-based methods used for optimization of radar-based system configuration, this requirement is then covered by the selection of STGEM (OGAN-based) method that is, within the UC, used for the configuration of individual radar sensors. This AI-based method is helping us to tune parameters of device configuration and thus meet requirements such as low power consumption and low heat dissipation.

Therefore, definition of any metric to measure requirement coverage is not applicable here.

5.2.3. Evaluation of results considering KPIs

The next iteration of KPI evaluation within the CAMEA case study is specified below in Table 5.5.

Table 5.2 Case study KPI “CAMEA_UCS3_KPI_3.1”

KPI Identifier: CAMEA_UCS3_KPI_3.1		Scenario Identifier: CAMEA_UCS3	
KPI Description:	Automating radar configuration generation and best candidate selection which is currently manual.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 20%

KPI Identifier: CAMEA_UCS3_KPI_3.1		Scenario Identifier: CAMEA_UCS3	
KPI Measure:	k = Ratio of parameters automatically adjusted to total parameters of the configuration.		
KPI Baseline:	<i>Source:</i>	CAMEA radar sensor API – No parameters adjusted at the beginning.	
	<i>Value: k₀ =</i>	0	
Target:	<i>Increase:</i>	k ≥	10% (†)
D5.6 Measure:	<i>Value: k₆ =</i>	2	k = 2% (‡)
D5.7 Measure:	<i>Value: k₇ =</i>	5	k = 5% (‡)

(†) The case study target is lower than the AIDOaRt target as there are hundreds of parameters and we don't expect to change so many.

(‡) We have just tested a few parameters as a proof of concept. More parameters will be added.

(+) More parameters within single configuration command has been tested and evaluated. More commands will be added as well.

5.3. Planned improvements

Planned improvements in the upcoming period: the configuration generation approach will be further tested. As some commands within the radar configuration have already been tested individually, we should extend the parameter space search approach to multiple parameters within multiple commands. As well, collected data from the radar should be extended and measurement periods adjusted to better reflect fast changes that happened during a very short radar transmission period.

As power consumption also differs with load, radar should be fed with artificial/recorded data. Although we see it as a necessary step, this could be more problematic. Data needs to be corresponding with configuration, which could be problematic in case of various configurations and pre-recorded data. Also feeding real radar devices with such data is not very easy. Thus, after some testing of the approach on the table, we should move the radar outside. We will also use CAMEA infrastructure and make radar accessible via the internet. There we can get some real vehicle passes so the radar processing chain could be more loaded and realistic (and thus higher temperature and power consumption can be observed). As load can be dependent on many things – one of them is traffic density - we can get some additional information from the radar. This could be average load and traffic statistics or individual detections, so we compensate our observations based on that.

5.4. Planned demonstration

The results can be demonstrated, as it is based on physical radar sensor, in form of live demo or video. The whole process or its individual parts can be then demonstrated (and possibly visualised) the following way: Radar sensor can be installed in real conditions on multi lane road with high and stable traffic density. This will be connected to the PC (via Ethernet) that will be controlling the sensor. As has

been said, radar chip has its own capabilities for junction temperature measurement. There is power meter chip added into the box and it is periodically read by the radar chip. At the beginning of each iteration, the sensor is restarted. Then, selected configuration is sent to the sensor and it starts with measuring. The measured values – current temperature, average and peak power consumption – will be sent to the PC with a defined period. Values will be collected and evaluated for each iteration. Individual tuned configurations are generated by using STGEM tool from ABO.

6. CSY_CS05 case study “Machine learning in interactive proving”

6.1. Case Study description

The case study is about using Artificial Intelligence to ease the process of proving Proof Obligations (PO) in the context of safety development of Railway Systems.

Developed by ClearSy, Atelier B is an industrial tool that allows for the operational use of the B Method to develop defect-free proven software using formal methods. Developing in B requires to write a formal specification, a machine-language implementation and to prove that the latter refines the former. Non-trivial projects generate thousands of proofs, of which a rough half can be automatically proved, and the rest needs to be manually demonstrated using a tool called the Interactive prover. Proving with the interactive prover requires mathematical skills, experience, and a lot of time. Moreover, modifying the system during its lifetime, even with a simple name change, often leads to “breaking all the proof”, as demonstrations need to be, slightly or deeply, modified to adapt to the change of hypothesis.

Therefore, we believe that Artificial Intelligence can help engineers in some aspects of the B development, by assisting or replacing those interactive proving or adapting proof to change of hypothesis.

Our industrial interests in the demonstrator that the case study develops are to reduce the time required to prove the correctness of development of proven software.

Our interests in data engineering are to tell if AI can adapt to our specific needs and can provide a framework in the domain of proving. To our knowledge, no previous work has been done in this specific area.

The *Case Study Synopsis* is in Table 6.1.

Table 6.1 Synopsis of the case study CSY

Use case scenario CSY_UCS1 — PO Classification	
Description:	<p>This UCS aims to classify proof obligations to choose early what automatic tool can be used to solve them. This is a fully automatized dev-ops process in which projects containing proof obligations will be analysed, classified and then solved while stored on a git repository.</p> <p>Development involves three steps:</p> <ul style="list-style-type: none"> • Manually classify a large sample of Proof Obligations • Building a vector structure from PO XML representations • Train a machine learning algorithm to classify • Use this machine to classify next Proof Obligations.
KPI:	CSY_UCS_1_KPI_1.1_1

Use case scenario CSY_UCS2 — Solve a PO with a variant of existing proofs	
Description:	UCS2 is about the part after the automatic proving, when the developer is supposed to manually solve the PO. When a model has already been worked on, there are manual demonstrations that are stored and that can be retried. If the PO is unchanged, the old demonstration can be reused. If it has slightly changed, it must be redone. Sometimes it is just one command that must be added or removed. This has to be done manually and takes time. This UCS tries to automatically write those new PO demonstrations.
KPI:	CSY_USC_2_KPI_3.1_1
Use case scenario CSY_UCS3 — Solve a branch of a PO using Reinforcement Learning	
Description:	Demonstrations are made by a developer, depending on the complexity of the PO, it takes between 3 and 30 minutes to solve a PO, with a 16 per hour mean. Sometimes, patterns can be found between PO and demonstrations, for example there are packs of PO that require variation of the same demonstration. UCS3 works on manual proving, by trying to recognize those patterns and solve parts of demonstration. We imagine it to be in the form of a button that the user presses to solve the rest of the demonstration.
KPI:	CSY_USC_3_KPI_3.1_1
Use case scenario CSY_UCS4 — Suggest a command in interactive proving	
Description:	Demonstrations are made by a developer and depending on the complexity of the PO, it takes between 3 and 30 minutes to solve a PO, with a 16 per hour mean. Sometimes, patterns can be found between PO and demonstrations, for example there are packs of PO that require variation of the same demonstration. UCS4 works on the manual proving, by trying to recognize those patterns at each step of the demonstration. We imagine it to be running permanently as a subtask and suggesting commands and parameters to the user while he or she solves the demonstration.
KPI:	CSY_USC_4_KPI_4.1_1
Use case scenario CSY_UCS5 — Automatic refinement of specification	
Description:	UCS5 is about generic B development. In B development, you have to write at least two models, one that is a formal specification close to the paper specification and one that is an implementation, ready to generate machine code. A large part of the PO comes from the obligation for the implementation to refine the specification. UCS5 is about having a tool that can write a refinement from a specification.
KPI:	CSY_USC_5_KPI_2.1_1

Use case scenario CSY_UCS6 — Automatic specification (abstraction) of machine code

Description:	UCS6 is about generic B development. In B development, you have to write at least two models, one that is a formal specification close to the paper specification and one that is an implementation, ready to generate machine code. A large part of the PO comes from the obligation for the implementation to refine the specification. UCS6 is about having a tool that can write a formal specification from a concrete implementation.
KPI:	CSY_USC_6_KPI_2.1_1

6.2. Use case scenario CSY_UCS1 — PO Classification

This UCS aims to classify proof obligations to choose early what automatic tool can be used to solve them. This is a fully automated dev-ops process in which projects containing proof obligations will be analysed, classified and then solved while stored on a git repository.

6.2.1. Summary of preliminary results

Previously, a plan had been developed, involving four steps:

- Manually classify a large sample of Proof Obligations.
- Building a vector structure for each proof obligation from the .poxml files (xml files containing all PO of a component)
- Train a Machine Learning algorithm to classify.
- Use this Machine to classify next Proof Obligations.

Manual classification has been realised using a scripted tool called Solyzer (figure 6.1), developed by Clearisy for AIDOaRt, that reuses data from past projects while it also measures the time of execution for coming benchmarking.



Figure 6.1 Solyzer splash logo

We also worked on data representation with the design and specification of the .poxml file.

In the last period, we worked on selecting the good technology to fulfil the next needs.

Cloud-based treatment

We first tried using cloud-based AI IDE like Google Colab for their models content and simplicity of use. In this case it was mandatory to use only anonymized data for confidentiality reasons.

We used a recurrent neural network, configured to classify the proof obligation Goals between unproved or proved states (and with different forces of automatic treatment).

The network was configured with 5 layers:

- embedding layer as entry layer,
- Long short-term memory with 64 cells,
- Dropout with frequency 0.3,
- Dense layer with 64 parameters and ReLu activation,
- Dense layer with one param to get the classification.

Using anonymized data, our results were not very good, mostly because it prevented us to use the Hypothesis part of the PO (results in Table 6.2):

Table 6.2 Results of first SolverPredict experiment

Model accuracy	0.8151
BCE Loss	0.4576

We realised that working with anonymized data was more of a burden than we foretold, so once that first results came we decided to transfer models and data locally.

Local treatment

We assembled a computer with enough memory and GPU to deal with RNN (Recurrent Neural Network) and started to reproduce the experiment with richer data, including hypotheses and more classification states.

The network was configured with 6 layers:

- embedding layer as entry layer,
- LSTM with 128 cells,
- Dropout with frequency 0.2,
- LSTM with 128 cells,
- Dropout with frequency 0.2,
- Dense layer with three units.

The model has been trained on 7 of our 30 projects, mostly to fit the capacity of our computer. The training was done during 6 epochs. Epochs are complete runs that train the model with the same full set of data but in different organisation and order. During the training we noticed a great convergence of the model, with a drop of the loss and increase of accuracy as it is shown in Figure 6.2 to 6.4.

```

717/717 [=====] - 432s 594ms/step - loss: 0.4999 - accuracy: 0.7898 - val_loss: 0.3952 - val_accuracy: 0.8435
Epoch 2/50
717/717 [=====] - 426s 594ms/step - loss: 0.3076 - accuracy: 0.8750 - val_loss: 0.3730 - val_accuracy: 0.8553
Epoch 3/50
717/717 [=====] - 427s 596ms/step - loss: 0.2413 - accuracy: 0.8998 - val_loss: 0.3485 - val_accuracy: 0.8616
Epoch 4/50
717/717 [=====] - 427s 595ms/step - loss: 0.2026 - accuracy: 0.9167 - val_loss: 0.3491 - val_accuracy: 0.8750
Epoch 5/50
717/717 [=====] - 426s 595ms/step - loss: 0.1757 - accuracy: 0.9294 - val_loss: 0.3518 - val_accuracy: 0.8768
Epoch 6/50
717/717 [=====] - 425s 593ms/step - loss: 0.1480 - accuracy: 0.9409 - val_loss: 0.3730 - val_accuracy: 0.8738
224/224 [=====] - 45s 200ms/step
    
```

Figure 6.2 Results of second SolverPredict experiment

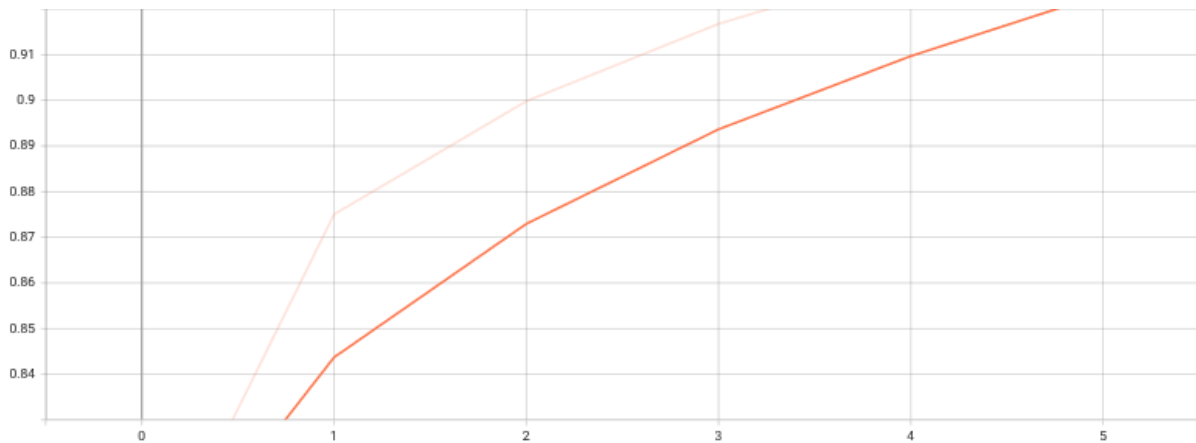


Figure 6.3 Evolution of accuracy through epochs of the second SolverPredict experiment

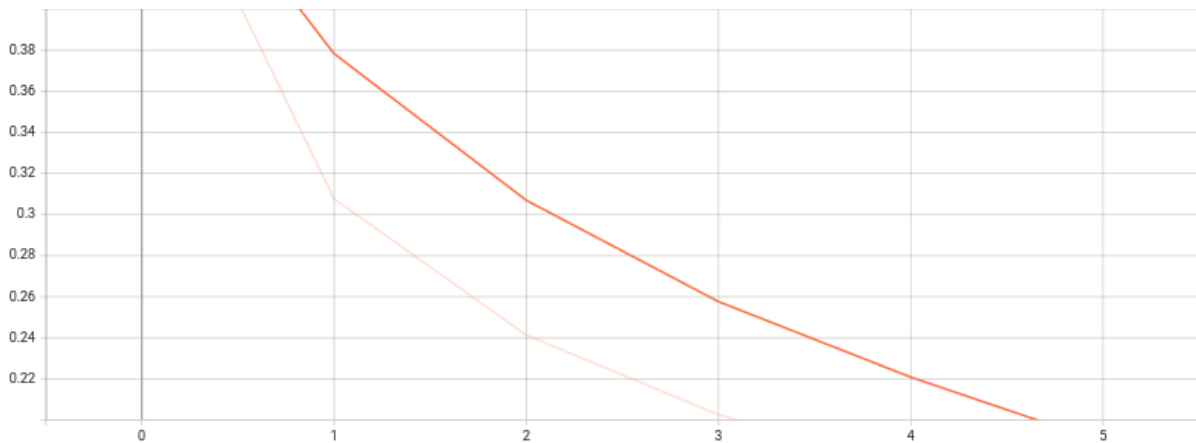


Figure 6.4 Evolution of entropy loss through epochs of the second SolverPredict experiment

Results on the test data showed that the model was able to correctly predict between 82% and 91% of each class as shown in the confusion matrix of Figure 6.5.

A confusion matrix is a table that gives the results of a classification over a test sample. Here the test sample is a part of the Training data that was put aside at the beginning of the training for this purpose.

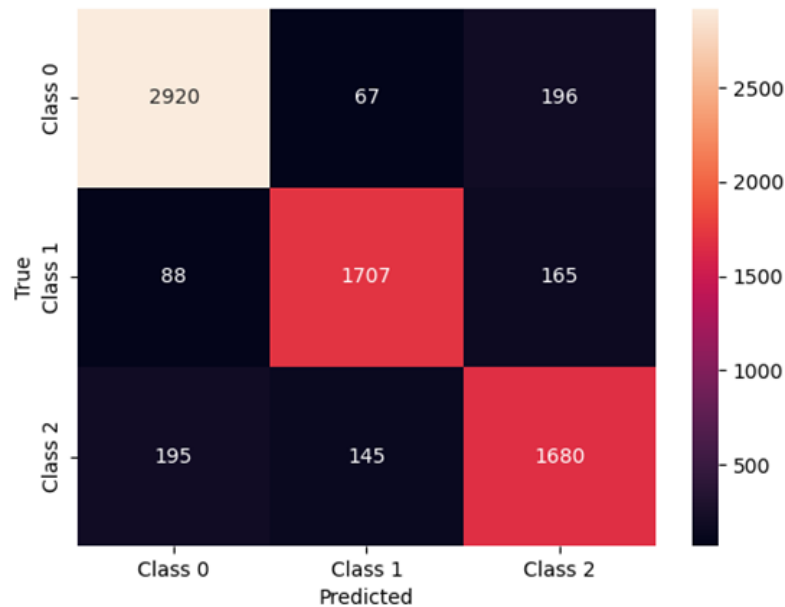


Figure 6.5 Confusion matrix of the second SolverPredict experiment

6.2.2. Evaluation of results considering KPIs

On the benchmark project *paso*, results were a little bit weaker than what we obtained on test data but still satisfying. The difference can be explained by the speciality of data on the project used for training and the one used for benchmarking. A larger training will be studied, there is still data that can be integrated, It could make the model more generic at the cost of a longer training.

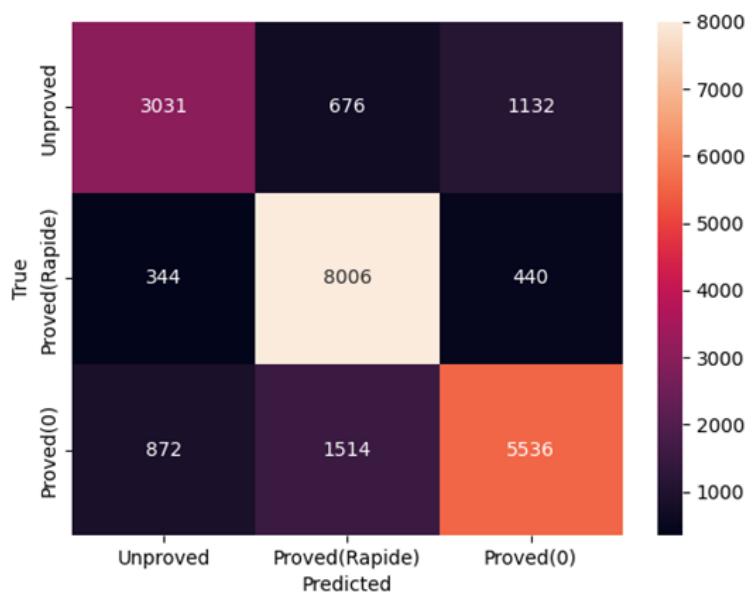


Figure 6.6 Confusion matrix of the second SolverPredict experiment

In the end, as shown in Figure 6.6, we obtained a satisfying result of 84% of proof obligations that were successfully proved in 30 minutes (10 for classification and 20 for proving) instead of 370 minutes for the initial benchmark. The gain of 90% is impressive, it shows that using AI classification for automatic proving is working. Nevertheless, 16% of proof obligations that were initially solved have been classified as unproved, this result mitigates the gain as it would need a time as long as the gain to identify them manually and solve them. This could be improved if we changed the weight and threshold of the model to make it prefer the most inclusive choices. For example, if a PO is FR, it can be classified as F0 or F1 and still be solved, a F0 can be classified as F1 and an unproved can be classified in any class. This kind of error would increase the time spent on proving but also the rate of proved PO. On the other hand, classifying every proof in the most permissive class would lead to the initial problem with a proving time of 370 minutes.

A last improvement of the tool could be in including other classes, ANR Project BLASST (<https://merz.gitlabpages.inria.fr/blasst/>) in which CLEARSY participates aims to use SMT and SAT provers with B proof obligations. For this, classifying on PO that can be solved with external solvers could be a help, but we need to get or build specific data since data from APERO or Prep does not contain SAT or SMT proved PO.

CSY_UCS_1_KPI_1.1_1

Table 6.3 Case study KPI “CSY_USC_1_KPI_1.1_1”

KPI Identifier: CSY_USC_1_KPI_1.1_1		Scenario Identifier: CSY_USC_1	
KPI Description:	Reduction of the proving time due to classification of the PO and reduction of unnecessary prover attempts.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 50%
KPI Measure:	k = Time in minutes for the total process of automatic solving		
KPI Baseline:	<i>Source:</i>	Time when each tool is tried on every unproven PO, from the fastest to the slowest (baseline process) on project 15 “paso”	
	<i>Value: k₀ =</i>	370	
	<i>Value : k =</i>	30	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 90%

6.3. Planned improvements

For the next cycle, we will address the problem of automatic proving. With the actual results, we confirmed that deep learning models are able to treat Proof Obligation material, we now need to find a way to train the model to propose commands of proof.

6.4. Planned demonstration

At this time, we are able to demonstrate the PoC of PO classification.

Using Solyzer and SoverPredict, we can demonstrate how data are processed, classified and actually proved. Choosing a short model or a part of a model, it could be done in a reasonable time for a live demonstration.

7. HIB_CS06 case study “AI DevOps in the restaurants business”

7.1. Case Study description

The purpose of the Case Study 06 by HI Iberia is to streamline the production of hardware/software solutions for small and medium business operations taking as a baseline the product TAMUS, offered by HI Iberia to restaurants and restaurant chains in Spain, with over 100 locations being exploited as of the writing of this report.

Managing the codebase, variants and day-to-day operations of the installed products is challenging for an SME, so the use case in AIDoArt looks for solutions to automate processes using AI and meet some of the Cyber Physical Systems risks posed by the system such as the data security problems in using mobile devices as order-taking systems and the usage of moving parts such as cash registers.

We divided the challenges posed by the Case Study in four general requirements that cover a multitude of details in the use case. These requirements can be fully consulted in D1.3 [2] but in summary they are:

- HIB_R01: Regarding the need to analyse using AI the produced logs generated by the system in order to detect anomalies in the operation of the system at runtime and anticipate points of failure.
- HIB_R02: Covering the usage of AI to manage the system requirements, maintained in a hands-on fashion by the TAMUS team and requiring substantial manual work to be analysed. The objective of this requirement is to enable requirement classification and assignment to developers by means of AI analysis.
- HIB_R03: Focusing on the packaging on new versions of the TAMUS system as ‘software bundles’ that can be then customised and sent to their intended restaurants of usage. AI is here used to validate each bundle before delivery.
- HIB_R04: Closing the DevOps circle by analysing the deployment process. AI is used to get a better understanding of the results of deployment and integration in the physical location of the restaurant. This relates to several CPS aspects such as the particulars of the restaurant’s layout, networking and connected hardware elements such as cash registers and locks.

Table 7.1 Synopsis of the case study 06 Restaurants

Use case scenario HIB_UCS1 — Automated log analysis	
Description:	<p>Analysis of the logs of the system using an automated tool that provides insight in key metrics, detects anomalies and produces issues that can be actioned upon by the development team.</p> <p>In this UCS we develop the means to (a) pre-process the logs generated by all actors in the restaurant system, (b) analyse them using text analysis AI tools and Natural Language Processing (NLP) systems to analyse user inputs in natural language and (c) generate reports for the systems analyst to take decisions on the deployment.</p>
Requirement	HIB_R01
Tools	HIB_logAnalyzer (HIBLA)
KPI:	HIB_UCS_1_KPI_1.2_1
Use case scenario HIB_UCS2 — AI requirements management	
Description:	<p>In this UCS we manage the requirements stored using a Trello²² board for the TAMUS system. This is done to get (a) automatic classification of the new requirements according to the categories used by the development team and (b) suggested developers from the team according to their past performance with a specific time of implementation of requirements.</p>
Requirement	HIB_R02
Tools	HIB Requirements Analyzer (HIBRA)
KPI:	<ul style="list-style-type: none"> ● HIB_UCS_2_KPI_1.1_2 ● HIB_UCS_2_KPI_1.1_3 ● HIB_UCS_2_KPI_3.1_1

²² <https://trello.com/> - Trello website

Use case scenario HIB_UCS3 — AI-enabled new versions of assets analysis	
Description:	Demonstrations are made by a developer, depending on the complexity of the PO, it takes between 3 and 30 minutes to solve a PO, with a 16 per hour mean. Sometimes, patterns can be found between PO and demonstrations, for example there are packs of PO that require variation of the same demonstration. UCS3 works on manual proving, by trying to recognize those patterns and solve parts of demonstration. We imagine it to be in the form of a button that the user presses to solve the rest of the demonstration.
Requirement	In this UCS we perform AI-enabled analysis of the newly developed components for TAMUS as new software bundles to install in the system for a given restaurant.
Tools	HIB_R03
KPI:	Custom CI/CD tools (TBD)
Use case scenario HIB_UCS4 — Deployment and analysis of new HW/SW versions	
Description:	In this UCS we perform and analyse the results of automatic updates to the code for the POS system including AI for compatibility and issue tracking. Aspects related to CPS are strongly monitored (e.g. link with the physical layout of each one of the restaurants such as kitchen, tables, etc. as well as the integration with hardware elements such as cash registers and printers).
Requirement	HIB_R04
Tools	Custom CI/CD tools (TBD) and HIBLA for analysis of the deployment logs
KPI:	<ul style="list-style-type: none"> • HIB_UCS_4_KPI_1.1_4 • HIB_UCS_4_KPI_3.1_2

In the period leading up to the development of this D5.7, good progress has been made in the completion of requirements HIB_R01 and HIB_R02, with progress in HIB_R03 and HIB_R04 more limited due to issues during the originally intended collaboration with AND. In the work for the resolution of the first two requirements good progress has been made with regards to the AI approaches, with selection of algorithms, baseline training and evaluation of the results undertaken. All of this will be detailed in the following subsections.

7.2. Use case scenario HIB_UCS1 — Automated log analysis

In this UCS, we develop the means to (a) pre-process the logs generated by all actors in the restaurant system, (b) analyse them using text analysis AI tools and NLP systems to analyse user inputs in natural language and (c) generate reports for the systems analyst to take decisions on the deployment.

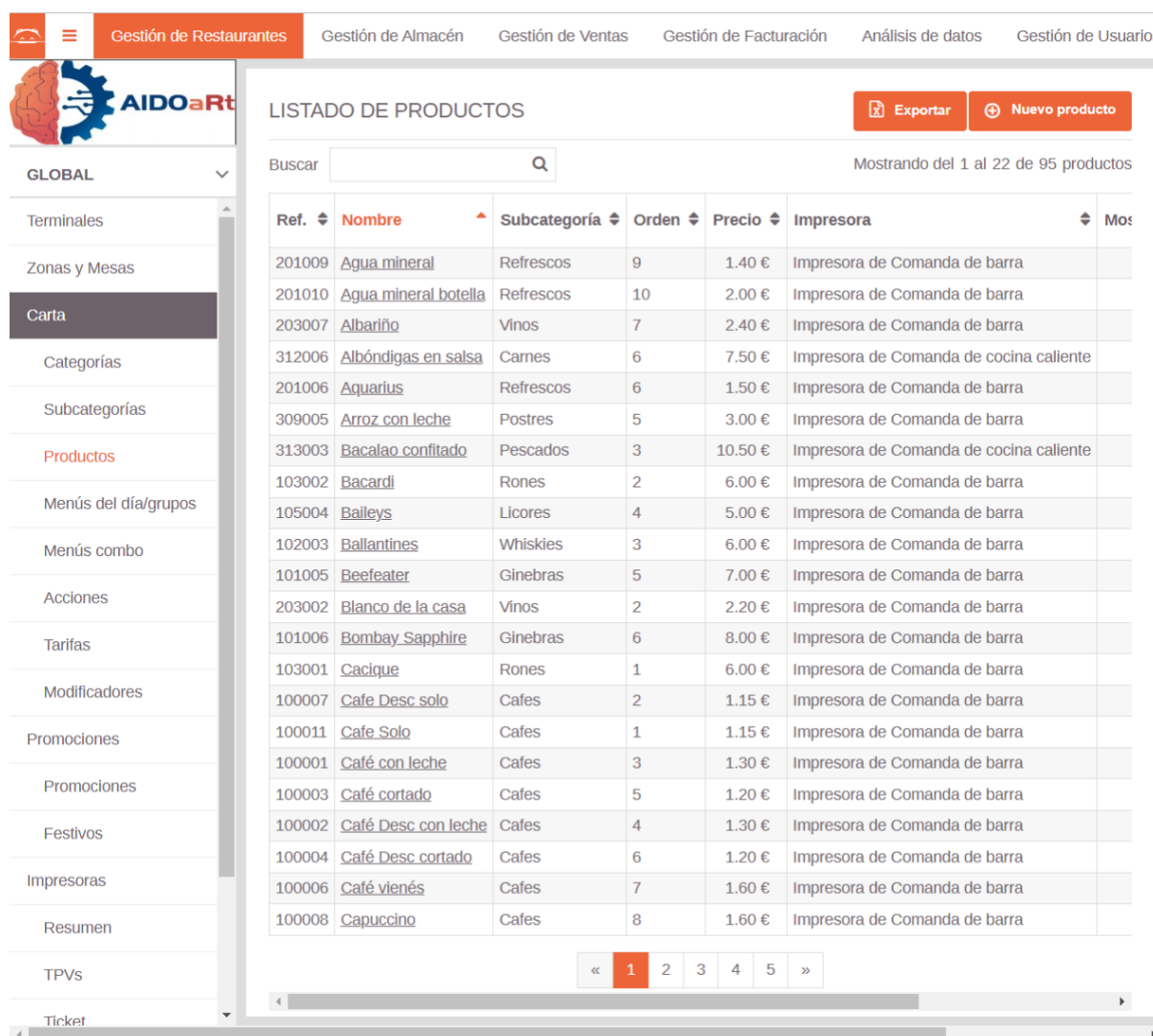
Challenges

The challenges in this UCS are:

- Acquisition of source data from log files of various assets in the system (from the main application but also from the OS for the gateway and cloud server, system logs from hardware devices such as smartphones used as order taking device, cash registers, etc.).
- Homogenization of the different kinds of log files into a unified format (comprising, for example, application log traces, human generated comments, developer notes, etc.)
- Process all of this data with a variety of AI tools including text processing tools, NLP toolchains, etc.
- Organising the process of periodically performing the data acquisition, analysis and generation of reports.

7.2.1. Summary of preliminary results

In this Use Case scenario much progress has been done in establishing a proper Text Analytics engine that reads the aggregated logs from our test system at <http://aidoart.tamus.io>, as depicted in Figure 7.1:



The screenshot shows the 'LISTADO DE PRODUCTOS' (Product List) interface in the AIDOaRt system. The interface includes a navigation menu on the left with options like 'Terminales', 'Zonas y Mesas', 'Carta', 'Categorías', 'Subcategorías', 'Productos', 'Menús del día/grupos', 'Menús combo', 'Acciones', 'Tarifas', 'Modificadores', 'Promociones', 'Festivos', 'Impresoras', 'Resumen', 'TPVs', and 'Ticket'. The main content area displays a table of products with the following columns: Ref., Nombre, Subcategoría, Orden, Precio, Impresora, and Mos. The table contains 22 rows of product data. At the top right of the table area, there are buttons for 'Exportar' and 'Nuevo producto'. A search bar is located above the table, and a status indicator shows 'Mostrando del 1 al 22 de 95 productos'.

Ref.	Nombre	Subcategoría	Orden	Precio	Impresora	Mos
201009	Agua mineral	Refrescos	9	1.40 €	Impresora de Comanda de barra	
201010	Agua mineral botella	Refrescos	10	2.00 €	Impresora de Comanda de barra	
203007	Albariño	Vinos	7	2.40 €	Impresora de Comanda de barra	
312006	Albóndigas en salsa	Carnes	6	7.50 €	Impresora de Comanda de cocina caliente	
201006	Aquarius	Refrescos	6	1.50 €	Impresora de Comanda de barra	
309005	Arroz con leche	Postres	5	3.00 €	Impresora de Comanda de barra	
313003	Bacalao confitado	Pescados	3	10.50 €	Impresora de Comanda de cocina caliente	
103002	Bacardi	Rones	2	6.00 €	Impresora de Comanda de barra	
105004	Baileys	Licores	4	5.00 €	Impresora de Comanda de barra	
102003	Ballantines	Whiskies	3	6.00 €	Impresora de Comanda de barra	
101005	Beefeater	Ginebras	5	7.00 €	Impresora de Comanda de barra	
203002	Blanco de la casa	Vinos	2	2.20 €	Impresora de Comanda de barra	
101006	Bombay Sapphire	Ginebras	6	8.00 €	Impresora de Comanda de barra	
103001	Cacique	Rones	1	6.00 €	Impresora de Comanda de barra	
100007	Cafe Desc solo	Cafes	2	1.15 €	Impresora de Comanda de barra	
100011	Cafe Solo	Cafes	1	1.15 €	Impresora de Comanda de barra	
100001	Café con leche	Cafes	3	1.30 €	Impresora de Comanda de barra	
100003	Café cortado	Cafes	5	1.20 €	Impresora de Comanda de barra	
100002	Café Desc con leche	Cafes	4	1.30 €	Impresora de Comanda de barra	
100004	Café Desc cortado	Cafes	6	1.20 €	Impresora de Comanda de barra	
100006	Café vienés	Cafes	7	1.60 €	Impresora de Comanda de barra	
100008	Capuccino	Cafes	8	1.60 €	Impresora de Comanda de barra	



Figure 7.1 Deployed AIDOaRt version of TAMUS with a sample list of available products for purchase in the Menu ('Carta' in the Spanish original) connected with their sub-categories, price and routed printer for ordering ('Impresora')

The deployed system simulates a number of restaurants under the AIDOaRt demonstrator umbrella. Some of these simulated restaurants have access to physical devices such as cash registers, doors in cupboards for ingredients (used for inventory tracking) and other elements.

For the analysis of logs, however, we found the deployed prototype a bit restrictive since it is not representative of a working system in terms of log generation (the system mainly sits idle). We had thus two opportunities: to use a simulation to generate logs on the deployed system or to use system logs for a physical location where TAMUS is actually used. We chose the latter as using simulations to generate synthetic data that then is analysed using AI defeats the purpose of the scenario.

For that, we chose to evaluate the results of AI analysis on the logs by the real TAMUS system. For the evaluation testbed, logs for 6 consecutive days in April 2023 were acquired. This is representative of the real system operation, as logs are collected in large batches and then analysed by humans manually.

```

[0mPOST /api/v1/UpdateClose [32m200 [0m5.690 ms - 15[0m
[08-04-2023 03:32:24][app.js:223] No Session - /api/v1/getTamusChanges
[0mPOST /api/v1/getTamusChanges [32m200 [0m2.825 ms - 57[0m
[08-04-2023 03:32:24][app.js:223] No Session - /api/v1/getTamusChanges
[0mPOST /api/v1/getTamusChanges [32m200 [0m2.705 ms - 58[0m
[08-04-2023 03:32:25][app.js:223] No Session - /api/v1/UpdateClose
[08-04-2023 03:32:25][v1.js:132 - updateClose()] CLOSEINFO: Company: 11, Restaurant: 33, BBDD: tamus-catering45 --> Total: 1241.25, Pending: 0.00, Customers
(Curr-Tot): (0-324), Closed: true
[0mPOST /api/v1/UpdateClose [32m200 [0m4.774 ms - 15[0m
[08-04-2023 03:32:26][app.js:223] No Session - /api/v1/getTamusChanges
[0mPOST /api/v1/getTamusChanges [32m200 [0m2.780 ms - 58[0m
[08-04-2023 03:32:26][app.js:223] No Session - /api/v1/pingAlive
[08-04-2023 03:32:26][v1.js:308 - pingAlive()] Ping alive from ECR 790039810 to tamus-lateral database. [OK]
[0mPOST /api/v1/pingAlive [32m200 [0m5.676 ms - 44[0m
[08-04-2023 03:32:27][app.js:223] No Session - /api/v1/pingAlive
[08-04-2023 03:32:27][v1.js:308 - pingAlive()] Ping alive from ECR -732953433 to tamus-besthotel database. [OK]
[0mPOST /api/v1/pingAlive [32m200 [0m12.391 ms - 44[0m
[08-04-2023 03:32:27][app.js:223] No Session - /api/v1/pingAlive
[08-04-2023 03:32:27][v1.js:308 - pingAlive()] Ping alive from ECR 679218314 to tamus-stjeremis database. [OK]
[0mPOST /api/v1/pingAlive [32m200 [0m24.281 ms - 44[0m
[08-04-2023 03:32:27][app.js:223] No Session - /api/v1/pingAlive
[08-04-2023 03:32:27][v1.js:308 - pingAlive()] Ping alive from ECR 400770897 to tamus-lateral database. [OK]
[0mPOST /api/v1/pingAlive [32m200 [0m5.670 ms - 44[0m
[08-04-2023 03:32:27][app.js:223] No Session - /api/v1/UpdateClose
[08-04-2023 03:32:27][v1.js:132 - updateClose()] CLOSEINFO: Company: 57, Restaurant: 2, BBDD: tamus-stjeremis --> Total: 4545.84, Pending: 0.00, Customers
(Curr-Tot): (0-541), Closed: true
[0mPOST /api/v1/UpdateClose [32m200 [0m5.386 ms - 15[0m
[08-04-2023 03:32:28][app.js:223] No Session - /api/v1/pingAlive
[08-04-2023 03:32:28][v1.js:308 - pingAlive()] Ping alive from ECR 1326098166 to tamus-lateral database. [OK]
[0mPOST /api/v1/pingAlive [32m200 [0m5.305 ms - 44[0m
[08-04-2023 03:32:28][app.js:223] No Session - /api/v1/getTamusChanges
[0mPOST /api/v1/getTamusChanges [32m200 [0m3.433 ms - 57[0m
[08-04-2023 03:32:28][app.js:223] No Session - /api/v1/pingAlive

```

Figure 7.2 Excerpt from TAMUS logs from 08 April 2023

The main objective of the AI analysis of the logs for now is to analyse the temporal behaviour of the devices connected to the TAMUS server. This is usually tracked using a simple keepAlive() RESTful call. Logs from these calls can be appreciated highlighted in the Figure 7.2. The system as of today is 'dumb', it only knows if the keepAlive() calls are made by the devices and received by the server. In case there is a disruption, it is a signal that the system is not connecting properly and an alert is raised for manual inspection.

In the HIB_UCS1 we want the system to improve its operational intelligence by providing a more nuanced analysis of these keepAlive() calls. That way, not only major disruptions can be detected but



also signs of future ones (e.g., if short interruptions pile up, if interruptions are consistently being longer or more frequent). Thus, we have decided to use AI for complex event processing here.

We use Python text tools to ingest the data and then the detection of failures is done using plain NumPy in addition to a layer of PySiddhi²³ for grouping together the individual instances of issues. This is then fed to a plot generating submodule that builds plots to be interpreted by humans. As a result of this analysis, we have obtained the following plot shown in Figure 7.3 of connectivity for devices in mid April 2023 on the real TAMUS system.

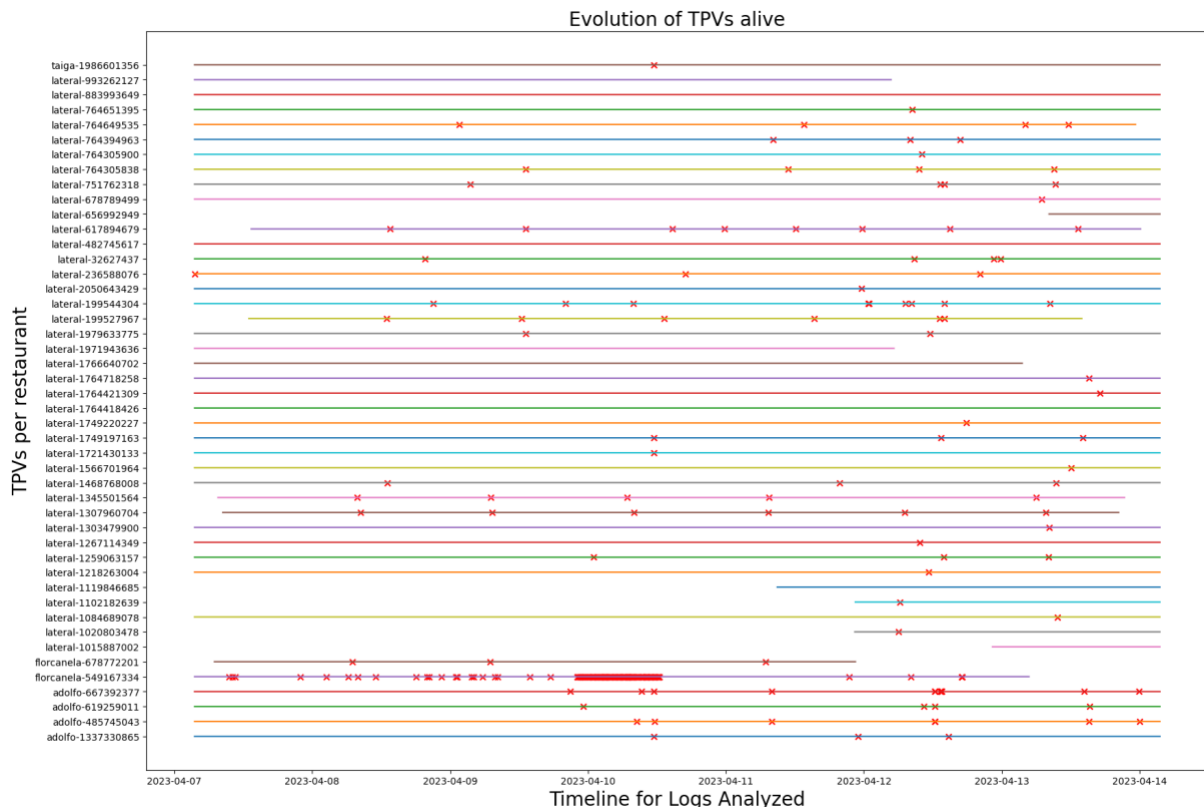


Figure 7.3 TPV (point of sale terminal) evolution of connectivity over April 2023 for the commercial TAMUS deployment. Explanation of the figure in the text body.

In the Figure 7.3, disruptions of connectivity are marked by crosses in the timeline for each device. We can quickly see some interesting results, such as major disruption in device forcanela-549167334 (fifth from the bottom) with a sequence of minor faults on April 07, 08 and 09 which lead to a catastrophic failure on April 10th that lasts for hours.

The catastrophic failure can be easily detected using traditional analysis, but the lead-up to it in the prior days is more interesting as it enables us to build an AI model that detects this build-up of minor issues automatically.

²³ PySiddhi Complex Event Processing: <https://siddhi-io.github.io/PySiddhi/>



7.2.2. Evaluation of results considering requirements coverage

The requirement upon which this UCS is built is quite generic:

HIB_R01: *Regarding the need to analyse, using AI, the produced logs generated by the system in order to detect anomalies in the operation of the system at runtime and anticipate points of failure.*

We can claim we cover the basics in this requirement as we can analyse the logs with ease now. The requirement can still be pursued however, as more automation can be extracted from it (the detection of build-ups, for example). This is a topic for the future of the work in AIDOaRt.

7.2.3. Evaluation of results considering KPIs

The relevant KPI for this UCS is as follows:

Table 7.2 KPIs HIB_UCS1

KPI Identifier: HIB_UCS_1_KPI_1.2_1		Scenario Identifier: HIB_UCS1	
KPI Description:	Reducing time spent on issues detection based on logs.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the early detection of system deviations	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> ≥ 30%
KPI Measure:	k = hours per week spent on analysing the logfiles for a single restaurant installation.		
KPI Baseline:	<i>Source:</i>	Internal calculations by the development team.	
	<i>Value: k₀ =</i>	30	
	<i>Value : k =</i>	10	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 66%

The detection of connectivity issues is the major aspect upon which the Operations team in HIB spends time manually checking logs. The usual time for this was around 30 minutes per day of logs generated by the system. With the current system as described, this has been reduced to around 10 minutes, thereby resulting in a reduction of around 66%. This can still be optimised by improving the detection of trends and buildup to failure as described above.

7.3. Use case scenario HIB_UCS2 — AI Requirements Management

In this UCS we manage the product requirements, stored using a Trello board for the TAMUS system. This is done to get (a) automatic classification of the new requirements according to the categories used by the development team and (b) suggested developers from the team according to their past performance with a specific time of implementation of requirements.

Challenges

The following are the challenges for HIB_UCS2:

- Connecting via the API to the relevant Trello boards and acquisition of the data contained in the boards into a usable format.
- Use of NLP to cluster the different topics of the Trello cards into those relevant for the development workflow.
- Use of Machine Learning to identify the most relevant developers to tackle a proposed task based on prior performance with similar tasks.
- Generation of reports for the product owner.

7.3.1. Summary of preliminary results

In the past months in AIDOaRt we have successfully integrated AI analysis tools (using NLP toolchains) with the Trello requirements management process used by the TAMUS developers. We can see a snapshot of the board in Figure 7.4 and a detail of a single requirement in Figure 7.5.

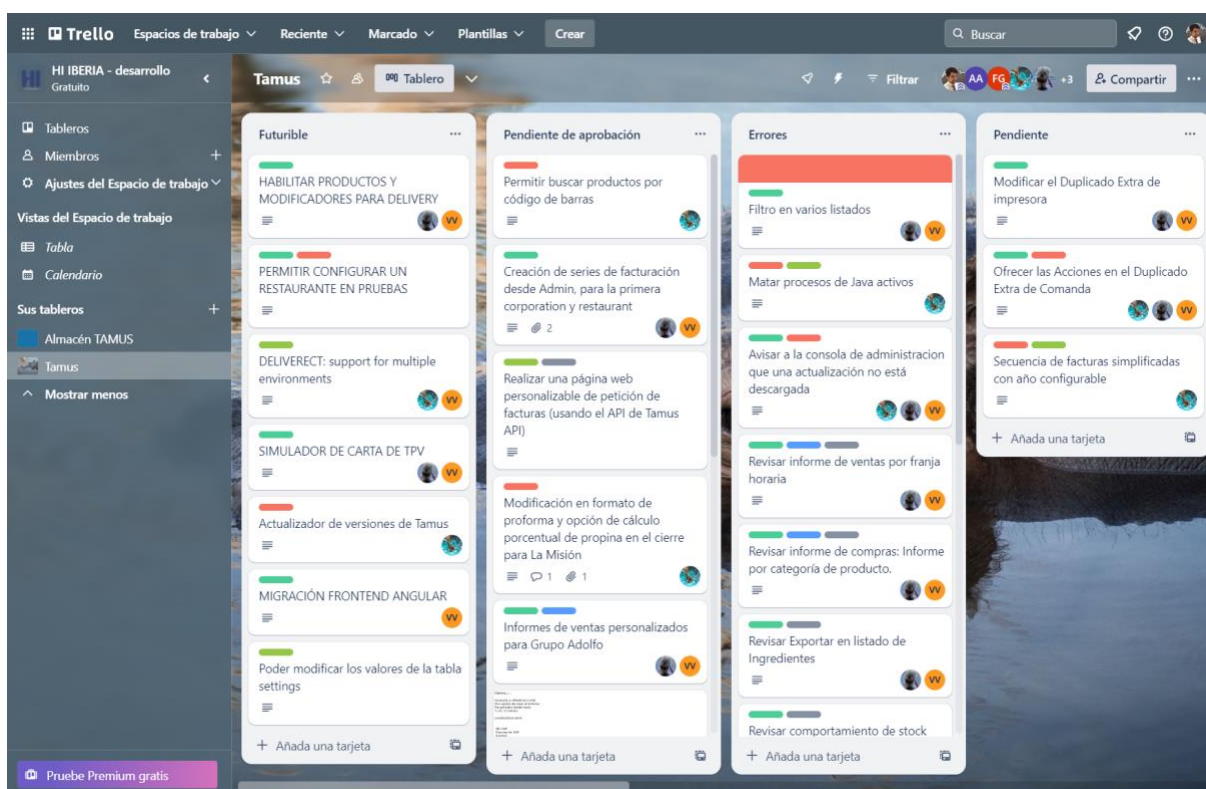


Figure 7.4 TAMUS Trello board with requirements in different stages of production (in the stacks such as 'Futurable': Future features, 'Errores': Errors and 'Pendiente': Pending. Each requirement is stored in a single card.

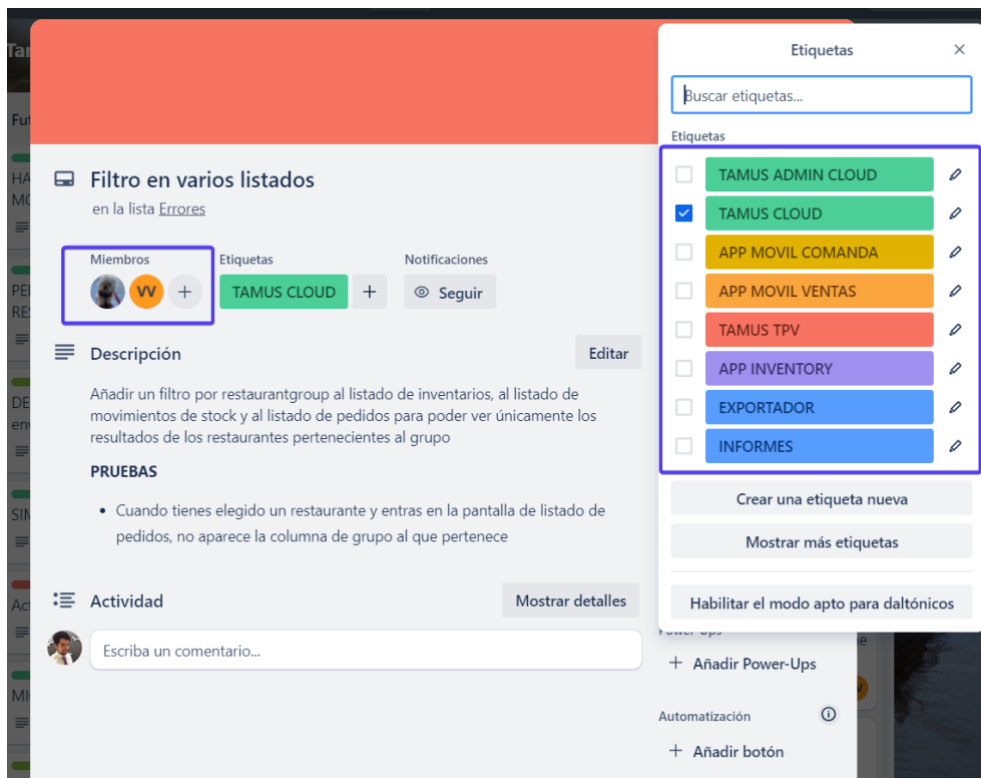


Figure 7.5 TAMUS Requirement in Trello (detail) with highlights on the assigned ‘tags’ or topics and the ‘members’ or assigned developers.

Footer: TAMUS requirement card with highlights on the possible ‘tags’ or topics and the ‘members’ or assigned developers.

The requirements are encoded by the product manager in natural language (in Spanish but mixed with much jargon and technical terms). The overall goals of the AIDOaRt system for now would be two:

- Detect the ‘topic’ and assign Trello ‘tags’ based on the relevant one(s).
- Assign the requirement to the most likely developer, based on past performance of said developer.

Since we are using a system based on manual inputs, we need to use AI tools that allow us to incorporate a certain aspect of ‘fuzziness’. For example, when reading the text of the requirement to detect the topic, any analysis using purely vanilla NLU tools is very likely to fail. We had then to do several layers of pre-processing to (a) remove unnecessary terms (language connectors, empty words), (b) analyse the topics based on word frequency including these jargon terms and mixed english-spanish words (e.g., ‘stock’ is not a Spanish word but can be seen in the text for the requirement depicted above) and (c) associate n-grams (vectors of consecutive words) with the assigned developers in the past because it is expected that their performance in the past can be of use in the future. In layman terms, and based on the requirement depicted above, it is expected that future requirements containing the terms ‘stock’, ‘inventario’ and ‘pedidos’ can be automatically pre-assigned to the tag ‘TAMUS-CLOUD’ and to the developer ‘VV’.

For this we used a Natural Language Processing engine built on Python and following a zero-shot learning proposed by RISE in the railway use case²⁴.

Using the tools in the standard NLP Python community, we came up with the following phases of analysis:

Phase 1 - Data cleaning:

Some typical processes applied such as: removing unwanted characters, i.e., special characters, converting all text to lowercase, text tokenization, removing stop words, i.e. 'and', 'the', etc. and getting the root of words (lemmatization).

Phase 2 - Data exploration:

A variety of techniques are used to get a better understanding of the captured data before running the AI algorithms on them. These include:

- Calculating basic statistics such as the number of documents, the average length of the documents and the most frequent words.
- Investigating label distribution to understand class balance and identify potential class imbalance issues, which would affect the training of our algorithms.
- Analysing the most common and unique words for each tag to identify distinctive patterns.
- Calculating the TD-IDF – inverse document frequency. This is the frequency of occurrence of the term in the collection of documents. It is a numerical measure that expresses how relevant a word is to a document in a collection.
- Trying graphical visualisations with the reduced dimensionality such t-SNE graphics with TD-IDF vs embedding.

These elements of analysis provide values for each of the words and tags in a high dimensionality mathematical model. With that in hand, the end objective is to understand how far away from each other are the data points representative of each tag. In our analysis we could see they are much overlapped, meaning that the tags are somehow representing overlapping concepts in the requirements analysis.

- The polygons (defined by the calculated points in the multidimensional space) are the different classes that are obtained using k-means clustering both for the description and in the cell below, for the requirements name/title. The points are the different descriptions for the case of descriptions and the different names/titles and in which group they fall. As mentioned above, we observed that the polygons defining our 'tags' were very much overlapping.
- In summary we observed that with traditional ML techniques it will be very difficult to assign the requirements to different tags, as there is no clear difference among them, due to the reduced number of data that we have that is translated in unbalanced datasets.
- So, finally, we tried a Deep Learning approach with the Zero Shot learning process currently being used. The reason is that Language Models need a lot of requirements to be properly trained, but with the Zero Shot we are able to train deep language models with just few requirements.
- The model to be implemented relies on a pre-trained ML backbone that is used to perform tokenization and create task sentence embedding and label task embedding. Then, the classifier processes the embedding results by computing the relatedness between the

²⁴ Bashir, Sarmad, et al. "Requirement or not, that is the question: A case from the railway industry." *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Cham: Springer Nature Switzerland, 2023.

sequences embedding and the label embeddings using cosine similarity. Finally, the overall similarity scores will be fed into a classification function, and the probabilities of all labels will be computed to select the maximum score as the most related labels to a given task. Following this process we arrived at a classifier that, given the text of the Trello card, can make a confident prediction of the topic discussed within. Using a similar process, we can make predictions upon who would be the most likely developer to be able to resolve it.

7.3.2. Evaluation of results considering requirements coverage

The original requirements for this UCS read as follows:

HIB_R02: *Covering the usage of AI to manage the system requirements, maintained in a hands-on fashion by the TAMUS team and requiring substantial manual work to be analysed. The objective of this requirement is to enable requirement classification and assignment to developers by means of AI analysis.*

We have achieved much of the AI development needed to fulfil this requirement. The remaining work required involves the automation of the process from a technical point of view (using the Trello API bidirectionally to get the newly produced requirements and then to assign tags and members to them). This is expected to be resolved in the coming months of AIDOaRt.

7.3.3. Evaluation of results considering KPIs

The relevant KPIs for this UCS were as follows:

Table 7.3 KPIs HIB_UCS2

KPI Identifier: HIB_UCS_2_KPI_1.1_2		Scenario Identifier: HIB_UCS2	
KPI Description:	Decrease of time spent in the assigning of tags to cards as compared to a human operator.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of 25% of the time required for identification of design problems thanks to the analysis of the collected data.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 25%
KPI Measure:	k = Time spent in total on requirements topic tagging by the product owner in minutes per week per installation of TAMUS.		
KPI Baseline:	<i>Source:</i>	Estimates of the current product owner.	
	<i>Value: k₀ =</i>	30	
	<i>Value : k =</i>	20	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 33%

Table 7.4 KPIs HIB_UCS2

KPI Identifier: HIB_UCS_2_KPI_1.1_3		Scenario Identifier: HIB_UCS2	
KPI Description:	Decrease of time spent in the assigning of tags to cards as compared to a human operator.		
Refined AIDOaRt KPI:	<i>Description:</i>	Precision of the assigning of tags to developers as compared to a human operator.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 25%

KPI Identifier: HIB_UCS_2_KPI_1.1_3		Scenario Identifier: HIB_UCS2	
KPI Measure:	k = Time spent in total on requirements assigning by the product owner in minutes per week per installation of TAMUS.		
KPI Baseline:	<i>Source:</i>	TAMUS product owner experience.	
	<i>Value: k₀ =</i>	45	
	<i>Value : k =</i>	45	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 0%

Table 7.5 KPIs HIB_UCS2

KPI Identifier: HIB_UCS_2_KPI_3.1_1		Scenario Identifier: HIB_UCS2	
KPI Description:	Coverage of the automatic system reading the Trello board in terms of cards fully processed (input into the system for analysis).		
Refined AIDOaRt KPI:	<i>Description:</i>	KPI 3.1: Automate a 30% of the processes which are currently manual (e.g., predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = Number of new tasks/cards issued per week.		
KPI Baseline:	<i>Source:</i>	Estimates of the TAMUS product owner.	
	<i>Value: k₀ =</i>	3	
	<i>Value : k =</i>	3	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 0%

For **HIB_UCS_2_KPI_1.1_2 - Decrease of time spent of the assigning of tags to cards as compared to human operator**, our baseline value was 30 minutes per week. With the automatic system used (although it is not fully integrated with Trello, as mentioned) we have reduced this to around 20 minutes, thus reducing the value up to 33%. This is very promising as it is expected that when the Trello automations are in place, the value will be even lower.

For **HIB_UCS_2_KPI_1.1_3 - Precision of the assigning of tags to developers as compared to human operator**, our stated baseline was 45 minutes per week. In this case, the developer selection by the system is not overly reliable for now, so the time spent remains unchanged. We expect to improve recall (the ability of the system to provide an answer on a query for a given tag) in the next iteration by providing the training of the AI models with more data.

Finally, for **HIB_UCS_2_KPI_3.1_1 - Coverage of the automatic system reading the Trello board in terms of cards fully processed (input into the system for analysis)**, the value was 3 automated readings from Trello per week. Since this is still not functional, the progress in this KPI is 0%.

7.4. Use case scenario HIB_UCS3 – AI-enabled new versions of assets analysis

In this UCS, we perform AI-enabled analysis of the newly developed components for TAMUS as new software bundles to install in the system for a given restaurant.

Challenges

The challenges in this UCS are as follow:

- Implementation of a CI/CD workflow in the development system for TAMUS, which currently relies mostly on human developers making all of the integration and deployment work.
- Make the system able to analyse the proposed new software versions of TAMUS proposed as 'bundles' and model them in an intermediate format.
- Detect anomalies in the proposed 'bundles' and generate reports accordingly for the product owner to take actions such as flagging the bundles for human inspection or rejecting the bundle.

7.4.1. Summary of preliminary results

This UCS was not fully tackled during this period due to one key collaboration with AND being delayed around M18 of the project. After that, no suitable alternative could be found within the consortium so we are planning to tackle it with a mixture of off-the-shelf DevOps tools and dedicated HIB developments. However, the current progress is limited.

7.4.2. Evaluation of results considering requirements coverage

For the reasons stated above, the requirement is still not covered.

***HIB_R03:** Focusing on the packaging on new versions of the TAMUS system as 'software bundles' that can be then customised and sent to their intended restaurants of usage. AI is here used to validate each bundle before delivery.*

The requirement remains nonetheless relevant and some preparatory work has been undertaken to improve this implementation in the next period.

7.4.3. Evaluation of results considering KPIs

No achievement means that the treatment of new versions in TAMUS is still very much in its initial state. The KPIs remain unchanged and 0% progress is achieved, although this is expected to be solved shortly after the finishing of this deliverable.

7.5. Use case scenario HIB_UCS4 – Deployment and analysis of new HW/SW versions

In this use case scenario, we perform and analyse the results of automatic updates to the code for the POS system including AI for compatibility and issue tracking. Aspects related to CPS are strongly monitored (e.g. link with the physical layout of each one of the restaurants such as kitchen, tables, etc., as well as the integration with hardware elements such as cash registers and printers).

Challenges

The challenges in this UCS are as follow:

- Analysis of the deployment process of the assets analysed at HIB_UCS3 and integration into the log analysis toolchain proposed at HIB_UCS1.

7.5.1. Summary of preliminary results

Same as with the connected HIB_UCS3, we are missing key CI/CD software that we couldn't find within the consortium. Thus, the major part of this work is delayed until the next period where we will integrate off the shelf tools with dedicated AI algorithms developed by HIB. We have focused during the period for this deliverable in cleaning up the definition of update software packages in TAMUS (known as 'bundles' and containing both compiled code and deployment instructions encoded as shell scripts) so that they can be ingested by the AI algorithms.

7.5.2. Evaluation of results considering requirements coverage

HIB_R04: *Closing the DevOps circle by analysing the deployment process. AI is used to get a better understanding of the results of deployment and integration in the physical location of the restaurant. This relates to several CPS aspects such as the particulars of the restaurant's layout, networking and connected hardware elements such as cash registers and locks.*

As per the prior requirement and UCS3, little progress has been achieved so the requirement still remains fully relevant and not covered.

The requirement remains nonetheless relevant and some preparatory work has been undertaken to improve this implementation in the next period, such as generalising the 'bundle' model as proposed in the previous subsection and providing a serialization strategy so that the metadata of the bundle can be transferred to the AI nodes without including the binary assets.

7.5.3. Evaluation of results considering KPIs

No achievement means that the treatment of new versions in TAMUS is still very much in its initial state. The KPIs remain unchanged and 0% progress is achieved, although this is expected to be solved shortly after the finishing of this deliverable.

7.6. Planned improvements

The planned improvements for the next cycle are as follows per UCS:

- In UCS1 we plan to improve the AI that predicts failure in connectivity as we have discussed in the relevant section. For this, we will use more time series analysis as available in the PySiddhi library that we have mentioned above. The overall goal is to be able to anticipate catastrophic connectivity issues that are preceded by minor deviations. This is expected to be fully operational in the next iteration of this work.
- In UCS2 we need to solidify the developer assignment as currently results in bad predictions that do not save any time to the product manager since they have to be revised in many instances. In addition, we plan to improve the end-to-end Trello automation by enabling the AI system to read from Trello and output its results on the board. This can be achieved by using the Trello API so it is expected for the next iteration.

- In UCS3 and UCS4 we have had to make considerable changes to the initial plan after a collaboration that did not succeed in providing all of the required CI/CD tooling. Thus, we have pivoted to design and implement an in-house system for AI-assisted assets inspector and deployment manager. For the first part it is still unclear how to use AI models to link lists of software assets in a bundle to expected operation. For the second, an evolution of the logs analysis used in UCS1 and also a planned collaboration with Westermo for generalised logs analytics is the current line of research.

7.7. Planned demonstration

It is expected that for future phases of the project at least Use Case Scenarios 1 and 2 can be fully demonstrated. Given that UCS2 has an explicit UI (the Trello board used for the requirements) that can be used as a visual example (e.g. the process of creating a requirement and automatically tagging and assigning it can be showcased), it might be the most viable demonstration as it can readily show the results of the processing.

8. PRO_CS07 case study “Smart Port Platform Monitoring (SPPM)”

8.1. Case Study description

The case study is based on an Industry 4.0 Big Data solution in charge of monitoring the activities of a port in real time, through the analysis of data from sensors (IoT) installed in the port's air quality stations, cranes, parking slots, ships and information systems (legacy and external systems). Thanks to this solution, port operators can better monitor in real time the productivity and problems of all the operations carried out in the ports. It also allows the analysis of historical data to review past productivity and problems as a forensic review.

Additional objectives/challenges would be the detection of specific or recurring bottlenecks in the port in order to detect them as quickly as possible, even predicting them in advance if possible, since these bottlenecks are associated with a loss of productivity and, consequently, a greater economic loss. Therefore, a correct definition of the architecture for processing the large amount of information produced (big data platform) and the necessary customisation for each port and terminal (automation of customised deployments with different cloud providers) would be desirable, since the correct sizing of an infrastructure avoids oversizing and, therefore, reduces monthly operating costs.

On the other hand, monitoring the running platform is another important aspect to ensure that the information is processed correctly and then all this information collected by the system can be used to determine anomalies and behavioural patterns to predict future problems, and even suggest infrastructure resizing to improve the quality and availability of the solution.

The main challenges of SPPM Use Case are: (1) the definition of a customizable infrastructure independently of the cloud provider; (2) improve the validation of the platform provisioning; (3) monitoring the platform and detect anomalies in the infrastructure and data gathering, and (4) predict future problems and try to automatically correct them. The *Case Study Synopsis* is in Table 8.1.

Table 8.1 Synopsis of the case study Smart Port Monitoring Platform

Use case scenario PRO_UCS1— Infrastructure as code (IaC)	
Description:	We want to use a mechanism for defining the deployments that is easily adaptable to each new project, for this we will use IaC languages for the definition of the infrastructure to be deployed.
Requirements:	PRO_R01- IaC Deploy different providers,
Tools:	Terraform, PIACERE-IDE (PRO)
KPI:	PRO_UCS_1_KPI_4.1_1
Use case scenario PRO_UCS2— Automatic validation of the platform provisioning	
Description:	Develop a tool that ensures the correct deployment of the infrastructure automatically.
Requirements:	PRO_R04- Test infrastructure

Tools:	TATAT (PRO)
KPI:	PRO_UCS_2_KPI_3.1_1
Use case scenario PRO_UCS3 — Detection of anomalies of the platform performance	
Description:	This use case focuses on data monitoring and anomaly detection. This is a double objective: on one hand verify that the data collection is correct and on the other hand the use of AI techniques to detect possible anomalies.
Requirements:	<ul style="list-style-type: none"> • PRO_R05- Detect anomalies + SLA • PRO_R07- Monitor Platform
Tools:	<ul style="list-style-type: none"> • Position Monitoring for Industrial Environment (ACO), • AsyncAPI Toolkit (UOC), • Prometheus, Grafana
KPI:	PRO_UCS_3_KPI_1.2_1
Use case scenario PRO_UCS4— Detection and correction of service interruptions	
Description:	The objective of the scenario is to apply AI techniques, in order to automatically recover the system after a problem or malfunction.
Requirements:	<ul style="list-style-type: none"> • PRO_R06- Predict demand • PRO_R07- Monitor platform
Tools:	Position Monitoring for Industrial Environment (ACO)
KPI:	<ul style="list-style-type: none"> • PRO_UCS_4_KPI_1.1_1 • PRO_UCS_4_KPI_3.1_1
Use case scenario PRO_UCS5 — Resizing of resources based on current workload	
Description:	In this use case, we try to prevent future problems by applying prediction techniques.
Requirements:	<ul style="list-style-type: none"> • PRO_R06- Predict demand • PRO_R08- Self Learning / Self Healing
Tools:	<ul style="list-style-type: none"> • ak2-modev (ITI), • ak2-depman (ITI)
KPI:	PRO_UCS_5_KPI_4.2_1

8.2. Use case scenario PRO_UCS1 — Infrastructure as code

First scenario is the automatic deployment of the solution based on Infrastructure as a Code (IaC). At the beginning of the project, the normal flow of the deployment is as follows: first of all, the architecture and the components/nodes of the platform are estimated, then the Dev team prepare the solution based on the architecture definition and a DevOps specialist define a Amazon Web Services (AWS) IaC of the solution and the specialist deploy the solution in AWS adding some monitoring tools of the platform using AWS solutions.

8.2.1. Summary of preliminary results

Before the AIDOaRt project, PRO did not have any experience in doing deployments using IaC. The deployments were done fully manual using the web Application of AWS without using any automation. To carry out the manual deployment, a scheme with the necessary infrastructure to host the application was created, security aspects were considered in this initial deployment.

The templates were manually rewritten in the Terraform²⁵ format. Some Terraform files that deploy the entire infrastructure to receive, analyse and monitor the data that has been collected from maritime sensors have been made (completely manually). Moreover, Terraform can be configured to deploy the Docker composition in an AWS EC2 instance²⁶, with specific characteristics.

Therefore, we can deploy the entire infrastructure with ease using less time than at the beginning. This reduces the costs of the deployment significantly (50%). The deployment has been 70% automated using Terraform templates.

8.2.2. Evaluation of results considering requirements coverage

- **PRO_R01 - Use an Infrastructure as Code Language able to deploy the solution in different cloud providers and using different architectures / approaches (Containers & virtual machines)**

From a requirement point of view (defined in D1.1 [1]), it can be said that 75% of the requirements have been covered because with the use of Terraform, it is easier to adapt deployments to different service providers. However the ultimate goal is to use MDE (Model Driven Engineering) to define our Use Case requirements and automate 100% of the deployment. Also, we are trying some of the tools developed in another European project called PIACERE (<https://piacere-project.eu/>) to optimise the creation of the Terraform templates. In particular, the tools specified in the development phase of the solutions will be used:

- **DOML: DevOps Modelling Language**, it allows modelling the automation of the whole lifecycle of DevSecOps activities.
- **PIACERE IDE**: it is an integrated development environment (IDE) to develop infrastructural code that it will unify the automation of the main DevSecOps activities and will shorten the learning curve for new DevSecOps teams.
- **ICG: Infrastructural Code Generator**, it translates DOML specification into source files for different existing IaC tools. In this particular UCS , the target language will be a Terraform specification.

²⁵ <https://www.terraform.io/>

²⁶ https://aws.amazon.com/ec2/?nc1=h_ls

8.2.3. Evaluation of results considering KPIs

From a KPI point of view, the objective has already been achieved as the target was to reduce the 100 hours needed to prepare a deployment manually and now the time has already been reduced from 100 hours (on average) to 61. This implies that there has been a significant improvement in productivity in the percentage range in the DevOps process as set out in the KPI target for this scenario (Table 8.2).

Table 8.2 Case study KPI “PRO_UCS_1_KPI_4.1_1”

KPI Identifier: PRO_UCS_1_KPI_4.1_1		Scenario Identifier: PRO_UCS1	
KPI Description:	Improve productivity by reducing the effort needed in the DevOps process, by automatizing the deployment process using IaC. For this KPI, the time to develop an infrastructure deployment will be measured in hours.		
Refined AIDOaRt KPI:	<i>Description:</i>	Productivity improvement in the range of the percentage in the DevOps process.	
	<i>Identifier:</i>	KPI_4.1	<i>Target</i> ≥ 30%
KPI Measure:	k = percentage of improvement taking a totally manual development as reference		
KPI Baseline:	<i>Source:</i>	100 hours to develop a new infrastructure manually.	
	<i>Value: k₀ =</i>	100	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%
D5.6 Measure:	<i>Value: k₆ =</i>	70	$100 \cdot \Delta_6 / k_0 = 100 \cdot (100 - 70) / k_0 =$ 30%
D5.7 Measure:	<i>Value: k₇ =</i>	66	$100 \cdot \Delta_7 / k_0 = 100 \cdot (100 - 66) / k_0 =$ 34%

8.3. Use case scenario PRO_UCS2 - Automatic validation of the platform provisioning

At the beginning of the project each deployment had to be carried out by the operations team and this required a lot of time. The aim of this scenario is to ensure that the infrastructure is automatically evaluated and tested after each deployment.

8.3.1. Summary of preliminary results

As commented in previous deliverables, TATAT tool contributes to the use case scenario PRO_UCS2, where the challenge is to improve the validation of the SPMP after each deployment. TATAT is a solution that orchestrates the execution of automatic tests and it can be used to validate the deployment of the SPMP. This tool allows once defined a set of automated tests to validate the different configurations of the SPMP (using any test automation tool that allows test execution through a command line or API). These tests can be run on demand after each deployment by TATAT, taking into account the specific configurations of the deployment.

The TATAT tool is completely ready to use with Cucumber (<https://cucumber.io/>) and Selenium (<https://www.selenium.dev/>), and bash scripts.

8.3.2. Evaluation of results considering requirements coverage

- **PRO_R04 - Automatically test the architecture (infrastructure tests)**

Taking into account the requirements (defined in D1.1 [1]), most of them have been covered with the TATAT tool, as it allows to check if the defined infrastructure has been deployed correctly by launching a series of tests that the user must define beforehand to check that everything is correct. The only thing missing is the execution of a larger number of tests, which will be carried out in the following months.

8.3.3. Evaluation of results considering KPIs

As mentioned in the previous point, there is still more testing to be done with real scenarios to cover the target set in the KPI (Table 8.3). The objective is to "Increase in the percentage of the automated parts of the processes which are currently manual" and this has already been achieved because previously there were no automated checks and now there are, but it remains to be tested how many of these checks that were previously done manually can be automated.

Table 8.3 Case study KPI "PRO_UCS_2_KPI_3.1_1"

KPI Identifier: PRO_UCS_2_KPI_3.1_1		Scenario Identifier: PRO_UCS2		
KPI Description:	Increase in the percentage of automated tests when validating the infrastructure after each redeployment (at first the validation is made manually). For this KPI, the % of automated tests out of the total number of tests will be measured.			
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual.		
	<i>Identifier:</i>	KPI_3.1	<i>Target</i>	≥ 30%
KPI Measure:	k = percentage of automated tests versus total number of tests required to validate the deployment of the platform			
KPI Baseline:	<i>Source:</i>	0, no tests were automated at the beginning of the project		
	<i>Value: k₀ =</i>	0%		
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0)$	≥	30%
D5.6 Measure:	<i>Value: k₆ =</i>	1%	$100 \cdot \Delta_6 / k_0 = 100 \cdot (0.01 - 0)$	= 1%
D5.7 Measure:	<i>Value: k₇ =</i>	10%	$100 \cdot \Delta_7 / k_0 = 100 \cdot (0.1 - 0)$	= 10%

8.4. Use case scenario PRO_UCS3 - Detection of anomalies of the platform performance

In complex scenarios like smart ports, it is essential to understand the environment's time evolution. The goal of this use case scenario consists of finding problems in the collected data and ensuring that all devices linked to the Smart Port Platform are behaving as agreed.

8.4.1. Summary of preliminary results

Two partners are involved in this UCS3. Firstly, UOC provides a tool based on the AsyncAPI specification to formalise and (semi)automate the design and implementation of APIs for message-driven architectures, ensuring its data integrity in terms of syntax correctness. In this scenario, a definition of the architecture (including components, messages, and topics) from the ports and the AIDOaRt framework was created using the graphic modeller plug-in in the AsyncAPI Toolkit. As a result of the model-to-text transformations that the tool provides, more than 9,000 lines of API code were automatically generated from the model. Additionally, a quality of service agreement has been defined as a set of service-level objectives which have been integrated in the generated code as JSON information. These JSON definitions have been validated against an extension of the AsyncAPI specification aimed to define the quality of service for asynchronous messaging, which tailors the ISO/IEC 25010 quality model and the key concepts of WS-Agreement [17].

The second collaboration is with ACO which is contributing by investigating, developing and evaluating a generic anomalies analysis (GAA) solution applied on time series, which is expected that can be eventually applied at two levels:

- On signals monitored at an industrial plant, i.e. on a maritime port in this use case.
- On execution traces at embedded-level, specifically, on traces from runs of virtual models based on virtual platforms (VP), e.g. a virtual model of a PIoT device.

From D5.6 [9], up to now, relevant progress has been made. A main focus has been done on developing the edge monitoring infrastructure. So far, a former version of the gateway for enabling monitoring and analysis at operation time (AIOps) has been implemented and demonstrated (at past Västerås plenary meeting). This gateway can run the required services to read sensor data from an Ethernet port, to dump them on an edge time series database (Influx) and to run initial versions of the Generic Anomalies Analysis (GAA) investigated. This gateway is a main element of the edge infrastructure for sensor data gathering (positioning data among it), as it connects sensors to the cloud, and thus enabling remote monitoring, analysis and management. Relying on this gateway, and in direct relation to UCS3, a former demo of detection on a non-straightforward anomaly on load-unload operation of a fixed crane (e.g. STS) has been run. The load-unload operation data was synthetically generated, considering value ranges and feedback from Prodevelop. At this phase of the development, this is being beneficial for understanding the automated learning possibilities and limits of the DL-based analysis, and moreover to ensure a ground truth in terms of “real anomalies” which enable a more quantitative validation.

8.4.2. Evaluation of results considering requirements coverage

- **PRO_R05 - Detect automatically Anomalies in the solution during the execution based on AI**
- **PRO_R07 - Monitor the platform in real time to reduce the downtime and the data lost**

Taking into account the coverage of the requirements (defined in D1.1 [1]), it cannot yet be stated that they are fully met. However, the evolution of the model being built to monitor and automatically detect anomalies in the quality of service is progressing as expected, and will allow us to comply with requirement PRO_R07, and additionally enable PRO_R05 (section 4.7, deliverable D1.1 [1]).

Regarding requirement PRO_05, work has been done on enabling Generic Anomaly Analysis (GAA) at the edge and preliminary results advance a good line of progress in achieving the requirement, as already shown in the demonstration at the GA meeting in Sweden (Figure 8.1). There, a case of container loading and unloading operation showed the possibility to detect, not only explicit but also subtle anomalies.

Figure 8.1 shows Grafana panels of the monitored signals. On top of it, the combination of the 3 considered signals (load height, load operation time and load tension) is shown. Below, each of them is represented in its own panel. The GAA yields a set of anomaly intervals, which are represented by their start and end boundaries (dotted red lines). The core DL-based anomaly detection performs an anomaly inference per sample. A clustering algorithm groups detected anomaly samples into anomaly intervals, relying on the presumption that closer anomaly detections are likely related by the same anomaly origin.

It is important to highlight that in the presented test case, the detected anomaly intervals were reflecting unprecedented combinations of load stress when considering both, load weight and load-unload operation time, capable of causing stresses and an eventual breakage event. These were subtle anomalies in the sense that none of the monitored signals were out of nominal value, not even out of the historical ranges observed in the training.



Figure 8.1. Grafana-based visualisation of anomaly intervals in the load-unload operation detected after GAA ran on the gateway.

In addition, the gateway base load was measured (~25% CPU, 2G RAM, <11GB disk) for a benchmark considering a basic data transfer to the gateway. When running the GAA, the gateway consumed ~75% CPU, keeping RAM and disk consumption about the same, which can be assessed as an affordable load on top of the base monitoring infrastructure for the designed GW.

8.4.3. Evaluation of results considering KPIs

The final asset that is planned for this use case scenario is a monitoring dashboard capable of identifying violations of the quality of service agreements agreed upon to the parties involved in real time. So far, we have made progress in generating the definitions required for automatically generating

such a dashboard following a model-driven approach. The next steps include the MDE code for transforming the service level objectives defined in JSON format to a running dashboard. When finalised, we will be able to measure the number of anomalies automatically detected and contrast that figure against the total occurrences.

At the edge side, as reported on the previous section, synthetic data was injected and analysed in the edge monitoring infrastructure. Synthetic data enabled full control on the injection of synthetic anomalies on the test data sets. This way, a ground truth is available where it is known which load-unload operations are normal and which ones have anomalies, which in turn enables the quantitative evaluation of detection accuracy. This will be explained in more detail as follows: On top of Figure 8.2, the measured tension (load weight per gravity) is represented in red. Notice this is a measurable parameter that can be, and indeed was monitored and displayed in the Grafana dashboard (Figure 8.1). On the bottom of Figure 8.2, the relation (tension x operation time) is displayed (this is not a measured parameter), it is just a computation performed in the experiment that reflects the anomalies. The innovation is subtle in the sense that it does not provoke any of the monitored parameters to exceed any historical max/min value. However, the relation of two parameters (tension and operation time) exceeds in some cases (as designed in our experiment) the limit (max. value) for the historical values of the (tension x operation time) relation observed in training. Such a limit is reflected by the black horizontal line at the bottom of Figure 8.2. This anomaly is detected in most of the cases by ACO GAA solution as reflected on the light green intervals of Figure 8.2, which corresponds to the anomaly intervals of Figure 8.1. A main advantage of this synthetic data set is that the exact amount of subtle anomalies, i.e., the amount of times the green signal exceeds the black horizontal line threshold can be counted, which makes possible a quantitative evaluation of the confidence of the GAA for this case.

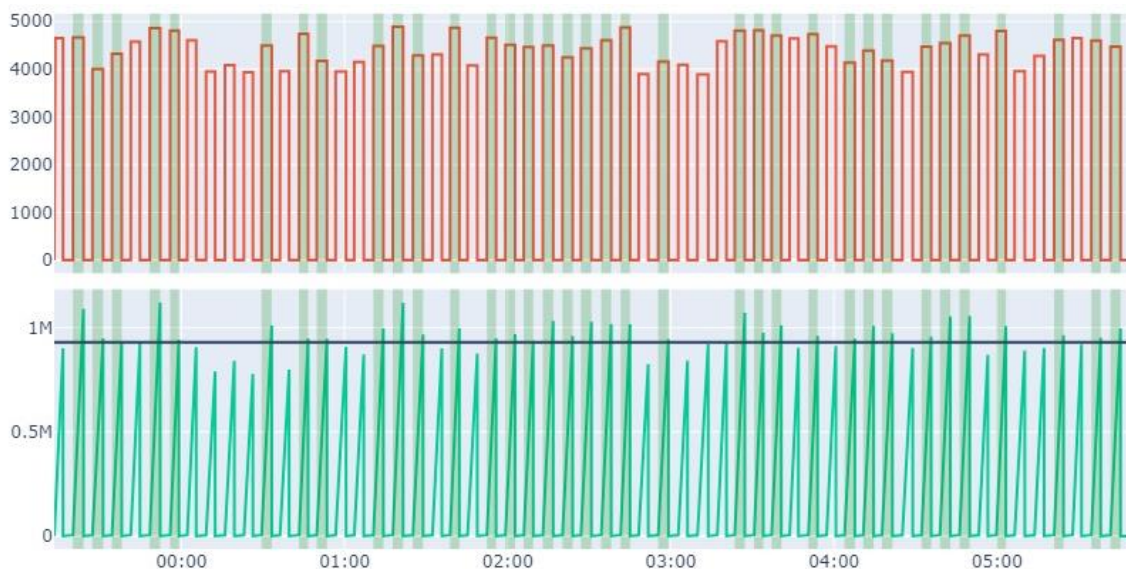


Figure 8.2. Evaluation based on synthetic data enabled accuracy evaluation of the GAA method.

Specifically, in the test demo shown in the third AIDOaRt plenary meeting in Västerås (Figure 8.2), 57 load/unload operations were simulated in a test where 35 of those load/unload operations were anomalous (due to the aforementioned subtle stress relation). This enabled us to compute the confusion matrix, and rate the accuracy of the method. For this case, the DL-based GAA method reached an optimum value, 81% accuracy in a test of 57 load/unload operations. This accuracy figure considers 35 correct anomaly detections (out of 57 operations) and 11 right normal load/unload operations (in a case where no false anomalies are detected).

A very interesting observation was that, when the configuration considered 100% of the training history as part of normality, it involved a degradation of the detection accuracy down to 40%. This was due to the high number of missed anomalies (34 out of 57). However, it was also found that, if the percentage of training data considered normal falls below certain level (e.g., 95%), then the accuracy is also remarkably degraded, e.g., down to 61% for the previous 95% (Figure 8.2), due to the increase in false positives (detection of false anomalies).

The following table (Table 8.4), shows the definition of the KPI, the target and the iterations that have been done for this particular scenario:

Table 8.4 Case study KPI “PRO_UCS_3_KPI_1.2_1”

KPI Identifier: PRO_UCS_3_KPI_1.2_1		Scenario Identifier: PRO_UCS3	
KPI Description:	Increase in the percentage of detection of anomalies in the infrastructure. For this KPI, the % on anomalies automatically detected out of the total number of anomalies will be measured.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated early detection of system anomalies.	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> ≥ 30%
KPI Measure:	k = percentage of automatically detected anomalies versus total number of anomalies and problems occurred during operation of the platform		
KPI Baseline:	<i>Source:</i>	Anomalies occurred during the operation of the platform but initially no anomalies are being automatically detected.	
	<i>Value: k₀ =</i>	0%	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0)$	≥ 30%
D5.6 Measure:	<i>Value: k₆ =</i>	1%	$100 \cdot \Delta_6 / k_6 = 100 \cdot (0.01 - 0) =$ 1%
D5.7 Measure:	<i>Value: k₇ =</i>	5%	$100 \cdot \Delta_7 / k_7 = 100 \cdot (0.05 - 0) =$ 5%

8.5. Use case scenario PRO_UCS4 – Detection and correction of service interruptions

In order to provide self-healing capabilities, it is needed that the problems are detected as soon as they occur, or even better, before they happen. This self-learning mechanism is related to the PRO_UCS_03 that is in charge of monitoring the system, which is a prerequisite of this scenario.

8.5.1. Summary of preliminary results

With regard to the positioning solution, several advances can be also reported. The positioning sensing requirements have been finally resolved. This was an important discussion, as the specific port element to be positioned was fixed (straddle carrier), and as a consequence, a diverse set of requirements, in terms of update rate, precision, environment protection, and presumed scenario facilities (e.g., powering, connectivity). Based on this, a former solution design is ready, composed of a base station device and a Positioning IoT (PIoT) per crane to be positioned. A first version of the hardware design of the base station and of the PIoT has been done. Software and application developments are ongoing.

When it comes to the application to the GAA at embedded-level, the advance done so far refers to having a tracing platform able to dump to a database, so that the GAA can be applied in a similar way as for the time-series at the distributed level (i.e., against the time series database and with similar formats). With regard to the D5.6 report [9], tracing capabilities from the VP-based virtual model have been extended, to support multi-level tracing at all the relevant levels, i.e., application, RTOS and HW VP levels. Moreover, these multi-level traces, previously dumped to CSV format, can be now dumped to a time-series database. Therefore, a former basis for the application of the GAA methods to embedded-level signal sets, explored so far for signal sets on the distributed-level, is settled.

8.5.2. Evaluation of results considering requirements coverage

- **PRO_R07 - Monitor the platform in real time to reduce the downtime and the data lost**

The work of ACORDE Position Monitoring for Industrial Environment contributes to PRO_R07 (defined in D1.1 [1]), in a way that not only contributes to the real-time monitoring of the position of the different elements working in a Smart Port, but also helps to reduce downtime and lost data because it is able to keep track of the position of the different vehicles working at the same time on the platform and send alerts as soon as one of them deviates from its normal behaviour.

Further testing, both in a laboratory and real-world environment, is still needed, but progress is promising.

8.5.3. Evaluation of results considering KPIs

As discussed in the previous UCS, using the tool developed by ACO, tests have been carried out with synthetic data (to control the gold standard). The results obtained with this experience, as in UCS3, also demonstrated the potential of the technology to anticipate/reduce the detection time of problems to potential problems ([Table 8.5](#) and [Table 8.6](#)). For example, in some of the tests performed,

the first anomaly is detected more than 5 hours before the final failure. Obviously, this depends on how the synthetic test with anomalies is designed, but the results are promising.

Table 8.5 Case study KPI “PRO_UCS_4_KPI_1.1_1”

KPI Identifier: PRO_UCS_4_KPI_1.1_1		Scenario Identifier: PRO_UCS4	
KPI Description:	Reduction of the time needed to detect a problem in the platform with the collected data. For this KPI, the time needed from the occurrence of the problem until the detection of it will be measured.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement (reduction) in the percentage of the time required for identification of problems in the platform (thanks to the analysis of the collected data).	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 25%
KPI Measure:	k = time reduction in minutes		
KPI Baseline:	<i>Source:</i>	current availability statistics (logs)	
	<i>Value: k₀ =</i>	360	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 25%
D5.7 Measure:	<i>Value: k₇ =</i>	$100 \cdot \Delta / k_7 = 100 \cdot (360 - 300) / 360$	20%

PRO_UCS_4_KPI_3.1_1

Table 8.6 Case study KPI “PRO_UCS_4_KPI_3.1_1”

KPI Identifier: PRO_UCS_4_KPI_3.1_1		Scenario Identifier: PRO_UCS4	
KPI Description:	Automate the detection and correction of system crashes (based on Self-learning and Self-healing techniques). For this KPI, the % of recurrent system crashes automatically solved will be measured.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of infrastructure crashes that can be automatically solved.	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = percentage of crashes automatically corrected versus total number of crashes occurred during operation of the platform		
KPI Baseline:	<i>Source:</i>	0% Non-automatic correction implemented	
	<i>Value: k₀ =</i>	0%	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 30%

Tests have not yet been carried out for the last KPI because the mechanisms for correcting system crashes are not yet in place. This phase is foreseen for a later stage when the automatic detection is fully polished and as reliable as possible.

8.6. Use case scenario PRO_UCS5 – Resizing of resources based on current workload

This use case focuses on preventing future problems from occurring by using modelling techniques to help predict the behaviour of the infrastructure.

8.6.1. Summary of preliminary results

The initial results achieved for UCS5 are as follows:

Developing **new modelling artefacts** to accurately represent the **dynamic behaviour** of the workload of the **smart port platform**. The smart port platform is composed by: (1) a centralised computing platform responsible for processing the messages coming from the IoT devices and the Terminal Operating System (TOS); and (2) hundreds of these IoT devices and Gateways that are deployed over the port infrastructures, mainly different kind of cranes, trucks and tractors.

The main challenges that have been addressed are twofold. On one hand, the IoT devices, and the corresponding workload they generate, are only active depending on the TOS assignments for a given ship arrival schedule. Modelling these activation intervals requires a **second level of scheduling** and a set of pseudo-activities that allows art2kitekt (a2k) tool suite to properly **simulate and evaluate the smart port behaviour** (Figure 8.3).

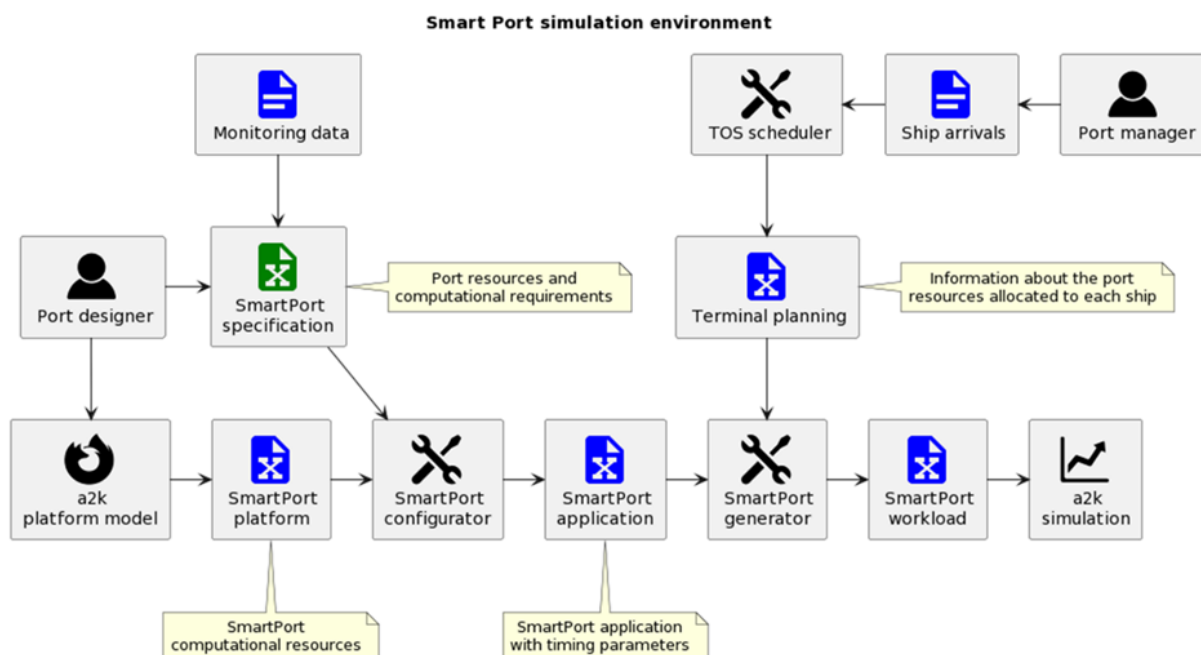


Figure 8.3 - Smart Port Simulation Environment.

On the other side, the application model required to represent the smart port platform is quite large, mainly due to the amount of IoT devices that are involved and the number of modelling artefacts that are needed to activate and deactivate the low-level activities when a ship arrives or departures. To

tackle the introduction of these application models and the new scheduling artefacts, a **tool for importing large models** based on simple spreadsheets has been developed. All the data required to build the model can be easily introduced by the UC provider in a custom spreadsheet that is then processed by this new tool and the resulting models are imported into the art2kitekt tool suite for its evaluation.

New algorithms for automatic placement of software services in a complex cyber-physical system (a2k-optimiser) have also been developed and researched. The system is represented as a graphical network of processing elements interconnected by communications links with various properties such as transmission bandwidth, latency, cost, etc. (Figure 8.4). This model can be generated from the methods described in the previous paragraphs. The software running on the processors is also represented as a graph which models the dependencies of each software service along with the resources it requires, such as its worst-case execution time. A set of optimisation objectives and constraints have been identified and defined to allocate the software services to specific processing nodes. These objectives include: power consumption, end-to-end latency, service spread, and resource usage, amongst others. Constraints include ensuring all the software tasks meet their required deadlines and that the processors are not overloaded. The a2k tool suite has been used to calculate system timing metrics (e.g., task response times) as well as simulation to obtain visual chronograms of multiple tasks sharing a computing resource and thus interfering with each other.

Several multi-objective optimisation solution methods are currently being evaluated, including, for example, genetic and evolutionary algorithms, simulated annealing, and particle swarm algorithms.

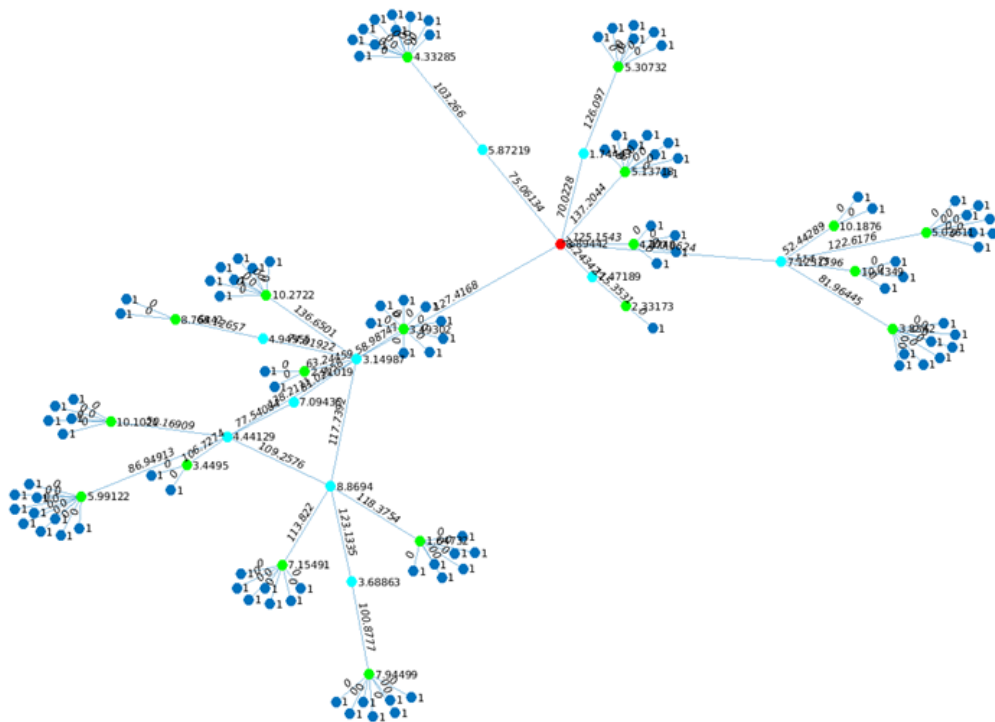


Figure 8.4 - Graph of Possible Smart Port Architecture showing processing nodes (blue: IoT devices, green: gateways, cyan: fog processors, red: cloud servers).

Developing **prediction methods** to help resize resources at the IoT device level: firstly, seven regression models have been evaluated and compared to determine which model can estimate different parameters with the best performance. The main idea is to estimate the optimal workload with the smallest error. With this purpose, the following regression models have been adjusted and compared: Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Support Vector Regression, Random Forest Regressor, and XGBoost Regressor. The XGBoost model showed the best performance and was selected for subsequent analysis.

Next, the selected regression model has been adapted for time series prediction (Figure 8.5). This way, it could also be used to predict the system's behaviour and recommend system mode changes before anomalies related to workload arise. The main idea is to apply the regression model using sliding windows that move forward through time. To do this, we compose each observation with information from n time instants (window size: t_n, \dots, t_2, t_1), which will be used to predict the next instant (t_{+1}).

Finally, the previous model has been modified so that it is possible to predict several time instants ($t_{+1}, t_{+2}, t_{+3} \dots$). This change has been studied in two ways:

1. Direct multi-output. The multiple-output regression problem is divided into multiple single-output regression problems. This strategy consists of fitting one regressor per target.

2. Chained multi-output. The multiple-output regression problem is divided into dependent single-output regression problems. In this strategy, each model makes a prediction in the order specified by the chain using all the available features provided to the model plus the predictions of models that are earlier in the chain.

All the algorithms have been developed using simulated data created by ITI. The necessary pipeline has also been developed to prepare the different models (Regression, Simple Prediction, Multiple Predictions) for their deployment.

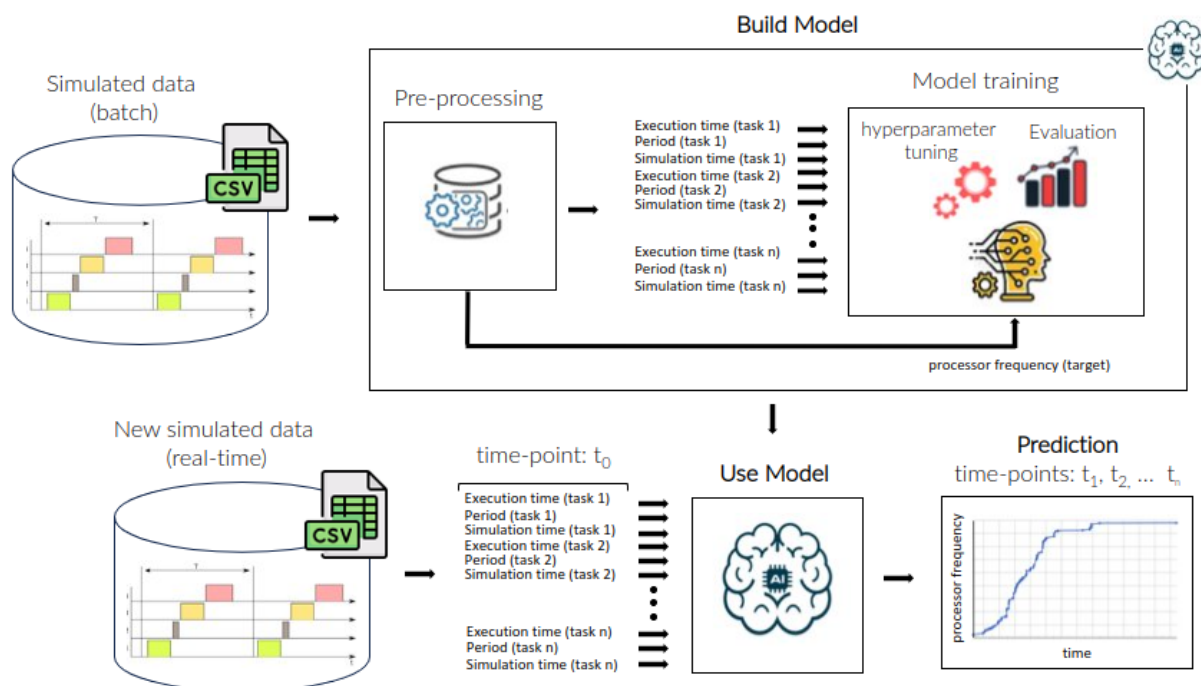


Figure 8.5 - Workflow used for the development and validation of prediction models.

This model is under refinement. The main purpose is to enable the possibility to predict the potential performance and enable feasibility assessment, or find needs for platform redimensioning, when the expected operational conditions for the gateway have to be changed, e.g., to support more sensor connections, different sensor bandwidths, more bandwidth with the cloud, or run heavier anomalies analysis.

8.6.2. Evaluation of results considering requirements coverage

- **PRO_R06 - Detect and predict high/low resources demand based on AI**
- **PRO_R08 - Self-healing and self learning solution to minimise the downtime of the platform by detecting and correcting the problems automatically. Avoiding the manual recovery of the problems**

The smart port computing architecture model allows for system timing analysis and simulation of the smart port architecture with different load demands. This approach is used to detect potential computational and communications bottlenecks. Therefore, it covers the requirement PRO_R06 - Predict Demand (defined in D1.1 [1]).

Regarding the development of prediction methods to help in the resizing of resources at the IoT level, the work carried out has been approached in two different ways:

1. On the one hand, the development of regression models to estimate the optimal workload from data corresponding to different system conditions. This work is trying to cover the case study requirement PRO_06 - Predict demand.
2. On the other hand, the development of prediction models that employ sliding windows to predict the system's behaviour and anticipate behaviour. In this way, future problems due to overload or lack of resources (PRO_06-Predict Demand). This prediction is intended to recommend system mode changes before anomalies arise, which is related to the case study requirement PRO_R08- Self Learning / Self-Healing.

8.6.3. Evaluation of results considering KPIs

Table 8.7 Case study KPI “PRO_UCS_5_KPI_4.2_1”

KPI Identifier: PRO_UCS_5_KPI_4.2_1		Scenario Identifier: PRO_UCS5		
KPI Description:	Automate the resizing of resources by developing a high-level model of the smart port computing architecture to modify the platform’s capability, anticipating future problems from occurring. For this KPI, we will anticipate % of future problems related to bottlenecks.			
Refined AIDOaRt KPI:	<i>Description:</i>	Quality Improvement in the percentage by improving predictability, conformance to specifications and proposal of system design refinements.		
	<i>Identifier:</i>	KPI_4.2	<i>Target</i>	≥ 30%

KPI Identifier: PRO_UCS_5_KPI_4.2_1		Scenario Identifier: PRO_UCS5	
KPI Measure:	k = Number of problems caused by excessive platform loading predicted before they occurred. This requires monitoring for such problems and checking whether the generated model had predicted them.		
KPI Baseline:	<i>Source:</i>	Non-automatic correction implemented	
	<i>Value: k₀ =</i>	Not yet calculated.	
Target:	Decrease	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	\geq 30%

In this section, preliminary results will be assessed with the KPIs (Table 8.7) for the Scenario-5 in this use case.

- Modelling, Load Simulation, Analysis & Optimisation:** We are working towards assessing the solutions using a slightly modified version of the performance index **PRO_UCS_5_KPI_4.2_1**. The goal is to eventually enable the operation of a system with zero potential for system crashes. The aim is to provide sufficient computing resources to the port architecture under the different loads expected throughout the day and night. The amount of “sufficient computing resources” will eventually be determined by our simulation, timing analysis, and optimization tools. The idea is to have a baseline platform architecture which is dynamically extended when an increased processing load demand is predicted by port handling requirements, and conversely reduced when a lower demand is predicted. While, at this stage, the exact performance metrics are still under investigation and development the baseline KPI will be calculated from a simulation of this static platform architecture under high demand conditions. Then the expected KPI improvement can be measured by simulation of the dynamic architecture algorithm.
- Prediction Methods:** Developing prediction methods to help in the resizing of resources at the IoT device level: one way to measure the system crashes automatically solved (**PRO_UCS_5_KPI_4.2_1**) is by developing models capable of estimating/predicting the required system resources with minimal error. The smaller the error, the more reliable the estimation/prediction. Consequently, the probability of avoiding failure increases.

To this end, the metric used to evaluate the performance of the developed regression and prediction models has been the R²-score, a popular metric for identifying model accuracy. It ranges from 0 to 1; the closer the value is to 1, the better the prediction and, therefore, the smaller the error and the greater the probability of avoiding failures. If R²-score equals 0, the model is not performing better than a random model.

So far, the developed models have been evaluated using a simulated dataset unrelated to the use case. However, new data already related to the use case is being simulated, so it is expected to adjust and validate the models developed with the latest data and see the error.

8.7. Planned improvements

- The roadmap for the AsyncAPI Toolkit encompasses the development of a real-time monitoring dashboard. This dashboard will be automatically generated from the quality of service agreement and its set of service-level objectives specified via the domain-specific language already provided. The dashboard will be capable of identifying violations of such objectives by design.
- Modelling and Load Simulation: It is planned to validate the system models using more accurate and realistic architecture data and parameter values (e.g., execution times, software dependencies, communications latencies, etc.) This plan will require close collaboration with the partners ITI, PRO, and ACO. The a2k analysis and simulation software will also have to be extended with some new modules to model the behaviours of the smart port. These will include, for example, components to represent novel artefacts of the port system (e.g., models for the ACO sensors and gateways, analysis and simulation of hierarchical schedulers, tasks for modelling messages held in queues, and tools for managing extensive models and multiple, possibly noisy, communications paths).
- Architecture Optimisation: It is planned to investigate in more detail the use of different multi-objective optimisation algorithms for the allocation of the software services to the processing nodes in the network. An important feature of multi-objective optimisation is that a solution is not a unique point in the search space. Rather, it is a set of potential solutions (the Pareto set). Therefore, a second stage of intelligent searching is needed to choose an appropriate solution from this Pareto set. Research is currently underway on how to do this in the context of the smart port. Further work is also needed towards closer integration of the analysis tools which are being developed. Currently we are using the internal a2k software tool suite to perform the modelling as well as the timing analysis and simulation activities. On the other hand, MATLAB²⁷ tools are being employed for optimisation and graphics output (*gamultiobj* toolbox). The intention is, once a good optimisation method is identified, to recode this directly into a2k.
- Prediction Methods: The regression and prediction models will be adjusted and validated using a new simulated dataset (IoT device level). On the one hand, to estimate the optimal workload from the data corresponding to specific system conditions and, on the other hand, to predict the optimised frequency for a real-time task set.

For the Generic Anomalies Analysis the planned improvements are:

- Extension of analysis to additional monitored signals, specifically positioning signals of straddle carriers. The analysis will be done first for synthetic data models, ranging from simple ideals to more realistic models, to assess the capabilities and limits in the GAA method for this scenario.

²⁷ <https://es.mathworks.com/products/matlab.html>

- Complete Implementation of positioning sensing solution (PIoT). The first goal is first to validate its integration with the gateway. Together the PIoT will complete the edge monitoring infrastructure.
- Complete second version of the gateway.
- Achieve a former analytical model of the gateway able to perform a basic feasibility analysis, e.g to assess which analysis loads can be run on the gateway.

Depending on the smooth run of previous tasks, and the efforts remaining, also expect to have the possibility for:

- Data retrieval in operational conditions. Use the PIoT developed in real operational conditions to retrieve actual 2D positioning data and assessment vs the synthetic evaluations.
- Have a former experience to apply the GAA method on traces from a virtual model of a positioning device.

8.8. Planned demonstration

At the end of the project, two demonstrators will be ready.

On the one hand, it will be delivered a demonstration of the a2k software tool applied to the smart port use case scenario 5. This demonstration will illustrate the following results:

- High- and low-level modelling, simulation, and timing analysis of the smart port architecture in different configurations.
- Design Space Exploration using multi-objective optimisation algorithms with non-linear constraints for the allocation of software modules to processing nodes under different system architectures.
- Machine Learning-based prediction algorithms for control of CPU clock frequencies with the objective of power aware-optimisation.

And on the other hand, it will present an extended version of the demonstrator shown at the last plenary meeting in Västerås. The new demonstrator will be able to:

- Demonstrate the applicability of GAA to additional types of data, i.e., to positioning traces monitored on mobile cranes (straddle carriers), which it will increase the diversity of monitored signals, and should serve to show the challenge to tackle such generality (in terms of number and types of signals) and specificity (port operation signals) at the same time
- Show a consolidated I/O interface for the GAA functionality, adding hints on the anomaly origin.

All this second demonstrator will be running on the edge device (gateway) designed and implemented by ACORDE, as was done in past Västerås demo.

Depending on the eventual availability of results of some parallel activities (2nd version of gateway platform, pilot data) and remaining effort for the demo, a more ambitious demo configuration will

target showing the implementation over the second optimised version of the gateway platform; and/or using pilot data, to demonstrate higher TRLs.

9. TEK_CS08 case study “Agile process and Electric/Electronic Architecture of a vehicle for professional applications”

TEKNE case study “Agile process and Electric/Electronic Architecture of a vehicle for professional applications” develops a cloud-enabled Prognostics and Health Management System (PHM). PHM functions deal with control, anomalies detection and classification, diagnostics, prognostics, and maintenance of the on-board power electronics of electric vehicles. The case study considers a traction inverter.

The case study deals with three aspects among those that TEK considers prominent for improving its industrial development process: design-time support to the modelling and to the models verification; automation of the run-time test preparation, execution, and results interpretation; AI/ML technologies and components.

Each of the above aspects is covered by a use case scenario, as summarised in Table 9.1.

Table 9.1 Synopsis of the case study TEK_CS08

Use case scenario TEK_UCS_01 — Design choices verification	
Description:	Design space exploration and verification of the models. Verification in a semi-automatic manner, at design time, with respect to the requirements, of the adequacy (the response versus the resources) of the real components on-which/with-which the system architect has in mind to map/realise the architecture.
Requirements:	TEK_R_102, TEK_R_103, TEK_R_104
Tools:	HEPSYCODE (UNIVAQ), S3D (UCAN)
KPI:	TEK_UCS_01_KPI_1.2_1
Use case scenario TEK_UCS_02 — Run-time verification	
Description:	Automatic execution of unit tests.
Requirements:	TEK_R_202, TEK_Data_02.
Tools:	devmate (AST).
KPI:	TEK_UCS_02_KPI_3.1_1
Use case scenario TEK_UCS_03 — Operating life monitoring	
Description:	Evaluation, in a semi-automatic manner, of the state of health of the system, interpreted on the basis of the data produced by the system.
Requirements:	TEK_R_104, TEK_R_201, TEK_R_203
Tools:	ConvHandler, Bridger, and DataAggregator (ROTECH)
KPI:	TEK_UCS_03_KPI_3.1_1

Globally, the case study develops a demonstrator whose development uses AIDOaRt solutions, tools and components, provided by UNICAN, UNIVAQ, AST and ROTECH. Figure 9.1 depicts the block diagram of the demonstrator and where the solutions are utilised.

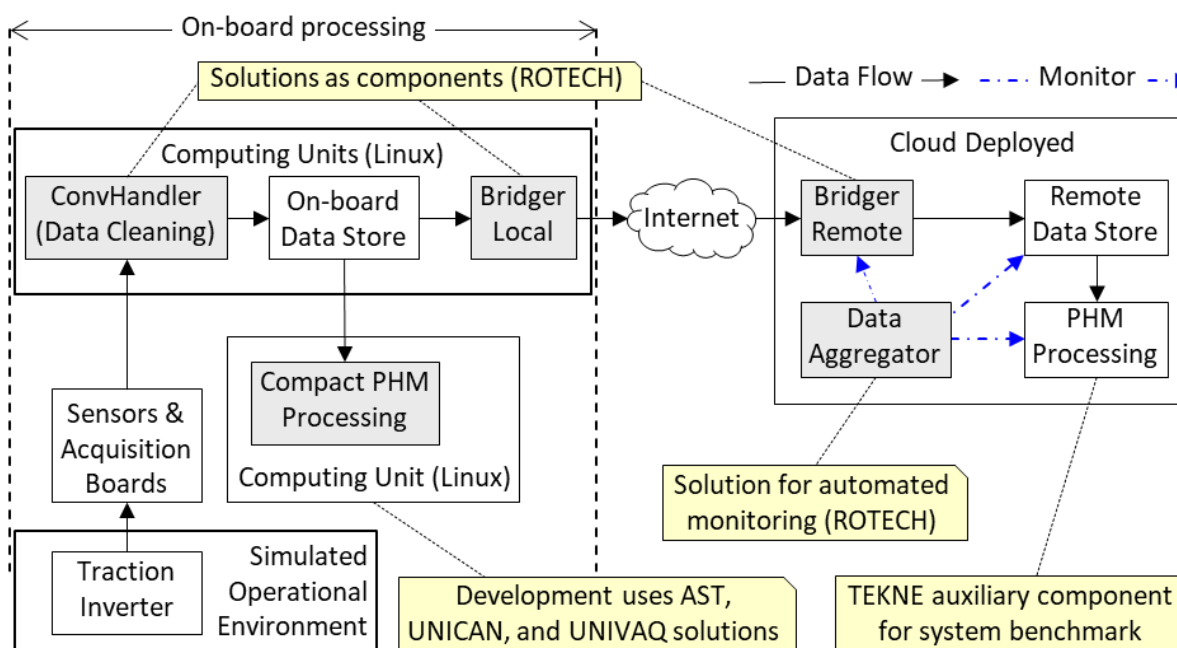


Figure 9.1 TEK demonstrator

The system, on one side, has near real-time diagnostics/prognostics capabilities, provided by a vehicular computing platform. On the other side, there is a cloud deployed computing platform. This stores data collected from a fleet of vehicles and runs complex AI algorithms, whose results can also be used for AI models validation and updating.

The inverter works in a simulated operation environment (battery, motor, mechanical load). Data (voltages, currents, and temperatures) are collected by sensors and acquisition boards. The ConvHandler component is in charge of data cleaning before the processing that is carried out by the Compact PHM Processing component. This is said to be “compact” because its AI methods run on a computing unit with limited capability. The local and remote Bridger components transfer data to the cloud deployed platform. Here, there are enough resources available to the PHM Processing component in order to run a PHM system with full capabilities. The DataAggregator component is in charge of monitoring the system for detecting and classifying defects during the operating life and for better performance.

9.1. Use case scenario TEK_UCS_01 — Design choices verification

The scenario TEK_UCS_01 “Design choices verification” deals with the verification of the models and with the design space exploration. The goal is to verify the adequacy (the functional aspects, as well as the response versus the resources) of the target components that the system architect has in mind to map/realise the design of the system. The development tools used in the use case are SD3 of UNICAN and HEPHYCODE of UNIVAQ.

HEPSYCODE (<https://www.hepsycode.com/>) is a hardware/software co-design methodology and framework designed for the development and co-simulation of heterogeneous multi-many-core embedded systems. The executable rendezvous system specification model is created in SystemC,

starting from a Graphical User Interface (GUI) where model-driven engineering aspects, trace, and model analysis capabilities are realised using the Eclipse project. The model-to-text transformation, followed by automatic SystemC translation, incorporates co-simulation features and functions that verify input constraints and requirements coverage.

Intermediate design space exploration enables the evaluation of multiple alternative solutions rapidly, employing Artificial Intelligence/Machine Learning algorithms such as evolutionary algorithms, Deep Neural Networks (DNN), regression trees, and random forests. Pareto analysis is applied to reduce the solution space and identify suboptimal implementations that meet input requirements (including timing performance, energy consumption, manufacturing, and implementation costs). Co-simulation activities are utilised to perform a final check for requirement satisfaction and recommend solution(s) suitable for the selected embedded application.

S3D (<https://s3d.unican.es/>) is a model-driven system design framework around a single UML/MARTE model. The model is used as a centralised repository of all the information about the system which is relevant for the design. From the model, the specific information required to perform a certain simulation and performance analysis task (e.g. performance simulation of a solution in a design-space exploration) is extracted and the corresponding simulation model, synthesised. This Model-to-Model generation is performed by the mSSYN tool. The M2M technology used to generate the simulation models automatically from the system model can be extended to synthesise the SW stack to be executed in each computational node. The corresponding synthesis tool is eSSYN (<https://essyn.unican.es/>).

The fundamental S3D modelling methodology is based on required-provided services and is able to support a publish-subscribe communication middleware. Model-Driven Design (MDD) is proposed for modelling, simulation and performance analysis of SW intensive robot-based services. Robots, as well as embedded systems in general can be integrated into a system engineering framework without affecting their modelling, design and development features. In order to seamlessly integrate additional kinds of electro-mechanical components, a platform-based approach is proposed.

SoSIM is a System-of-Systems Simulation tool. From the Single-Source System Design (S3D) model, the specific information required to perform simulation and performance analysis tasks is extracted and the corresponding simulation model is synthesised.

There are two AI technologies being implemented as SoSIM modules. The first, will infer performance and energy predictions on a target processor taking as inputs concrete performance indicators obtained from the execution of the code in the host. Training a neural network with data from both, the host and the target platforms, and using the host HW performance registers to take the run time figures of merit in the host, the tool will be able to exploit the capacity of a modern GPU (Graphics Processing Unit) to bring into the simulation environment on-line data of the actual performance of the code in the target. This technique will enable the simulation of code subject to much more complex structures than the basic flat uninterrupted block of code. The second, will use AI to statically predict from the target binary (RISC-V) its performance figures (time and energy) when executed.

The key idea of this approach is to execute a program on an intel x86 based machine while logging its intel's hardware performance counters with the exa-PAPI library. Then, feed that counted events into a Machine Learning model and predict the time and energy consumption that that same code would have when executed on another different architecture platform.

9.1.1. Summary of preliminary results

S3D — During the period covered by this deliverable, we have worked trying to replicate as close as possible the accuracy shown for this technique in Gerstlauer et al. paper [12]. We have worked in contact with Gerstlauer's team to verify our technique and analyse our findings. So far we have achieved a precision lower than the one achieved by them but still it is quite competitive when applied in the simulation context we are exploring.

We have characterised the nature of our benchmarks to compare ours against [13]. In addition, we have defined and calculated some metrics for the theoretical predictability of our generated dataset. At this point we are defining the theoretical limits for the actual accuracy of the technique in the presence of actual data (subject to various noise sources).

We are now using a more robust evaluation dataset and methodology and have solved some problems with our k-fold evaluation phase, which was giving us misleading readings. All in all our current technique yields good improvements over our previous results.

HEPSYCODE — Over the past months, UNIVAQ focus has been on refining the UC models by incorporating behavioural computation aligned with the TEKNE code and data model into the process network, as illustrated in Figure 9.2. Subsequently, UNIVAQ evaluated metrics for Design Space Exploration (DSE), such as workload, energy consumption, and manufacturing cost. These metrics were utilised in the AI search algorithm, specifically evolutionary algorithms, to identify sub-optimal architectural partitioning and mapping solutions.

Co-simulations were utilised to validate requirements fulfilment based on the specified metrics. Furthermore, we will conduct thorough analysis and evaluation to assess the reduction in design time. Machine Learning algorithms will also be applied to optimise DSE time, thereby increasing the number of feasible solutions found and identifying implementation alternatives not previously identified by the designers.

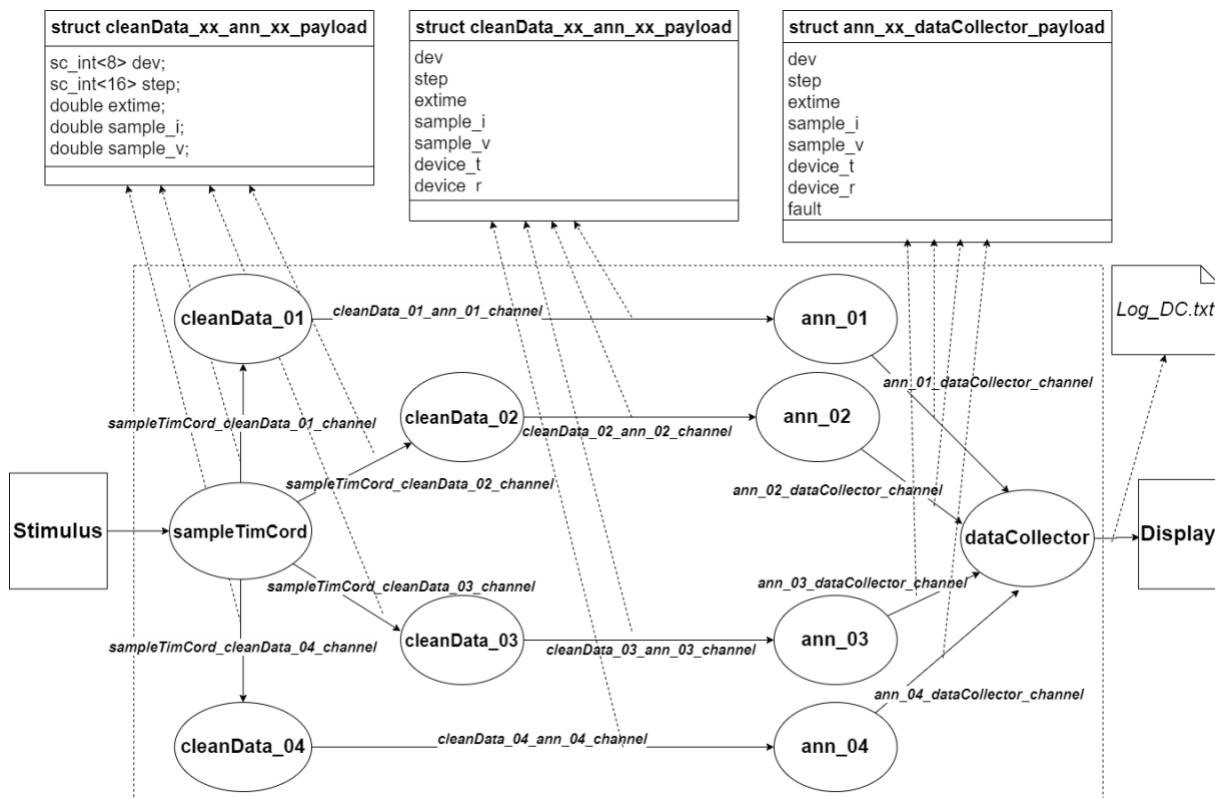


Figure 9.2 TEK System Specification Model using HEPHYCODE Framework

9.1.2. Evaluation of results considering requirements coverage

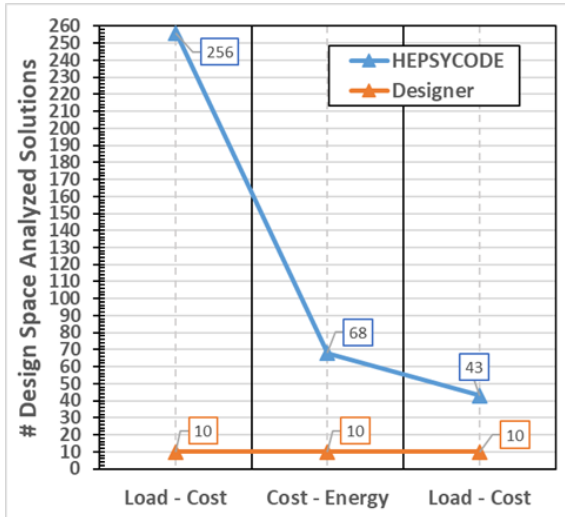
The following requirements will be evaluated in the use case scenario TEK_UCS_01:

- TEK_R_102: The AIDOaRt Framework verifies in a semi-automatic manner, at design time, with respect to the requirements, the adequacy (the response versus the resources) of the real components on-which/with-which the system architect has in mind to map/realise the architecture.
- TEK_R_103: The AIDOaRt Framework synthesises in a semi-automatic manner the models needed for the verification at design time (the models that define both the tests and the results, i.e. the pass/fail criteria).
- TEK_R_104: The AIDOaRt Framework interprets in a semi-automatic manner the results of the design time verification.

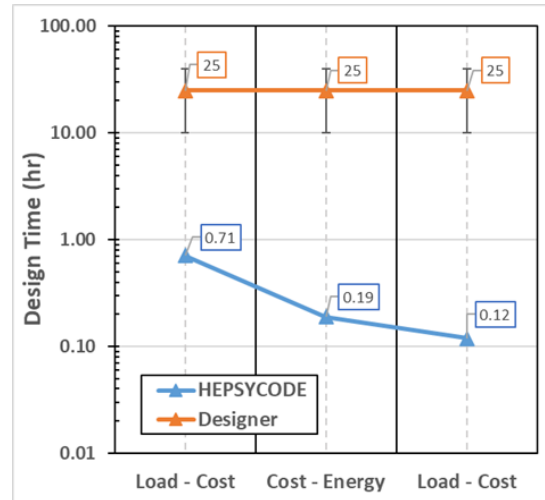
HEPSYCODE

Initial solutions were identified concerning workload, energy, and cost. The results demonstrate a simultaneous improvement in solution response time analysis and a greater reduction in design time, as depicted in Figure 9.3 (a), (b) and (c).

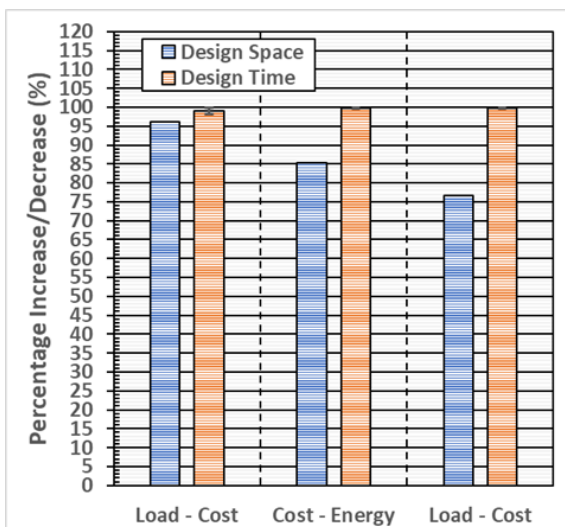
(a) HEPHYCODE Pareto Front size compared to designer's evaluated solutions



(b) HEPHYCODE Design Time reduction w.r.t classical approach



(c) HEPHYCODE design space size increase and design time decrease (in %)



(d) HEPHYCODE Pareto Frontier alternative solutions for TEKNE use case

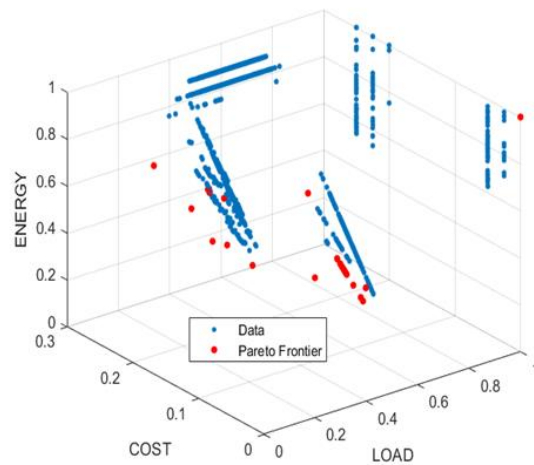


Figure 9.3 TEK System Specification Model using HEPHYCODE Framework

The analysis was conducted based on pairs of metrics, with the complete Pareto front results presented in Figure 9.3 (d). UNIVAQ was able to analyse a larger number of solutions than a traditional approach, while discovering a higher number of feasible solutions through rapid co-simulation methods.

UNIVAQ was able to analyse a larger number of solutions than a traditional approach, while discovering a higher number of feasible solutions through rapid co-simulation methods.

HEPSYCODE evaluated the following requirements:

- TEK_R_102: This requirement was met by employing the Design Space Exploration (DSE) approach using AI/ML algorithms. Further results and scenarios will be analysed and implemented to refine the final selected architecture.
- TEK_R_103: This requirement was satisfied using the model-driven Eclipse-based tool and framework. Further improvements will be introduced through collaboration with other solution providers (e.g., S3D and SoSIM, or process mining from JKU).
- TEK_R_104: This requirement is partially addressed as UNIVAQ aims to enhance simulation and solution search algorithms to enhance the accuracy of the proposed results while reducing design time.

S3D

In terms of simulation performance and latency, our proposed technology is in line with specifications and our results meet our initial requirements.

As an initial evaluation experiment we have used an 80% portion of the MiBench dataset to train our model architecture and the other 20% for doing the evaluation of the prediction accuracy. So far these very preliminary results have been promising (around 4% MAPE). This prediction accuracy is good enough, in the context of the simulation technologies in which it might be exploited [13].

At this point, all software integration details are solved. We have already tested our code on a simple flight management system as well as in TEKNE's use case scenario. It has been tested with previous models that led to not quite good precision.

9.1.3. Evaluation of results considering KPIs

TEK_UCS_01_KPI_1.2_1

Table 9.2 Case study KPI “TEK_UCS_01_KPI_1.2_1”

KPI Identifier: TEK_UCS_01_KPI_1.2_1		Scenario Identifier: TEK_UCS_01		
KPI Description:	Optimise verification time in the design phase.			
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the time required for identification of design problems thanks to the analysis of the collected data. (‡)		
	<i>Identifier:</i>	KPI_1.1 (‡)	<i>Target</i>	≥ 25 %
KPI Measure:	$k = t_r/t_n$, the ratio between the time t_r to verify, using the tool, r selected models, and the time t_n to verify the total number n of models within the design space. Designers select these r models based on their experience and the indication of the tool.			
KPI Baseline:	<i>Source:</i>	Project management dashboards, interviews with technicians.		

	<i>Value:</i> $k_0 =$	0 % (no automation)		
Target:	<i>Increase:</i>	k	\geq	25 %

(‡) In the previous deliverable D5.6 “Use Case Development Report – 1” [9], the scenario TEK_UCS_01 was declared to address the project KPI_1.2 “Improvement of the early detection of system deviations”. Now, the project KPI is KPI_1.1 “Improvement of the time required for identification of design problems thanks to the analysis of the collected data”. The reason for this change is to better adhere to the activities carried out and the tools used in the scenario that considers performance analysis and design space exploration more than model verification.

Tools are still in development, so the KPI will be measured afterwards.

9.2. Use case scenario TEK_UCS_02 — Run-time verification

The scenario TEK_UCS_02 “Run-time verification” is related to software testing. TEK searches for a solution that, in a semi-automatic way, is capable of defining and executing the tests, as well as interpreting the tests results.

TEK use case scenario requires automatic generation of test code. The AST tool, *devmate*, uses technologies for equivalence-based testing, to produce a test model from which we can generate test code. Other features of *devmate* include the user interface to handle the input and output definition of the functions under test, a compatible parser, a plugin architecture for a suitable IDE and data handling of equivalence classes and test data, necessary for the proper fulfilment of the requirements.

To satisfy TEK use case scenario, the tool *devmate* of AST needed support for the C language. Hence, AST’s first tasks were to develop a parser for the C language and to adapt the user interface for C specific features. A code generator was developed that produces test code in C++, executable with GoogleTest.

The different modules are part of *devmate*, currently delivered in the form of an IDE plugin. As several difficulties arose from Eclipse IDE, a standalone version was developed to avoid these problems.

The tool *devmate* can be used in two ways:

- It can be used to define test data and expected output based on the requirements of the user and therefore the generated code is an automated code for the validation of the implemented function under test (check if the function works as intended).
- The generated test code can further be used in case of refactoring for the verification of the function under test (after refactoring the function works in the same way furthermore).

9.2.1. Summary of preliminary results

TEKNE reviewed the prototype for automatic unit test case and for test code generation for C. Also on the basis of the feedback (e.g. about test management features, complex test data, editor user

interface, some problems in C-parser), AST decided to make a redesign of the architecture and user interface to improve usability and create an enhanced basis for planned features.

Advantages of the new features:

- complete new features to manage tests and complex structures of tests in a visual canvas-based user interface;
- fuzzy search features for easier finding tests and data;
- possibility to define test cases in a more intuitive and easier way in the test editor also based on visual canvas-oriented user interface;
- enhancement of method and function test definition by also integration test definition features;
- optimisation of parser for C code;
- enhancement of parser for C++ code;
- easier handling of test cases and test data for the user.

9.2.2. Evaluation of results considering requirements coverage

The following requirements will be evaluated in the use case scenario TEK_UCS_02:

- TEK_R_202: The AIDOaRt Framework synthesises, in a semi-automatic manner, the models needed for the verification at run time (the models that define the tests and the tests results).
- TEK_Data_02: Monitoring data of test execution and processing of results.

More work and research is necessary to bring it to the expected level: devmate as a standalone tool that fulfils the requirements *TEK_R_202* and *TEK_Data_02* in the form of more automation.

9.2.3. Evaluation of results considering KPIs

TEK_UCS_02_KPI_3.1_1

Table 9.3 Case study KPI “TEK_UCS_02_KPI_3.1_1”

KPI Identifier: TEK_UCS_02_KPI_3.1_1		Scenario Identifier: TEK_UCS_02		
KPI Description:	Reduction of testing effort.			
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).		
	<i>Identifier:</i>	KPI_3.1	<i>Target</i>	≥ 30%
KPI Measure:	k = number of module tests that are automated, as percentage of the total.			
KPI Baseline:	<i>Source:</i>	Project management dashboards and interviews with expert technicians.		
	<i>Value: k₀ =</i>	0 % (no automation)		
Target:	<i>Increase:</i>		k ≥	30%

Currently the prototype of the new model and user interface is planned to be evaluated within the hackathon meeting in Dec. 2023 in Linz. At the moment no KPIs are systematically measured regarding the prototype state and the big changes carried out. So it is planned to gather some first usage metrics in this evaluation within the hackathon.

9.3. Use case scenario TEK_UCS3 — Operating life monitoring

In the scenario TEK_UCS_03 “Operating life monitoring”, the demonstrator operates and provides its services in an environment partly simulated. Measured data are collected and pre-processed (cleaning, filtering, features extraction) to obtain “monitoring data”. These are processed by the compact on-board PHM processing. Moreover, they are transferred to the remote computing and data storage whose resources are available to run a full capabilities PHM system.

9.3.1. Summary of preliminary results

In this scenario, ROTECH is developing 3 solutions that provide different capabilities, useful to achieve the requirements identified by TEKNE.

Bridger — The solution provides the communication between the On-board Platform and the Remote Platform. It introduces the features of secure communication using an encryption and decryption algorithm. There are two instances:

- The On-board Bridger, which is integrated in the On-Board Platform and receives the data to be encrypted and published via MQTT (Message Queuing Telemetry Transport)).
- The Remote Bridger, which is integrated in the Cloud Platform and subscribes to the topic where the data is published by the On-Board Bridger, performs decryption and stores the data in a database.

ConvHandler — The solution performs filtering and cleaning of the data to be fed to the Bridger in the On-board Platform specific to the scenario.

Data Aggregator — DataAggregator monitors and analyses the performance of the system, in order to support decisions about maintenance (corrections and improvements). Consistent evolution and the usability of the data play a key role too, by increasing the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).

DataAggregator manages four configurations:

- In-comm aggregation: use of a multi-hop system for the process of gathering and routing tests information inside the Data Aggregator.
- Tree-based approach: use of an aggregation tree, mapping and putting the data from leaves to root (source and sink nodes respectively).
- Cluster-based approach: used to collect large amounts of data across the entire Platform.
- Multi-path approach: use of partially aggregated data sent to the root or parent internal table which then can send the data along various paths.
- Enhancement of method and function test definition by also integrating test definition features.

For the ConvHandler and the Bridger, tests have been conducted in a simulation environment to assess the correct operation of the tools and their deployment in an environment similar to the one used in the Scenario. Both solutions' functionalities are reported in the deliverable D2.3 [5].

The solutions are working as intended and in the last period we focused on improving the latency of the Bridger to reduce the time to exchange the information received from sensors. In particular we introduced a compression algorithm to reduce the size of the exchanged data, reducing the time spent during the transmission. So far we have performed tests using two CSV (Comma Separated Values) files of different sizes provided by TEKNE. The files contain data samples similar to the data produced by the real environment. The first file is named Sb.csv and it is around 5 MB in size and the second file is named FFT.csv with a size of around 89 MB.

The following tables represent the average of 10 test executions with the time in seconds for each phase of the Bridger.

Table 9.4 - Average time (seconds) of test execution on Sb.csv (4.62 MB)

Operation	Bridger On-Board	Bridger Remote
File read	0,1314	n/a
Compression	0,0953	n/a
Decompression	n/a	0,0256
Encryption	0,0068	n/a
Decryption	n/a	0,0052
Data transfer	1,7037	n/a

Table 9.5 - Average time (seconds) of test execution on FFT.csv (88.5 MB)

Operation	Bridger On-Board	Bridger Remote
File read	2,8528	n/a
Compression	2,8828	n/a
Decompression	n/a	0,9949
Encryption	0,1616	n/a
Decryption	n/a	0,1157
Data transfer	22,8978	n/a

9.3.2. Evaluation of results considering requirements coverage

The following requirements will be evaluated in the use case scenario TEK_UCS_03:

- TEK_R_104: The AIDOaRt Framework interprets in a semi-automatic manner the results of the design time verification.
- TEK_R_201: The AIDOaRt Framework verifies in a semi-automatic manner the implemented software artefact (system, sub-system, component) with respect to the requirements as well as with respects to the architectural and detailed models.

- TEK_R_203: The AIDOaRt Framework interprets, in a semi-automatic manner, the results of the runtime verification.

DataAggregator parses in a semi-automatic manner the results of the scenario tests and provides the results to the AIDOaRt Framework runtime verification.

9.3.3. Evaluation of results considering KPIs

TEK_UCS_03_KPI_3.1_1

Table 9.6 Case study KPI “TEK_UCS_03_KPI_3.1_1”

KPI Identifier: TEK_UCS_03_KPI_3.1_1		Scenario Identifier: TEK_UCS_03		
KPI Description:	Improvement of capability of detecting and classifying defects during the operating life.			
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).		
	<i>Identifier:</i>	KPI_3.1	<i>Target</i>	≥ 30%
KPI Measure:	k = number of automated tests, as percentage of the total.			
KPI Baseline:	<i>Source:</i>	Project management dashboards and interviews with expert technicians.		
	<i>Value:</i> $k_0 =$	0 % (no automation)		
Target:	Increase:		k	≥ 30%

The KPI can be measured on the final, running version of the demonstrator.

9.4. Planned improvements

TEK_UCS_01 (S3D)

We are still trying to solve the limitations of our technology to get our accuracy as close as possible to what [12] claims as possible: in our case, the way of doing the data gathering and the calculations in real time are different, so it is not correct to compare the results directly. So, even if prediction accuracy is good enough compared to the simulation technologies with which we compete [13], we still plan to work on enhancing our model to improve our accuracy, in particular we plan to use PAPI library multiplexing to refine the counters we use.

TEK_UCS_02 (devmate)

Support for the C language proved to be difficult, mostly because of pointer mechanics. As such development of the parser focuses on features required by the use case. Similarly test code generation in C++ provided challenges in memory allocation and type handling.

Adapting the user interface to and integrating GoogleTest in the Eclipse plugin architecture caused several problems so the decision was to refactor the whole devmate tool which requires a new internal model and a complete new user interface (which will be called Vizitest). This is currently under development.

More work and research is necessary to bring it to the expected level: devmate/Vizites as a standalone tool will fulfil the requirements TEK_R_202 and TEK_Data_02 in the form of more automation.

TEK_UCS_03 (ConvHandler, Bridger, DataAggregator)

The main improvements that we are planning to do are related to the data transfer time of the Bridger to further reduce its latency and increase its performance. In particular, we are considering the idea of changing the encryption algorithm and using the MQTTS (TLS v1.4) to encrypt the data sent and received by both the Bridger On-board and Remote.

For the Data Aggregator, we are working on the data collection in order to optimise the scenario test execution.

9.5. Planned demonstration

For the use cases TEK_UCS_01 and TEK_UCS_02, which are relevant to the design and test, we plan to show (live and video) some significant steps that illustrate the usage of UNICAN and UNIVAQ tools for design space exploration and performance evaluation, as well as of AST tool for test automation.

For the use case TEK_UCS_03 we plan a video with the final demonstrator working in a simulated operating environment.

The demonstrations will show a summary of KPIs measurement and requirements verification.

10. VCE_CS09 case study “Data modelling to support product development cost and efficiency”

10.1. Case Study description

The VCE case study revolves around the transformation of system design activities towards model-based approaches. In particular, VCE develops product lines of Construction machines for various purposes and a wide range of contexts. In the AIDOaRt project architecture descriptions of a Dumper machine (currently formulated in documents and office tools) are used as a motivating example of the case study. Overall the use case has the goal of improving efficiency and introducing automation in the current primarily manual workflow of the system architecture description. A more thorough description of the use case can be found in deliverable D5.6 [9].

During the active work in AIDOaRt, a set of challenges are observed to be the main inhibitors of the successful application of partners solutions:

CH1: Heavy reliance on legacy tools and artefacts. In order for a solution developed in AIDOaRt to be successfully implemented it needs to be integrated and supported by surrounding legacy (e.g. Excel and Visio)

CH2: Uncertainty in design. The considered stage considers the architecture phase of development, and as such information might be missing from the overall context.

CH3: Tooling interoperability. Interoperability is often considered a weakness of model-based practices, and in the VCE use case it is necessary with interoperability not only between various modelling tools but also non-modeling tools (for example for simulation or legacy tools).

CH4: Non-modelling target audience. Although the use case primarily focuses on the introduction of modelling into the current workflow, the target users are mostly non-modellers. As such, the solution needs to be user friendly and support user adoption.

Currently, the work in AIDOaRt considers a collaboration of VCE with the partners: UNIVAQ, JKU, IMTA, DT, MDU, and AVL. The collaboration is focused on the development of a unified solution framework to address the concerns raised in the case study, where the different solutions are integrated with one another. The collaboration on the case study and the different use case scenarios is summarised in Table 10.1.

Table 10.1 Synopsis of the case study VCE

Use case scenario VCE_UCS_01 — Modeling system, software, data architectures	
Description:	VCE together with the solution providers are developing a toolchain which aims to enable the modelling of VCE systems along with added capabilities of AI and DevOps.
Requirements:	<ul style="list-style-type: none"> ● VCE_R05 - Customise standards based modelling frameworks (e.g. UAF, SysML, UML) and metamodels to develop system, software, data architecture models ● VCE_R07 - Development of standard data classification, reusable definition, representation, usage
Tools:	<ul style="list-style-type: none"> ● MORGAN (UNIVAQ), ● ATL (IMTA), ● EMF Views (IMTA), ● Modelio (Soft), ● AutomationML Modeling (JKU), ● Modelling process mining tool (JKU)
KPI:	<ul style="list-style-type: none"> ● VCE_UCS_1_KPI_1.1_1, ● VCE_UCS_1_KPI_2.2_1, ● VCE_UCS_1_KPI_3.1_1
Use case scenario VCE_UCS_02 — Validation and verification of architecture models	
Description:	VCE with the solution providers aims to introduce capabilities of validation and verification of the models developed through the overarching collaboration. In particular the use of co-simulation is envisioned to be a suitable solution with DevOps support.
Requirements:	<ul style="list-style-type: none"> ● VCE_R01 – Use automated reasoning and ML techniques for verification of specifications and high-level models ● VCE_R04 – Use of automated tools for compliance verification ● VCE_DR_01 – The models should include domain specific information in a standardised format ● VCE_DR_02 – Tool should be able to extract the essential information and create documentation
Tools:	<ul style="list-style-type: none"> ● Keptn (DT) ● MOMoT (JKU)
KPI:	<ul style="list-style-type: none"> ● VCE_UCS_2_KPI_1.2_1 ● VCE_UCS_2_KPI_3.1_1

Use case scenario VCE_UCS_01 — Modeling system, software, data architectures	
Use case scenario VCE_UCS_03 — AI augmented DevOps workflow and data analytics	
Description:	VCE with solution providers aim to further support the overall solution with analytical capabilities through AI integrated via DevOps flows.
Requirements:	<ul style="list-style-type: none"> ● VCE_R02 – AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS ● VCE_R03 – Use ML for predicting values which are actually not measurable ● VCE_R06 – Integration of DevOps workflows and continuous integration/configuration of models and corresponding technical solutions ● VCE_R07 – Development of standard data classification, reusable definition, representation, usage ● VCE_DR_02 – Tool should be able to extract the essential information and create documentation
Tools:	<ul style="list-style-type: none"> ● Keptn (DT)
KPI:	<ul style="list-style-type: none"> ● VCE_UCS_3_KPI_3.1_1 ● VCE_UCS_3_KPI_3.2_1

Although the collaborations with partners are performed towards different use case scenarios the overall collaboration can be considered to be unified. Joint weekly meetings in addition to other more sporadic activities are performed with all of the aforementioned partners, and the use case scenarios are more or less worked towards in the same solution framework, highlighted below in figures 10.1 and 10.2.

Figure 10.1 depicts a solution architecture that integrates existing partners' solutions, together referred as AIDOaRt toolset, and external components as open-source tools, and newly developed/under development components, like model transformations, to cope with specific automation and integration needs of the VCE case study, and possibly, part of the AVL case study (Simulink's based simulation).

Figure 10.2 depicts the engineering workflow supported by the proposed solution architecture in an activity-like diagram. The components' ids in Figure 10.1 are used to identify the corresponding supported engineering action in Figure 10.2, and macro-activities depicted as swimlanes i.e., SysML modeling, AutomationML modeling, and modeling recommendations.

Currently, the proposed architecture and workflow is semi-automated. Domain experts are expected to perform modeling and continuous delivery activities through different tools (Papyrus UML, CAEX Modeling Workbench, Keptn).

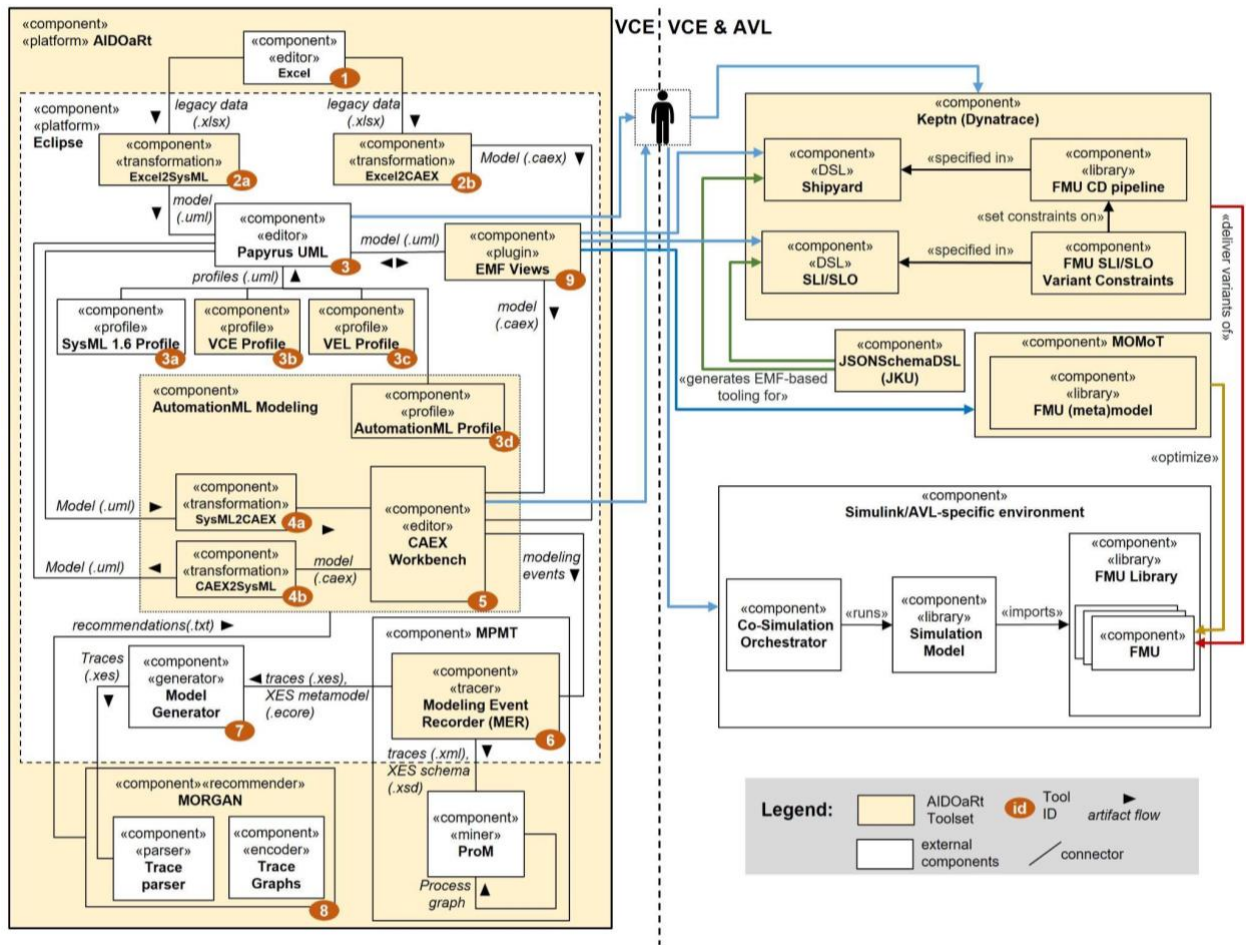


Figure 10.1 Overall view of the solution architecture in the VCE use case.

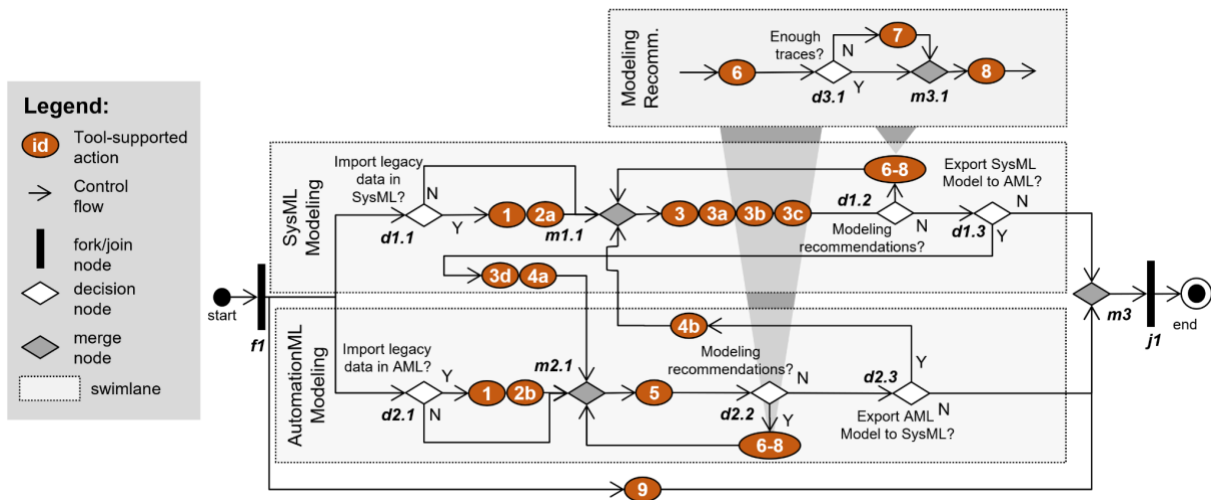


Figure 10.2 The usage of the proposed architecture solution depicted in Figure 10.1

10.2. Use case scenario VCE_UCS_01 — Modelling system, software, data architectures

This use case scenario regards the creation of modelling guidelines and patterns for architectures of VCE systems. Overall the use case scenario is the main one for the VCE case study, and most of the work in the collaborative efforts are related in some way to the scenario, or uses the implementation results as an input. The scope of the use case scenario is the transformation of non-formal artefacts (such as Excel and Visio) to model-based representations in standard languages (like SysML and AutomationML). Legacy descriptions are used as input for the work on the scenario, where the aim is to create corresponding representations in model-based approaches, while supporting legacy artefacts. Furthermore, the solution should enable the integration of AI and DevOps in a seamless fashion with added capabilities to demonstrate the value of model-based in comparison to legacy way of working.

10.2.1. Summary of preliminary results

To support modelling recommendations, UNIVAQ provides MORGAN, a modelling recommender system based on graph kernels. Using a predefined knowledge base, the tool can suggest relevant artefacts similar to the ones already in place, thus reducing the possible choices and the whole design time. To this end, we combined MORGAN with the MER component provided by JKU to collect the modeller's behaviour in terms of actions, i.e., creating, renaming, and deleting model elements.

The preliminary results of the hackathon show that MORGAN is capable of providing model operations even though the tool has been fed with random models. In future iterations, we plan to validate MORGAN using real data provided by VCE engineers. Furthermore, we will integrate all the capabilities offered by MORGAN and MER in an Eclipse plugin, thus supporting Sirius-based editors with modelling recommendations.

To support multiview modelling, IMTA provides EMF Views, an approach and corresponding Eclipse/EMF-based tool for specifying and building views over one or several models that potentially conform to different metamodels. In the present VCE context, multiview modelling is considered as an engineering activity independent from SysML and AutomationML modelling, so it can be also applied to federate any EMF-based models involved in the CPS engineering process.

The preliminary results of the last hackathons show that EMF Views can be used in practice to federate/interrelate SysML, AutomationML and FMU models into integrated views that VCE engineers can then navigate and query depending on their needs. This way, they can more easily get an overall vision of the system under study and take design decisions accordingly without referring anymore to legacy data from Excel sheets.

AutomationML modelling is enabled by the JKU CaeX workbench which is integrated in Eclipse. In the defined framework AutomationML modelling is added as a potential modelling language in

conjunction with SysML for the engineers. Apart from AutomationML, JKU also provides a model process mining tool which records the events of a modeller when using the Eclipse platform. In this way, data can be gathered on the traces of modellers when using the solution framework presented earlier.

The preliminary results show that both AutomationML and Modelling process mining can be integrated into the overall framework. Particularly, the process mining is able to generate data traces which can be used for the modelling recommendations, while the AutomationML models are possible to integrate with SysML models using various model transformations.

10.2.2. Evaluation of results considering requirements coverage

The requirements that are applicable for this Use Case are VCE_R05 and VCE_R07.

- **VCE_R05 - Customise standards based modelling frameworks (e.g. UAF, SysML, and UML) and metamodels to develop system, software, data architecture models**
- **VCE_R07 - Development of standard data classification, reusable definition, representation, usage**

So far, both of these requirements are fulfilled as the Use Case relies on standard modelling languages (UML, SysML, AutomationML) and customise the notations to enable VCE concepts. Furthermore, in order to facilitate re-use of legacy artefacts and data patterns templates are created to facilitate automatic transformations across languages and artefacts.

The modelling recommendations provided by MER+MORGAN have been evaluated by conducting an initial user study. In particular, we involved four different types of engineers from VCE, i.e. Verification Engineer, Software Engineer, System Architect, and System Engineer. Each session took roughly 1.5 hours and we collected their feedback in a structured questionnaire. Overall, the participants are satisfied with the examined aspects even though a deeper evaluation is needed. We plan to extend the evaluation by (1) collecting additional modelling traces and (2) involving more participants from VCE.

The multiview modelling support provided by EMF Views has been evaluated by considering only a limited set of models at this stage. In particular, we have been able to experiment on the specification and building of an initial “complete” view interconnecting SysML, AutomationML models (as design models) and more recently a FMU model (as a runtime model). This already showed promising results in terms of model federation capabilities within the VCE context.

In the coming year, we plan to extend the evaluation by(1) considering various sets of (possibly large) models of the same kind as inputs to the view, (2) integrating other languages/metamodels to be also federated within the same view or via complementary views, and (3) combining EMF Views with ML techniques in order to (semi-)automate the generation of the views.

For the end of the project, we envision to provide a demonstration, based on VCE models, to display the final results of our work on the previously discussed extensions (possibly reaching different levels of achievements depending on the needs of VCE and the underlying complexity).

The use of AutomationML has been integrated with the legacy concepts of VCE workflows, and so far captures the necessary concepts in a clear and understandable modelling pattern. In this regard it can in the future be extended to capture information needed for more advanced analysis capabilities or variability management.

The modelling process mining tool has been able to capture the basic elements of user operations and been able to accurately store the data for AI algorithm training. In this regard the tool acts as a baseline technology to record the events of modellers (expert or beginners) so that it can be analysed for further use.

10.2.3. Evaluation of results considering KPIs

VCE_UCS_1_KPI_1.1_1

Table 10.2 Case study KPI “VCE_UCS_1_KPI_1.1_1”

KPI Identifier: VCE_UCS_1_KPI_1.1_1		Scenario Identifier: VCE_UCS_01	
KPI Description:	Increase in development velocity, utilising MDE.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	
	<i>Identifier:</i>	KPI_1.1	<i>Target</i> ≥ 25%
KPI Measure:	k = Hours spent on tasks.		
KPI Baseline:	<i>Source:</i>	Measurement from current tasks from project management tools.	
	<i>Value: k₀ =</i>	Varies between tasks but measured in hours. Exact value is not decided as it depends on the scope of the final evaluation.	
Target:	Decrease:	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 10%

We will measure this KPI by comparing the expected time for certain architecture definitions by the methods introduced by the solution providers. It is expected that there will be improvements in the time spent on tasks, in particular we will focus on the development of a physical system architecture.

VCE_UCS_1_KPI_2.2_1

Table 10.3 Case study KPI “VCE_UCS_1_KPI_2.2_1”

KPI Identifier: VCE_UCS_1_KPI_2.2_1		Scenario Identifier: VCE_UCS_01	
KPI Description:	Re-use of architectural models in future projects.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the number of available data sources to be actually managed and handled in existing engineering practices.	
	<i>Identifier:</i>	KPI_2.2	<i>Target</i> ≥ 25%

KPI Identifier: VCE_UCS_1_KPI_2.2_1		Scenario Identifier: VCE_UCS_01	
KPI Measure:	k = Number of re-used artefacts		
KPI Baseline:	<i>Source:</i>	Current artefacts at stage of development	
	<i>Value: k₀ =</i>	0	
Target:	<i>Increase:</i>	k ≥	25%

Currently there is no re-use of artefacts created for architecture definitions. We expect that by the use of the technologies developed inside AIDOaRt we should be able to re-use a significant amount of artefacts previously defined. In this way, we are interested in evaluating how much of the information that is represented in non-formal descriptions can be re-used when creating model-based approaches, mostly in the form of excel tables.

VCE_UCS_1_KPI_3.1_1

Table 10.4 Case study KPI “VCE_UCS_1_KPI_3.1_1”

KPI Identifier: VCE_UCS_1_KPI_3.1_1		Scenario Identifier: VCE_UCS_01	
KPI Description:	Automation of modelling activities.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = Reduction of time spent on activities by replacing activities with automated means		
KPI Baseline:	<i>Source:</i>	Measurement from current tasks from project management tools.	
	<i>Value: k₀ =</i>	Varies between tasks but measured in hours. Exact value is not decided as it depends on the scope of the final evaluation.	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

We assume that most tasks performed in the workflow are currently manual, therefore the aim is to replace a significant amount of the currently performed tasks with automated means. In particular, the created solution architecture should replace a percentage of the estimated time spent on creating physical system architectures with automated processes. This is expected to be performed via the use of model transformations and MDE technologies.

10.3. Use case scenario VCE_UCS_02 — Validation and verification of architecture models

The second use case scenario in the VCE case study regards the introduction of improved V&V in the architecture definition of systems. In particular, previously used methods and tools do not contain any semantic meaning nor formal use of any language or notation. In this regard, a goal of the project is the added capabilities in this regard from the use of a primarily model-based approach to system

description. Expected outcomes of this use case scenario is the introduction of simulation, consistency checks, and automation in the workflows of architecture creations.

10.3.1. Summary of preliminary results

The MOMoT tool has been investigated for the possibility of automatically generating system architecture layouts in SysML based system requirements and lists of available components. The MOMoT tool leverages a metamodel and graphs transformation approach to generate an output graph that can consist of different elements and connections.

In the VCE case, the generated graphs correspond to Internal Block Diagrams in SysML, and can be translated from graph representation to a model. Based on the use case, a metamodel has been initially generated which can be used in conjunction with user constraints to generate viable architectures, seen below are some abstract examples in figures 10.3 and 10.4:

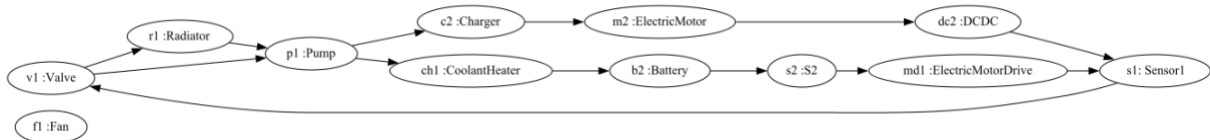


Figure 10.3 Example 1 of a generated architecture graph

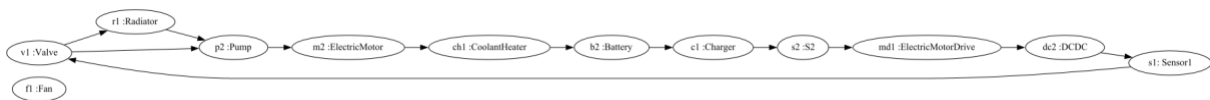


Figure 10.4 Example 2 of a generated architecture graph

The next steps would be to better reason about which generated architectures are more optimal, and improve the algorithms involved for finding a suitable architecture.

10.3.2. Evaluation of results considering requirements coverage

The requirements related to this use case scenario are:

- **VCE_R01 – Use automated reasoning and ML techniques for verification of specifications and high-level models**
- **VCE_R04 – Use of automated tools for compliance verification**
- **VCE_DR_01 – The models should include domain specific information in a standardised format**
- **VCE_DR_02 – Tool should be able to extract the essential information and create documentation**

VCE_R01 is partially fulfilled and is being worked towards in continued effort. In particular, we are interested in employing automated reasoning to automatically generate system architectures as

SysML Internal Block Diagrams, and AI can be used to reason about which solutions are more or less optimal. VCE_R04 is currently not fulfilled, and needs to be addressed in further development. Both VCE_DR_01 and VCE_DR_02 are currently fulfilled by the solution architecture defined in Figure 10.1.

10.3.3. Evaluation of results considering KPIs

VCE_UCS_2_KPI_1.2_1

Table 10.5 Case study KPI “VCE_UCS_2_KPI_1.2_1”

KPI Identifier: VCE_UCS_2_KPI_1.2_1		Scenario Identifier: VCE_UCS_02	
KPI Description:	Inconsistency detection in models.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the early detection of system deviations.	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> ≥ 30%
KPI Measure:	k = Increasing coverage of inconsistency automatically		
KPI Baseline:	<i>Source:</i>	Current tools employed internally.	
	<i>Value: k₀ =</i>	Number of inconsistency rules automatically identified.	
Target:	<i>Increase:</i>	k ≥	10%

There are many different types of artefacts containing information in the VCE development chain. Inside AIDOaRt we expect to improve the automatic inconsistency detection across various artefacts. The integrated solution framework should be able to detect inconsistencies across the various developed artefacts.

VCE_UCS_2_KPI_3.1_1

Table 10.6 Case study KPI “VCE_UCS_2_KPI_3.1_1”

KPI Identifier: VCE_UCS_2_KPI_3.1_1		Scenario Identifier: VCE_UCS_02	
KPI Description:	Automate V&V on architecture models.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = Time spent on V&V activities on architecture models		
KPI Baseline:	<i>Source:</i>	Measurement from current tasks from project management tools.	
	<i>Value: k₀ =</i>	Varies between tasks but measured in hours.	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

We are expecting that part of the currently performed activities should be automated, in particular the simulations that can be performed at this stage. We expect that the solution framework should be able to utilise DevOps concepts to offload the engineers.

10.4. Use case scenario VCE_UCS_03 — AI augmented DevOps workflow and data analytics

While the other use case scenarios primarily regard the use of modelling to improve the processes of development, the third use case scenario regards AI and DevOps. Indeed, by transforming the architecture descriptions and related information to a primarily model-based representation it is expected that AI and DevOps can be more readily applied. Therefore, the third use case scenario aims to augment the developed solutions in the first two use case scenarios to include the use of AI and DevOps to further improve the overall goal of reducing manual activities and leveraging the power of model-based approaches.

10.4.1. Summary of preliminary results

This use case scenario primarily uses the Keptn tool from DT. By linking SysML architecture models (e.g. from VCE_UCS_01) with simulation models following the FMI standard (FMUs), the Keptn tool is able to generate an automated pipeline for model execution and evaluation, independent of any particular tool. The overall approach is visualised in Figure 10.5:

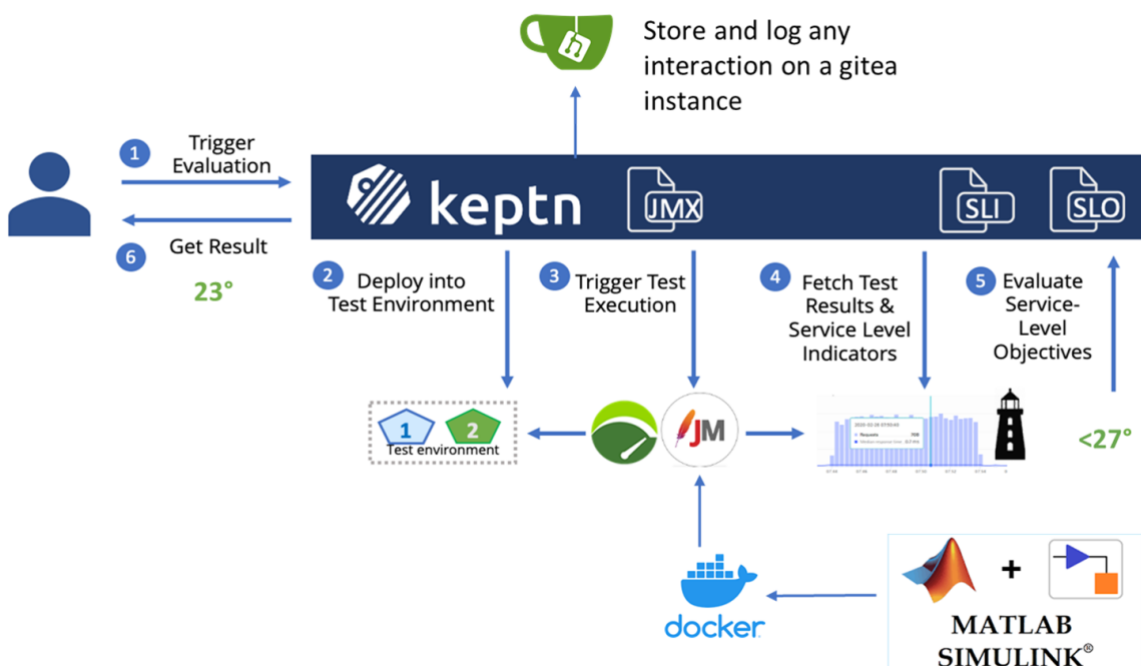


Figure 10.5 Overall architecture of the integrated Keptn pipeline in the Use Case Scenario

A user inputs a model, some initial parameters, and evaluation target of one or more outputs. This information is fed to Keptn which initialises simulations by deploying a simulation environment in a docker container and performing the simulations until a valid result is obtained. In our case, we have utilised a thermal management system as a simulation model and observed the battery temperature during simulation, in order to find a suitable parameterization of the model. Figure 10.6 shows how the case is presented in AIDOaRt, where different variants are highlighted through simulation traces.

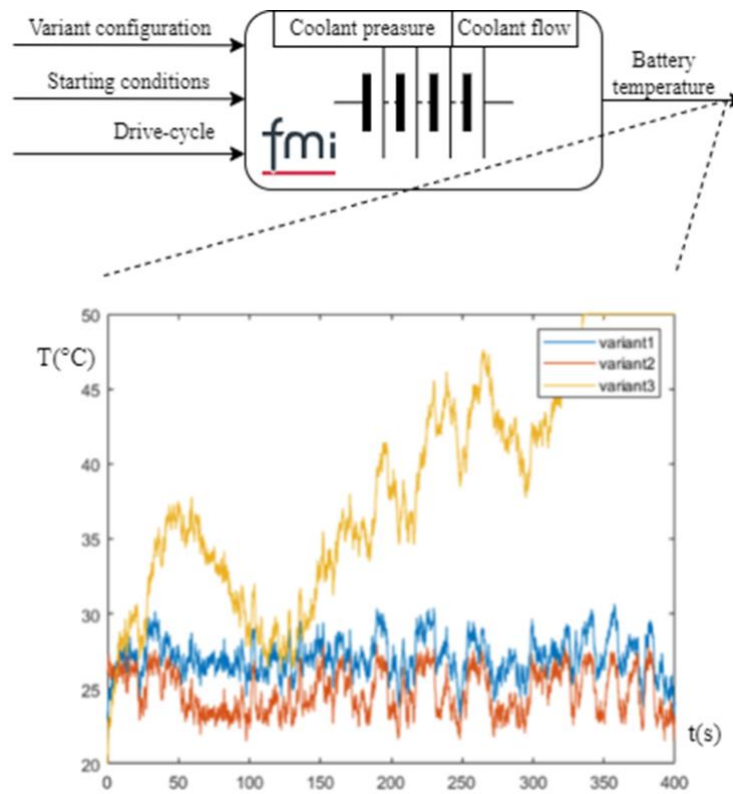


Figure 10.6 Thermal management system run as an FMU to generate simulation traces of variants

10.4.2. Evaluation of results considering requirements coverage

The related requirements are:

- **VCE_R02 – AI/ML method for auto-adjusting model parameters w.r.t. similarity of execution traces of a Digital Twin with a CPS**
- **VCE_R03 – Use ML for predicting values which are actually not measurable**
- **VCE_R06 – Integration of DevOps workflows and continuous integration/configuration of models and corresponding technical solutions**
- **VCE_R07 – Development of standard data classification, reusable definition, representation, usage**
- **VCE_DR_02 – Tool should be able to extract the essential information and create documentation**

VCE_R02 is expected to be met further on with the added work towards the integrated DevOps pipeline. VCE_R03 is similarly expected to be integrated with the co-simulation solution. VCE_R06 is currently being fulfilled via the use of the continuous integration and delivery of simulation models. VCE_R07 is met as it is a precondition for many of the other aspects of the tool-chain. VCE_DR_02 is currently met as the simulation traces can be stored and published for the user along with other information regarding the model(s).

10.4.3. Evaluation of results considering KPIs

VCE_UCS_3_KPI_3.1_1

Table 10.7 Case study KPI “VCE_UCS_3_KPI_3.1_1”

KPI Identifier: VCE_UCS_3_KPI_3.1_1		Scenario Identifier: VCE_UCS_03	
KPI Description:	Automate manual processes.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = The amount of time spent on manual processes for tasks.		
KPI Baseline:	<i>Source:</i>	Measurement from current tasks from project management tools.	
	<i>Value: k₀ =</i>	Varies between tasks but measured in hours. Exact value is not decided as it depends on the scope of the final evaluation.	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$	≥ 30%

We would like to automate processes within AIDOaRt, and expect that expected time will be reduced for architecture modelling and definition via the use of integrated automated methods. This will be measured by extracting the time spent on commonly performed manual tasks in the engineering workflows and compare the same task effort when considering the AIDOaRt framework. While concrete tasks might differ depending on the actual case and considered artefacts, examples might include transferring data from one source to another, creation of certain architectural descriptions, documentation, etc.

VCE_UCS_3_KPI_3.2_1

Table 10.8 Case study KPI “VCE_UCS_3_KPI_3.2_1”

KPI Identifier: VCE_UCS_3_KPI_3.2_1		Scenario Identifier: VCE_UCS_03	
KPI Description:	Improve coverage of data gathering.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase the coverage and quality of actionable feedback for the next DevOps iteration.	
	<i>Identifier:</i>	KPI_3.2	<i>Target</i> ≥ 25%
KPI Measure:	k = Increase in the types of gathered data		
KPI Baseline:	<i>Source:</i>	Current data gathering from engineering activities.	
	<i>Value: k₀ =</i>	Varies between activities. Exact value is not decided as it depends on the scope of the final evaluation.	
Target:	Increase:	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 25%

We expect that we can increase the amount of data that can be gathered and analysed at the architecture definition by the use of formal models that can be increasingly analysed and additionally be used for co-simulation. As such, we are expecting that the amount of available data at this stage should increase to make more advanced decision making. We expect to measure this by the amount of data that is generated in the automated processes considered in the AIDOaRt project compared to manual means over a given time (somewhere in the range of days.)

10.5. Planned improvements

The planned improvements in the VCE use case is expected to assist the work towards the commonly defined solution architecture. In particular, the following improvements are expected for the different solution components:

- Improve quality of recommendations of the MORGAN tool via more fine grained training on real gathered engineering data.
- Improvements in the visual layouts of the integrated solution architecture in Eclipse.
- Improvements in the performance of the Keptn integration to reduce resources spent on orchestration of simulation
- More structured standard views from EMF views which supports VCE engineer common problems.
- A more robust integration of MOMoT for the FMI standard by the implementation of a metamodel that captures co-simulation information with FMI.
- A more detailed and expanded data gathering from the process mining tool from JKU.

Apart from the solution components, there are planned improvements in terms of direction of the collaboration from the feedback gathered from internal workshops. An increased focus on the use of DevOps is seen as beneficial, and live demos which can be extended to a more general solution.

10.6. Planned demonstration

The planned demo at the end of the project is the commonly defined solution architecture being applied from start to finish in a realistic context. So far, each part has been demonstrated separately, but as a final product we aim for a holistic integration that can show how a user (in this case a VCE engineer) can be expected to use the framework for their needs. Additionally, we are aiming to make it possible for external users to apply the solution architecture by installation and user guidelines via publicly available sites (e.g. a public GitHub repository). We will utilise the same context that has been used so far, that is a Construction Machine, where we will highlight different parts of the system, such as the physical architecture for system modelling and the Thermal management system for simulation and analysis.

The demo should as such showcase how a system architect can model physical architectures of construction machinery via the use of different modelling languages which can represent particular VCE concerns. Additionally, it should be possible to re-use existing legacy office descriptions via automatic reading. The user should be able to employ the various integrated solutions in a semi-seamless fashion to define and model system architectures. In the modelling domain the usage of various tools should be applied for added benefits, and in particular increased automation. This includes for example model transformations, modelling recommendations, model generations, model simulation, and model management. For the simulation, we are expecting to highlight the use of heterogeneous simulation models being used together for a wider simulation, currently relying on integration based on the FMI standard. Technically, the creation and viability of FMI models are seen as out of scope for AIDOaRt, and instead the focus is on the usage, which we expect to utilise user-friendly API's. So far, this is expected to be done via cloud-based technologies that should hide the technical implementation for the end user (who is expected to be a systems engineer/architect without much simulation experience).

11. WESTMO_CS10 case study “Automated continuous decision making in testing of robust and industrial-grade network equipment”

11.1. Case Study description

The Westermo case study strives to increase the flow in the software development process of embedded systems (Figure 11.1), from humans designing and developing it (A), through the DevOps process with continuous integration and nightly testing (B,C,D,E), to the released system (F). Increasing the flow can mean many things, and in AIDOaRt the KPIs focus on reduction of effort needed for humans in functional and non-functional root cause investigations. Through the project, we achieve these goals by improving data collection (D) and introducing AI-based systems (G,J) that improve steps in the DevOps flow with monitoring, automation, and interpretation.

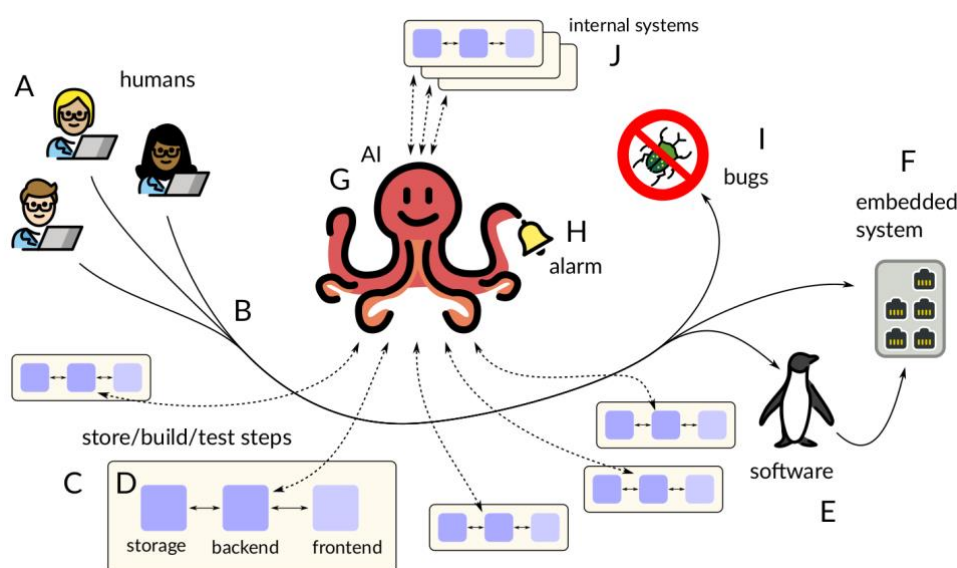


Figure 11.1 Overview of the Westermo case study

Main challenges encountered so far are on extracting and anonymizing data from Westermo, as well as collecting new types of data, at the pace initially planned. Also, evaluating and integrating solutions/tools into the Westermo context has come with challenges.

The Westermo use case has four use case stories and four requirements, but since the mapping between requirements and use case stories is not one-to-one this use case discusses requirements coverage, and also KPIs after discussing the use case stories, i.e. in Sections 11.6 and 11.7.

The *Case Study Synopsis* is in Table 11.1.

Table 11.1: Synopsis of the Westermo case study

Use case scenario W_UCS_1 — AI-Augmented devops development process	
Description:	<p>With the help of AIDOaRt solution providers, we strive to develop AI-extensions to the DevOps process such that suspicious links between code changes and test results are identified. This is developed in Python with standard AI tools like SciPy, NumPy, Pandas, and PyTorch in a containerized manner to support simple integration at Westermo.</p> <p>Another set of tools explore the test resource allocation and test selection processes. With RISE we are creating a model of the resource allocation, and ÅBO are exploring reactive test selection.</p>
Requirements:	<ul style="list-style-type: none"> ● W_R_1 — AI/ML-powered monitoring/automation of DevOps process. ● W_R_3 — Extract data from steps in the DevOps process. ● W_R_4 — Log file storing, indexing, searching, clustering and comparing
Tools:	<ul style="list-style-type: none"> ● LogGrouper (RISE) ● Bug-Inducing Commit (RISE) ● Test resource allocation (RISE) ● STGEM (ÅBO) ● CRT (Copado)
KPI:	<ul style="list-style-type: none"> ● W_UCS_1_KPI_2.2_1
Use case scenario W_UCS_2 — AI-powered root cause analysis w.r.t. functional issues	
Description:	<p>With solution providers, we strive to develop an AI-extension to the DevOps process, e.g. such that suspicious links between code changes and test results are identified. These are mainly developed in Python with standard AI tools in a containerized manner to support simple integration at Westermo.</p>
Requirements:	<ul style="list-style-type: none"> ● W_R_2 — Quality monitoring and predictions in devops process
Tools:	<ul style="list-style-type: none"> ● LogGrouper (RISE) ● Bug-Inducing Commit (RISE) ● STGEM (ÅBO) ● CRT (Copado)
KPI:	<ul style="list-style-type: none"> ● W_UCS_2_KPI_1.2_1
Use case scenario W_UCS_3 — AI-powered root cause analysis w.r.t. non-functional quality	
Description:	<p>We wish to develop tools for monitoring quality shortcomings, performance degradation, etc. To start with, more data needs to be collected. In collaboration with RISE, we explored using an existing tool set with a positive preliminary result. This has not been investigated further.</p> <p>At AIDOaRt Hackathons, we have also used Copado Robotic Testing (CRT) to investigate correlations between test case verdict data.</p>
Requirements:	<ul style="list-style-type: none"> ● W_R_2 — Quality monitoring and predictions in devops process
Tools:	<ul style="list-style-type: none"> ● Abandoned tool (RISE) ● CRT (Copado)

KPI:	<ul style="list-style-type: none"> W_UCS_3_KPI_1.2_1
Use case scenario W_UCS_4 — AI-powered log analysis/root cause analysis	
Description:	We develop a tool for clustering log files that are similar. A proof-of-concept tool has been developed and is being integrated and evaluated.
Requirements:	<ul style="list-style-type: none"> W_R_4 — Log file storing, indexing, searching, clustering and comparing
Tools:	<ul style="list-style-type: none"> LogGrouper (RISE) Bug-Inducing Commit (RISE)
KPI:	<ul style="list-style-type: none"> W_UCS_4_KPI_3.1_1

11.2. W_UCS_1 — AI-Augmented DevOps development process

W_UCS_1 aims to add AI to the DevOps development process in order to enhance it, and increase its flow (but there is some overlap between Westermo’s UCSs, in particular between W_UCS_1 and the following UCSs).

For software testing, there may be dependencies between test cases, e.g. if test case N causes a problem for a later test case N+M, e.g. by not cleaning up a resource they both use. In AIDOaRt, we have identified that (a) such correlations exist and (b) we seem to be able to identify causality as well.

Much of Westermo’s testing is done on physical hardware (and some on virtualized hardware). These test systems are a scarce resource. Currently, the test systems are manually allocated to the different software projects, but it would be desirable to introduce tool support.

11.2.1. Summary of preliminary results

Several tools link to W_UCS_1. In this report we discuss progress related to test case correlations, test resource scheduling as well as test case prioritisation. The work on LogGrouper and a tool for identifying bug-inducing commits are also related to W_UCS_1, but described under W_UCS_2.

Test case correlations

In several of the hackathons, we have explored relations between test cases, Figure 11.2 illustrates two ways of visualising relations: to the left we see a network of test cases with strong links, and to the right a matrix of all-to-all Pearson correlations. This has been explored by using tools from Copado and ÅBO. This knowledge will be relevant for the test selection described below, but was first investigated at a hackathon and considered an interesting finding in and of itself.

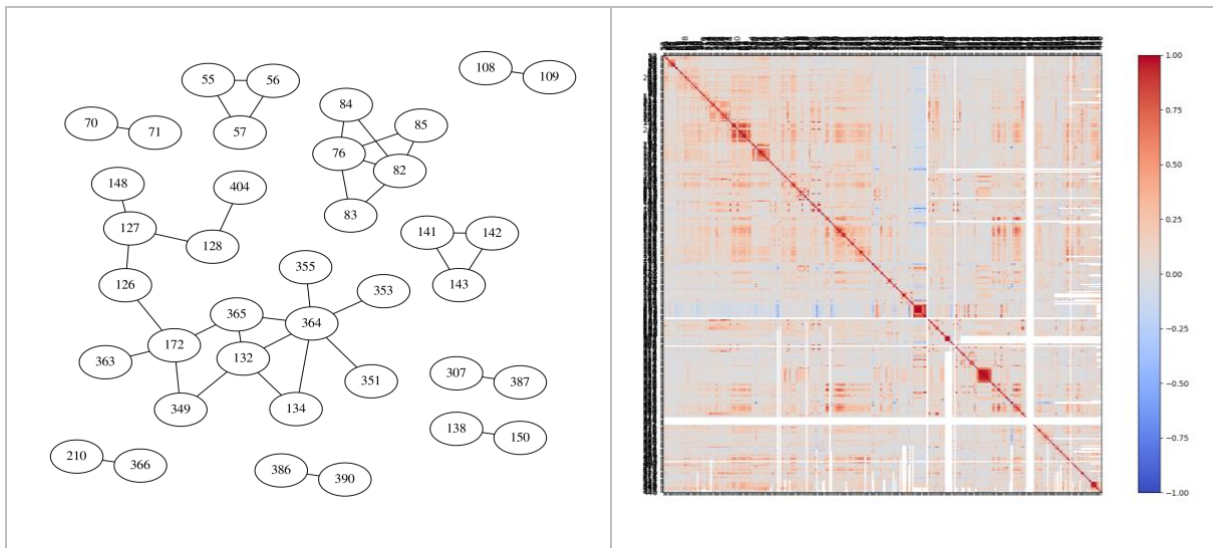


Figure 11.2 Two visualisations of relations between test cases.

Test resource scheduling

For one night in May 2023, Westermo had 12 active software branches being developed in parallel, and 30 test systems to use for testing. Most but not all branches were being tested in nightly testing, and most but not all test systems did some testing. The decisions to run testing of a certain branch on a certain test system are manual, and Figure 11.3 illustrates the user interface for doing this allocation (it has been anonymized, scaled and rotated to fit).

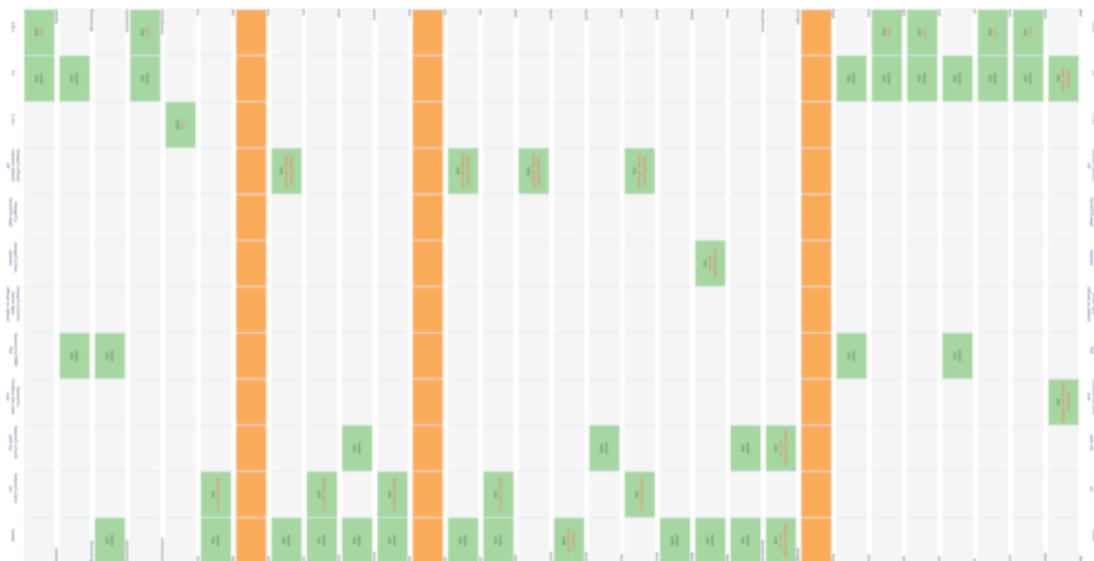


Figure 11.3 Illustration of manual test system resource allocation. Two branches (rows) are not allocated to any test system, and three test systems (columns) are not used (perhaps because of maintenance).

Intuitively, there seems to be potential for tool-support to improve the allocation process. With RISE we are exploring modelling of this process as a multi-objective optimization model whose solutions suggest the ‘best’ actions to be taken according to a given nightly testing resource allocation activity.

In particular, the model aims to maximise the hardware coverage and test coverage (in terms of tested software modules) while keeping the resource utilisation within certain thresholds.

Reactive (On-line) Test Case Prioritisation
Given the test resource allocation, we encounter the regression test selection problem: e.g., given 3 hours on test system X and 5 hours on test system Y, for testing branch A, we may select test cases. But given this allocation we might only be able to run 20-40% of the test cases. So how should we prioritise?

There are many techniques for test selection and prioritisation. However, it seems that existing techniques create a fixed test schedule before the testing starts. Using Westermo data, ABO is researching new techniques where tests are scheduled on the fly, while being executed. The idea is to first make a prioritised list, and then react to new knowledge gained during testing.

We utilise such historical data as verdicts of test cases in our algorithm. We compared our dynamic approach to the original sorting, as well to the random and the static approach based on individual probability of failure of each test. We modified our approach and gained a better performance of the dynamic algorithm. First, we create a prioritised list of test cases with some static scheduler, and then react to new knowledge gained during testing. To re-arrange pending test cases during the execution, we apply conditional probability of failure and success to re-arrange test cases. To compare our approach to some static ones, we chose Random (R), Optimal (O) and Worst (W) algorithms. As the names reveal, the random algorithm schedules tests in a random manner, Optimal creates the best possible schedule where all failing tests are prioritised first, and Worst prioritises all passing tests first. We evaluated the performance in terms of failing tests identified early in the suite, APDF (Average Percentage of Fault Detected, related to the area under the curve metric). The results are demonstrated in Figure 11.4. As we can see, the dynamic approach helps to improve the schedule created by the static ones, except for the optimal schedule, which is expected. Currently, ABO is leading a publication on this topic where Westermo's data set was used, and also two additional data sets from Company ABB Robotics Norway, see Spieker et al. [16].

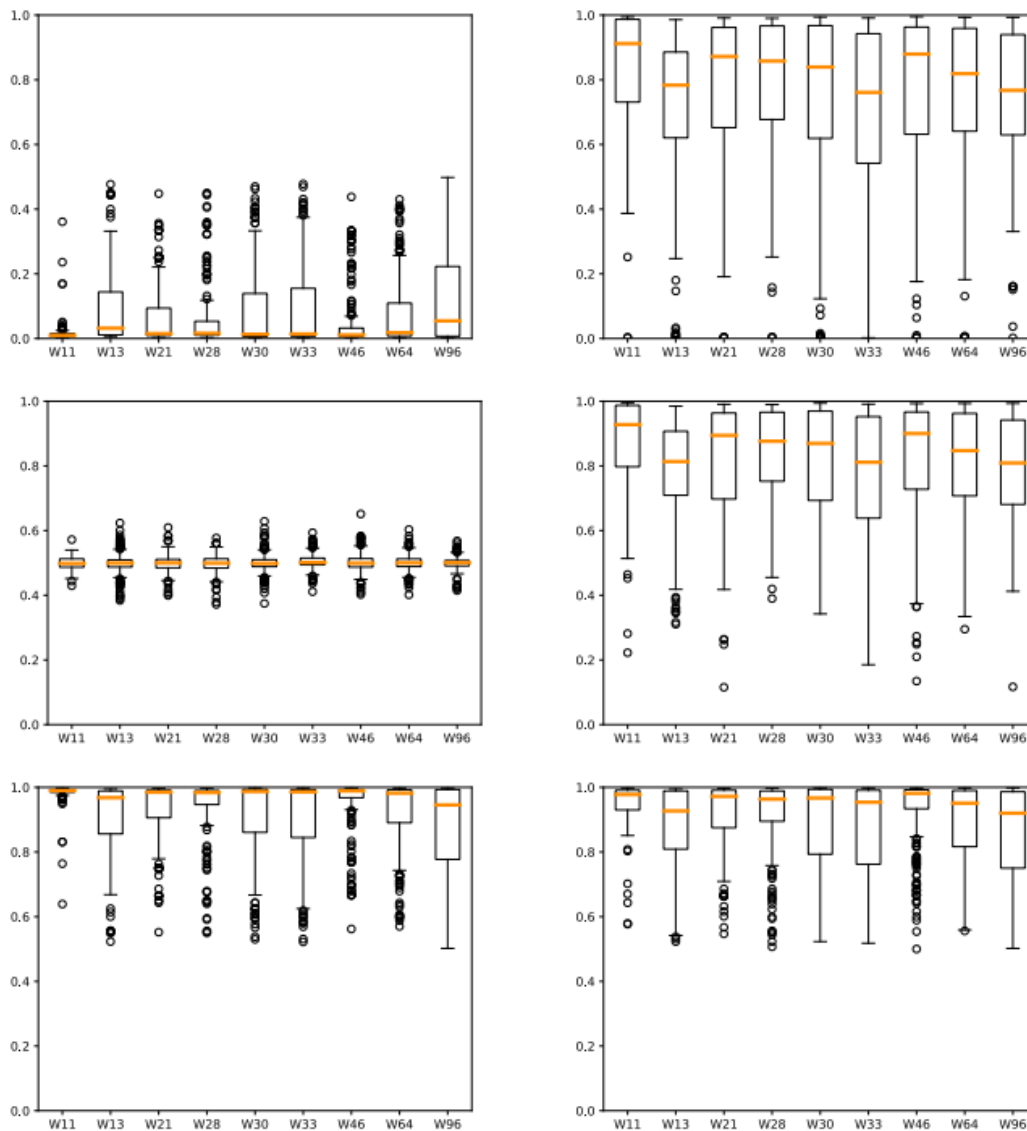


Figure 11.4 Performance in terms of APFD applied to three static approaches on all test systems (top: worst, middle: random, bottom: optimal, left: static prioritisation, right: reactive prioritisation).

11.3. **W_UCS_2 — AI-powered root cause analysis w.r.t. functional issues**

In order to speed up the DevOps feedback loop, we desire AI-support for root cause analysis of functional issues. One way to achieve this is by developing a tool that finds previously unknown links or dependencies between source code changes in the software under test, source code changes in the test framework, and changes in verdicts of test cases. With RISE, we are exploring two tools. First, a tool that identifies similarities in test case executions -- i.e. test cases that failed in the same way are

expected to have the same root cause. Second, a tool that also uses information from code changes to identify and rank links from failing tests to bug-inducing commits.

11.3.1. Summary of preliminary results

The software development process at Westermo is feature driven, this means that many small software teams work in parallel branches of the software. Each of the branches is developed and then tested in isolation. The testing is performed nightly on a fleet of heterogeneous test systems. Each night, there are typically thousands of verdicts produced, and each test execution produces anything from a handful to dozens of log files. For this reason, RISE developed a proof-of-concept implementation of LogGrouper with the goal to simplify the exploration of test results by grouping log files that seem to show the same error instantiated by several different test cases, possibly on different test systems, and maybe even on different code branches.

cluster_number	number_of_items	outcomes_id	Generated 2023-03-08 08:00
0	12	<ul style="list-style-type: none"> 62598800 62603636 62599572 62598950 62598835 62598965 62599131 62600860 62600967 62603677 62598905 62599502 	<ul style="list-style-type: none"> packets max allowed protocolX lb eth stream packet checks failed test protocolX fail packets max allowed protocolY lb eth stream packet checks failed test protocolY fail packets max allowed protocolZ lb eth stream packet checks failed test
1	9	<ul style="list-style-type: none"> 62603209 62598741 62599138 62601873 62592222 62598910 62578918 62598860 62599388 	<ul style="list-style-type: none">
2	5	<ul style="list-style-type: none"> 62599009 62598984 62577869 62577711 62608876 	<ul style="list-style-type: none">

Figure 11.5 Anonymised screenshot of one clustering.

In Figure 11.5, we see the current status of LogGrouper. It is running in Westermo’s environment, produces clusterings using on-line data. However, work on parameter settings for strictness of the grouping, and the integration with other systems has not been finalised.

The LogGrouper approach is evaluated at Westermo with three extracted data windows i.e. nightly failure logs, weekly, and all failure logs using two evaluation metrics as follows. Silhouette Coefficient, a value between -1 and 1 that represents the degree of separation between the resulting clusters of



failures. Calinski-Harabasz Index, a criterion for evaluating failure clustering techniques that lack ground truth labels (alternatively called the Variance Ratio Criterion). Based on the evaluation metrics, we found that DBSCAN and K-Means performs better in grouping failure logs and could aid functional root cause analysis and failure assignment.

As an extension of LogGrouper, we are with RISE working on the Bug-Inducing Commit (BIC) tool that employs a natural language processing technique and heuristics to conduct log-level analysis for the purpose of filtering out relevant attributes associated with each commit. The first step (see Figure 11.6) of the approach is to perform preprocessing of git log files extracted from the Westermo Operating System (WeOS), Fawly test framework, and test execution logs. The aim is to parse the data to introduce consistency and make sense of the logs for extracting the relevant features and ranking the bug-inducing commits based on their correlation. The filtering mechanism is based on two parameters: date range and specific feature branch of the source code changes (commits). Once the search space is reduced based on the filtering parameters, the relevant extracted features for WeOS and Fawly framework (commit messages, the number of additions, and subtractions in source code, etc.) are associated with each commit and test execution logs (when filtering out critical and error log entries). The ranking of the commits rely on the correlation between the extracted features.

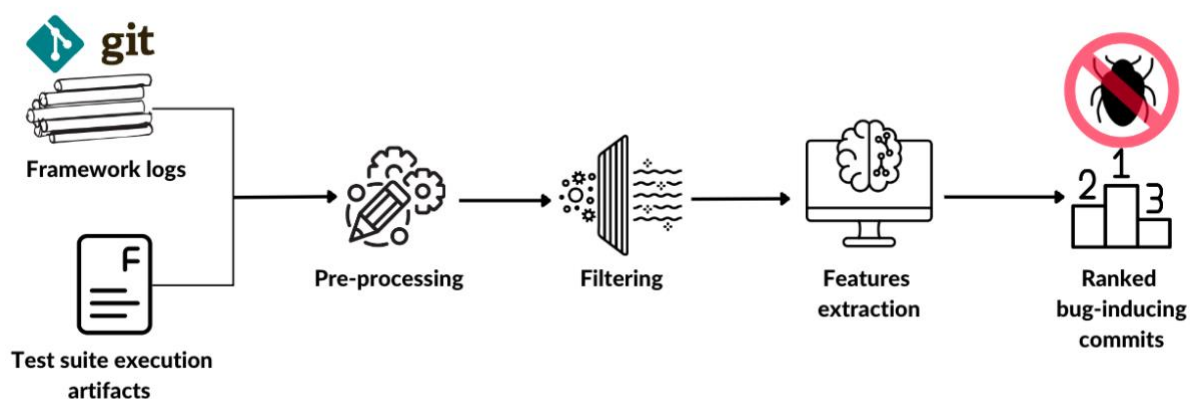


Figure 11.6 Overview of Bug-Inducing Commit workflow.

Currently, a first complete prototype of the tool has been developed (see example in Figure 11.7). Based on the input filters (date and branch) the tool suggests code changes that might have caused the failing tests. For readability, the figure only illustrates commits from the test framework, but a similar list with code changes from the software under test is available by scrolling. The tool will be evaluated with intrinsic evaluation metrics in the operational environment.

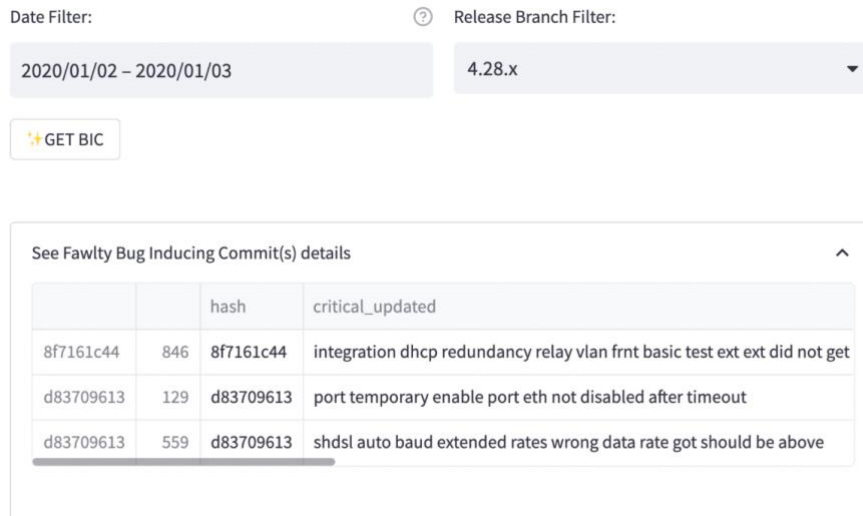


Figure 11.7 Overview of Bug-Inducing Commit workflow.

11.4. **W_UCS_3 — AI-powered root cause analysis w.r.t. non-functional quality**

In the DevOps process at the company, many systems and services are in use. On the servers that drive the nightly testing, we have started collecting performance data (prior to AIDoArT this data was not collected systematically). One challenge for Westermo is when a server degrades or has some problems. With several partners, and on several hackathons, we have explored tools for anomaly detection in this type of performance data. In addition, there is also some performance data coming from the test results database, in particular the duration of test case execution (e.g., time series data of the duration of a test case execution of a test case night after night on one test system as the software under test and test framework evolves).

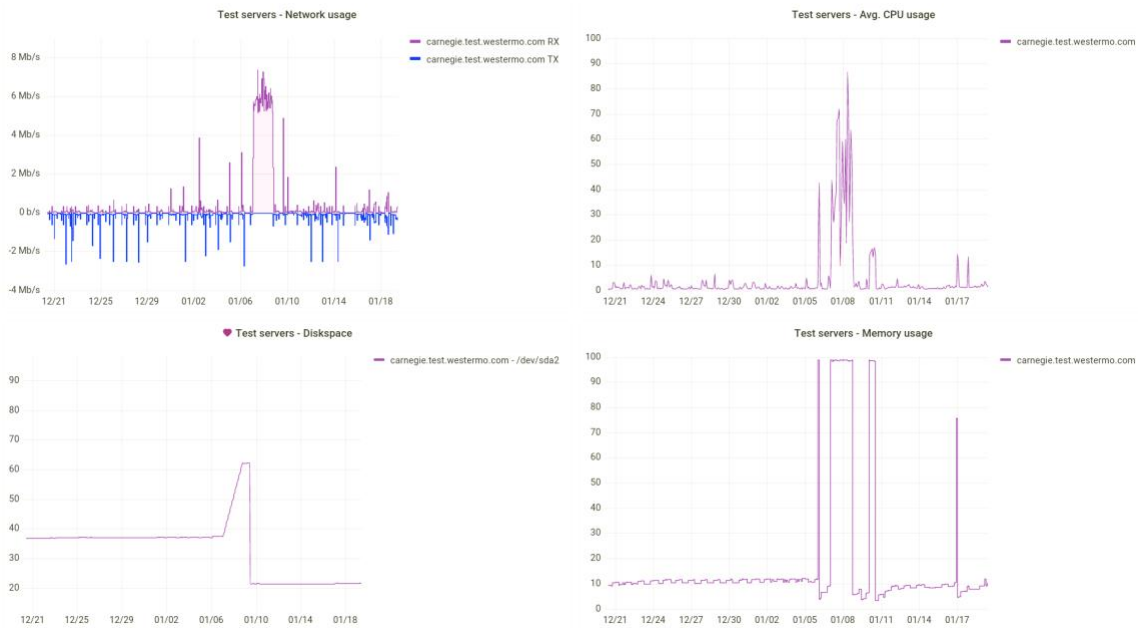


Figure 11.8 Extract of test system performance metrics visualised with Grafana

In Figure 11.8, metrics from one server in one of the test systems in use at Westermo, is illustrated. As can be seen, during the second third of the test timespan, a problem occurred: an uncontrolled network traffic was ongoing while at the same time, the traffic was recorded to file. This led to extreme CPU and memory usage as well as a disk starting to be full. After the weekend, a colleague corrected the issue, deleted undesired files and restarted the system.

11.4.1. Summary of preliminary results

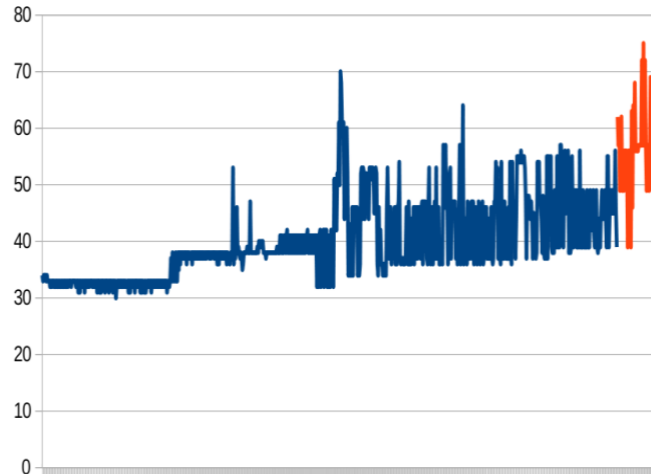


Figure 11.9 Illustration of the duration in seconds (Y-axis) of one test case over time (X-axis). The change in colour shows then the COPADO tool identified a change in duration

During the second AIDOaRt hackathon, Copado used their CRT tool to explore the Westermo test results dataset for identification of changes in the test execution time trends. A change point in this signal can be found with the statistical properties of the signal, see Figure 11.9. This illustrates that there are indeed change points in the performance of the test cases at Westermo, and that tools such as CRT identified them.

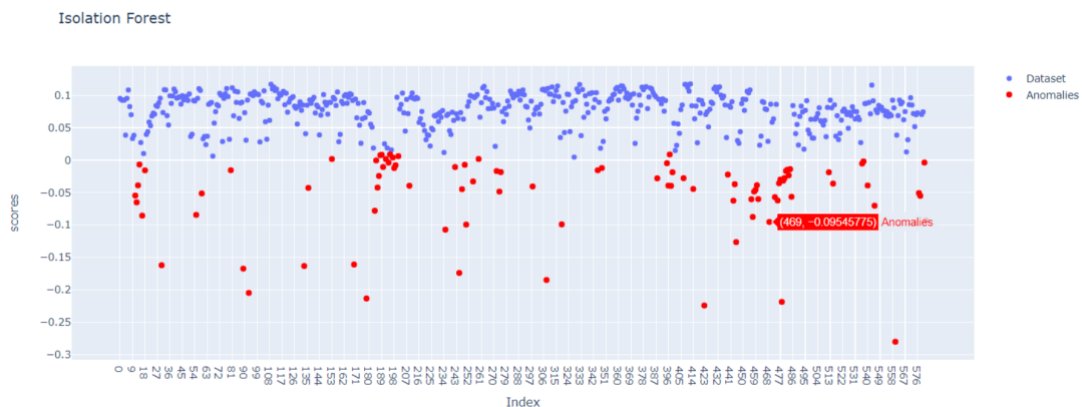


Figure 11.10 Isolation forest plot of metrics from one test system (anomalous stats in red)

During the summer of 2022, in collaboration with, we RISE analysed test system metrics for anomaly detection, based on a findings in a Master’s thesis. Using Principal Component Analysis (PCA) for dimensionality reduction, clustering approaches, and statistical analyses, anomalous states of the test systems were identified. Figure 11.10 illustrates one way of visualising the anomalies with an isolation forest plot.

For the fourth AIDOaRt Hackathon, Westermo prepared a dataset of performance data from about 20 test systems, with about 20 time series (load, CPU usage, memory usage, etc.) sampled minute by minute for a month, resulting in almost 320 MB of data in CSV format.

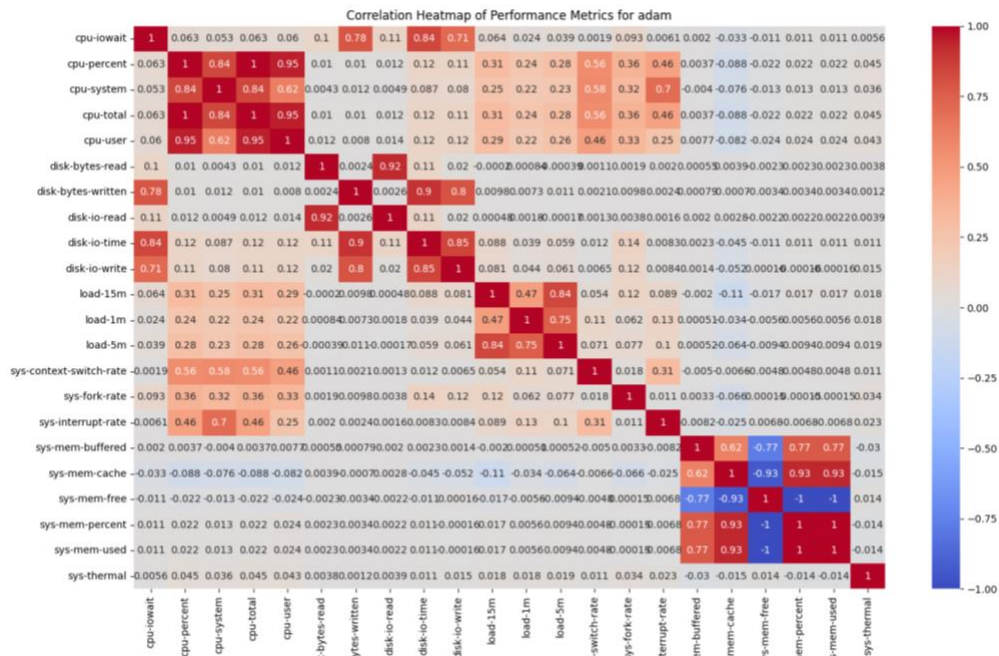


Figure 11.11 Pearson correlations between time series for one test system

When Copado analysed this data, they explored correlations, peak values, saturation, change points and methods for anomaly detection. To our surprise, they identified that some clusters of related time series were not correlated to each other. E.g. metrics related to CPU strongly correlated (red area in top left of Figure 11.11), metrics related to memory usage also had correlations (left right in same figure), but CPU usage *was not* correlated to memory usage (left bottom and top right in Figure 11.11).

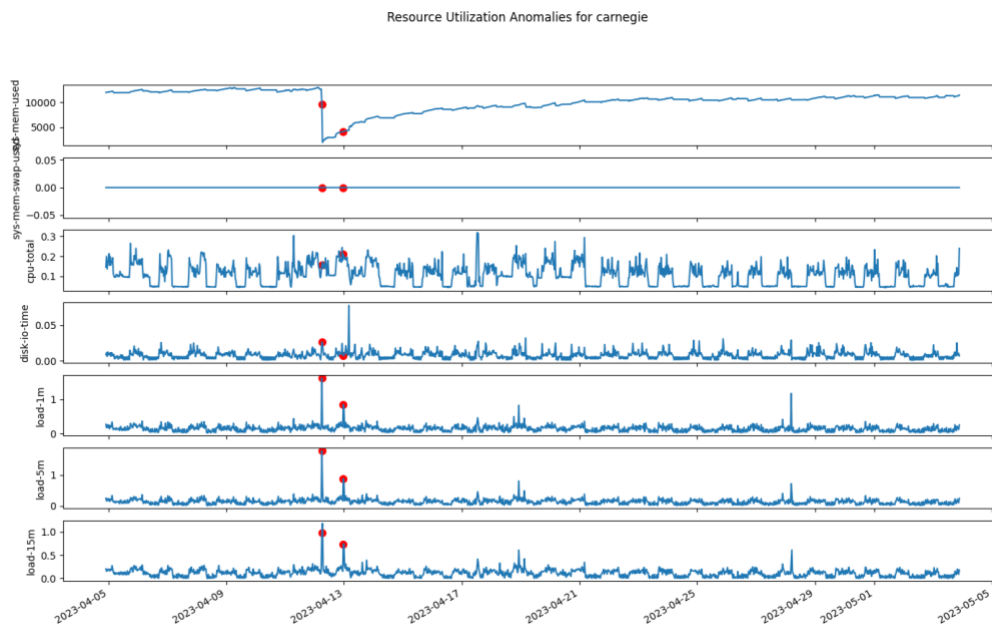


Figure 11.12 Anomaly detection with isolation forest

Copado also explored anomaly detection, and again, isolation forest seemed to be a feasible method (see Figure 11.12) in particular when using more than one time series together.

So far, none of these tools have reached beyond proof-of-concept, and no tool is integrated at Westermo. However, the Hackathons have shown that there is most likely great value in using these types of statistical analyses, and if this data could be combined with more data (e.g. information on what the test systems are doing when this data is being collected), then more sophisticated AI-based tools could also be meaningful.

Furthermore, these types of analyses are not present in tools currently offered by Copado. Results from the Hackathons could therefore indicate new potential venues for commercialization.

11.5. W_UCS_4 — AI-powered log analysis/root cause analysis

At Westermo, nightly automated regression testing is an important activity for quality control. A repetitive and time consuming task is for humans to investigate and explore test results from nightly testing by reading log files. This use case scenario aims at enhancing this activity. As described above (W_UCS_2), we use log analysis in the LogGrouper tool, and no other tool specifically targets this UCS.

11.5.1. Summary of preliminary results

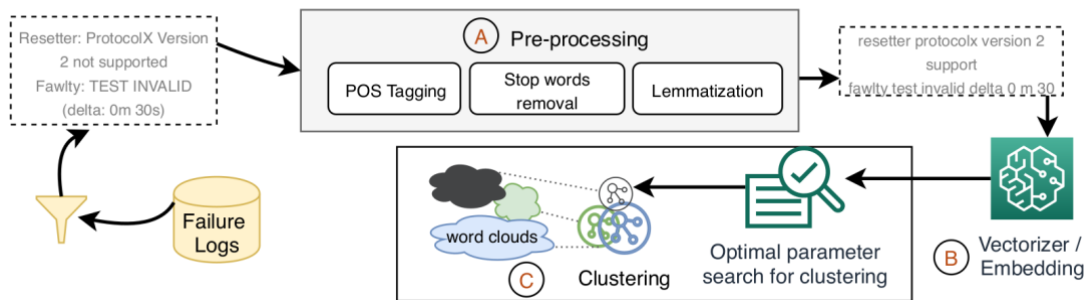


Figure 11.13 Overall process of LogGrouper.

In the LogGrouper tool (see Figure 11.13), log files are processed with (A) pre-processing, where logs are converted to lower-case, timestamps are removed, tokens were lemmatised, etc. the input and output of pre-processing is illustrated with a few lines of log in dashed boxes in the figure. Next, (B) for vectorisation, the tool supports tfidf, Fast-Text, and BERT. For vector dimensionality reduction, LogGrouper uses PCA. For the clustering, we evaluated DBSCAN, kMeans, spectral and agglomerative, where DBSCAN performed best (kMeans was sometimes better), and spectral has the poorest performance.

11.6. Requirements Coverage

The Westermo use case has four use case scenarios described in detail above. The use case also has the following four requirements:

- (W_R_1) AI/ML-powered monitoring/automation of DevOps process.
- (W_R_2) Quality monitoring and predictions in devops process.
- (W_R_3) Extract data from steps in the DevOps process.
- (W_R_4) Log file storing, indexing, searching, clustering and comparing.

The links between use case scenarios and requirements are mapped in Table 11.2:

Table 11.2: Mapping of Westermo use case scenarios and requirements

	W_R_1	W_R_2	W_R_3	W_R_4
W_UCS_1	X		X	X
W_UCS_2		X		
W_UCS_3		X		
W_UCS_4				X

Since the mapping between requirements and use case stories is not one-to-one this use case discusses requirements coverage in the below subsections, requirement by requirement.

W_R_1 - AI/ML-powered monitoring/automation of DevOps process

W_R_1 is a high-level requirement on introducing AI/ML in the DevOps process at Westermo. It could almost be considered a parent of W_R_2, W_R_3 and W_R_4, and all tools evaluated for the Westermo use case cover it. In particular, the solutions LogGrouper and BIC both address W_R_1 in that they use AI/ML to speed up the process of fault-finding.

Tools by ÅBO, Copado and RISE to automate detection of performance anomalies could also be considered as covering this requirement since they would reduce the number of stops of productivity in the DevOps process (e.g. if a test system server has a full disk and must be manually fixed).

Furthermore, a test resource allocation (RISE) and an on-line test prioritisation tool (ÅBO) would also increase the level of automation and improve resource usage at Westermo. Thereby also covering W_R_1.

To conclude, this requirement is well covered.

W_R_2 - Quality monitoring and predictions in devops process

With AI/ML, Westermo expects to enable monitoring of different quality attributes (both functional like failing tests, and non-functional like performance), thereby allowing the introduction of various alerts and predictions. In turn, this is expected to lead to increased flow in the development process. For quality monitoring, tools to detect performance anomalies (in particular tools by Copado, and the tool abandoned by RISE) targets this requirement well.

So far, these tools have worked on off-line data, and only been evaluated at Hackathons or by an abandoned tool by RISE. For this reason, the requirement has been targeted, but not as well as W_R_1, and no tool has left the stage of proof-of-concept.

W_R_3 - Extract data from steps in the DevOps process

To support the flow of the entire DevOps process, each step ought to have data that can be stored and extracted in suitable formats. This includes increasing the number of data sources, as well as making the data more accessible. This has been targeted by work at Westermo, e.g. by collection of performance data from test system servers, as well as by processing log files.

This requirement has been targeted to some extent, but more data should be collected (see e.g. W_UCS_1_KPI_2.2_1 for a deeper discussion).

W_R_4 - Log file storing, indexing, searching, clustering and comparing

From the development process, a large amount of data in the form of log files are produced. These ought to be stored (and perhaps indexed) so that a human or an AI could search in them, or use them in other ways. This requirement is targeted specifically by W_UCS_4 and the tools LogGrouper and BIC that make use of logs. Parts of this requirement are well covered, but more work could be done, e.g. on log exploration such as the scenario where the system would say ‘engineers that looked at this log also looked at these other logs’.

11.7. KPI Evaluation

W_UCS_1_KPI_2.2_1 — More collected logs

Table 11.3: W_UCS_1_KPI_2.2_1 — More collected logs

KPI Identifier: W_UCS_1_KPI_2.2_1		Scenario Identifier: W_USC_1	
KPI Description:	Increase of the number of available log sources for enabling AI-Powered monitoring, root causing, prediction, etc.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the number of available data sources to be actually managed and handled in existing engineering practices.	
	<i>Identifier:</i>	KPI_2.2	<i>Target</i> ≥ 25%
KPI Measure:	k = Number of log types collected and managed in the DevOps process		
KPI Baseline:	<i>Source:</i>	Discussions with the test framework and DevOps team.	
	<i>Value: k₀ =</i>	6 (see below)	
Target:	<i>Increase:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k - k_0) / k_0$	≥ 25%
D5.6 Measure:	<i>Value: k₆ =</i>	7	$100 \cdot \Delta_6 / k_0 = 100 \cdot (7 - 6) / 6 =$ 17%
D5.7 Measure:	<i>Value: k₇ =</i>	7	$100 \cdot \Delta_7 / k_0 = 100 \cdot (7 - 6) / 6 =$ 17%

This KPI measures the increase in number of log sources Westermo collects. As an example: before AIDOaRt we did not collect performance metrics from the PCs running our test framework, but we do this now. This is an increase of log source by 1. However, we measure this KPI as being relative to the number of log formats we had before AIDOaRt.

Log sources collected before AIDOaRt were: (1) Compilation log, from when a compiler in the DevOps chain compiles the software. (2) Static code analysis logs, that looks for memory leaks, etc. (3) Test framework log, from when the test framework is running test cases. (4) Device communication log (from when the test framework communicates with devices). (5) Source code history (git log) from the software under test (WeOS), (6) git log from the test framework.

Log sources now collected (thanks to AIDOaRt): (vii) Performance metrics from PCs running the test framework.

This KPI has not changed since D5.6. Log sources we hope to also collect during AIDOaRt: (viii) Performance metrics from devices under test. (ix) Remote logging from devices under test to the test framework. Currently, we design test systems with the assumption that we can only trust the console connection (a serial connection for configuration and other low level things). One could imagine changing the design of test systems such that remote logging would be possible over ethernet. Then

the test framework could consume the remote logs and react to events from these logs that would otherwise potentially be lost. Finally, (x) Crash logs details from devices under test: if an application crashes while testing, stack traces and/or memory dumps could be collected that could greatly speed up debugging, and potentially enhance an AI.

W_UCS_2_KPI_1.2_1 — Faster root causing of functional issues

Table 11.4: W_UCS_2_KPI_1.2_1 — Faster root causing of functional issues

KPI Identifier: W_UCS_2_KPI_1.2_1		Scenario Identifier: W_USC_2	
KPI Description:	Improvement of the time needed to root cause functional issues.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the early detection of system deviations	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> ≥ 30%
KPI Measure:	k = Average amount of time spent per functional root cause investigation.		
KPI Baseline:	<i>Source:</i>	See details below.	
	<i>Value: k₀ =</i>	See details below.	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$ ≥ 30%	

The challenge with this KPI (and the next) is to accurately measure the amount of time spent in root cause investigations, and to compare a situation at the start of the project with another situation at the end of the project. Confounding variables could be, e.g. that colleagues have learned to recognize certain types of problems and that the decrease in time is caused by this.

Therefore, in order to measure this progress, Westermo plans to use a proxy metric: to observe practitioners when they are conducting root-causing using tools developed under AIDOaRt, and to ask how much slower the root-causing would have been without these tools. We hope to conduct these measurements at least for D5.8 and D5.9.

W_UCS_3_KPI_1.2_1 — Faster root causing of non-functional issues

Table 11.5: W_UCS_3_KPI_1.2_1 — Faster root causing of non-functional issues

KPI Identifier: W_UCS_3_KPI_1.2_1		Scenario Identifier: W_USC_3	
KPI Description:	Reduction of the person time needed to be invested in root cause investigations of non-functional quality issues.		
Refined AIDOaRt KPI:	<i>Description:</i>	Improvement of the early detection of system deviations	
	<i>Identifier:</i>	KPI_1.2	<i>Target</i> ≥ 30%
KPI Measure:	k = Average amount of time spent per non-functional root cause investigation.		
KPI Baseline:	<i>Source:</i>	See details below.	
	<i>Value: k₀ =</i>	See details below.	
Target:	<i>Decrease:</i>	$100 \cdot \Delta / k_0 = 100 \cdot (k_0 - k) / k_0$ ≥ 33%	

Again, this KPI (like the previous one) is to measure effort spent in non-functional root cause investigation. An important difference is that data collection for quality attributes (e.g. test framework

server CPU usage) was absent prior to AIDOaRt. As discussed above, Westermo now collects data of this type in an automated manner. Thus, events can now be monitored in ways that were impossible prior to AIDOaRt. So despite not yet having a well-defined process to measure this KPI, we believe we are on the right track. Again, we hope to conduct these measurements at least for D5.8 and D5.9.

W_UCS_4_KPI_3.1_1 — Log Clustering

Table 11.6: W_UCS_4_KPI_3.1_1

KPI Identifier: W_UCS_4_KPI_3.1_1		Scenario Identifier: W_USC_4	
KPI Description:	Implementation of log clustering for some of the log types available at start of project.		
Refined AIDOaRt KPI:	<i>Description:</i>	Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).	
	<i>Identifier:</i>	KPI_3.1	<i>Target</i> ≥ 30%
KPI Measure:	k = percentage of log types available (at start of project) with clustering		
KPI Baseline:	<i>Source:</i>	At the beginning of the project, no log clustering was implemented.	
	<i>Value: k₀ =</i>	0%	
Target:	<i>Increase:</i>	k ≥	30%
D5.6 Measure:	<i>Value: k₆ =</i>	17	100·k ₆ = 17%
D5.7 Measure:	<i>Value: k₇ =</i>	50	100·k ₇ = 50%

At the beginning of the project, there were six log types stored systematically and explored as part of nightly testing. Currently, with the LogGrouper tool, one such log type has a proof-of-concept implementation of clustering, and logs from both the software under test and the test framework are used in the BIC tool. So this KPI is achieved. However, the BIC tool does not exactly cluster (but it uses the information in other ways), and the tools have not been fully evaluated.

11.8. Planned improvements

When it comes to planned improvements, first, the top priority for Westermo is to continue working on refining the LogGrouper and the Bug-Inducing commit tools, and if possible to integrate them more in our DevOps environment. Second, if AI-enhanced tools for monitoring performance data from test systems (or in the future, the devices under test) would show promising results, then a similar effort to integrate those tools could also be relevant. Third, for the topics of test correlations, resource allocation and test prioritisation, the knowledge or tools that come out of these collaborations is valuable and might be explored more in future projects.

From the perspective of other partners in AIDOaRt, work on the Westermo use case could produce insights of scientific value or tool improvements, as has been shown thus far in the project, and in particular in Hackathons.

Here are the planned improvements, tool by tool:

- **Test Correlations:** No tool is currently in use at Westermo for test correlations. Given that Westermo has released a test results dataset that identifies correlations, we would welcome future investigations of this.
- **Performance Metrics:** Again, no tool is currently in use at Westermo for monitoring performance metrics. We hope to be able to release an anonymized dataset of performance data to help stimulate exploration of new tools.
- **LogGrouper (RISE):** A prototype implementation of LogGrouper by RISE has been delivered to Westermo. A simple front-end has been implemented that interfaces with LogGrouper and just shows whatever it produces, see Figure 11.5. Future steps are to improve the explanations of the clusters, and explore parameters such as strictness of clustering. Currently, there are many irrelevant words that have to be added to the stop list, and Westermo should probably also improve the contents of the logging to improve the tools performance. Finally, the clustering ought to be integrated into systems already in use, instead of being an external tool.
- **Bug-Inducing Commit (RISE):** The plan is to improve the ranking mechanism of Bug-Inducing commits by considering more relevant test logs execution features using Machine Learning. Furthermore, there are plans to further investigate the prioritisation of resolving a failing test case among multiple failing test cases sharing a similar commit, but with different execution logs.
- **Test resource allocation (RISE):** The problem definition (and optimization model formulation) and the data set preparation are on-going activities. Working meetings are organised with RISE for defining (a) the Westermo resource allocation problem (state-of-practice analysis) and (b) formulating the optimization model (state-of-the art analysis). As next steps, an interesting research direction that we intend to investigate concerns an empirical study performed by using Westermo historical testing data.
- **STGEM (ÅBO):** For on-line test prioritisation, ÅBO are considering implementing additional performance metrics for evaluation and comparison as well as other techniques that could improve the performance of the on-line algorithm.
- **CRT (Copado):** The functionalities developed and showcased during the hackathon hold significant value to Copado. As a result, Copado is meticulously strategizing to integrate these features into CRT. These enhancements are anticipated to broaden the accessibility and adaptability of Copado's CRT solution across diverse use cases, thereby elevating its inherent value and potential for end users.

11.9. Planned demonstration

Given that the collaborations for the Westermo use case are somewhat heterogeneous, we aim to provide two main tracks of demonstrations:

- First, a demonstration of test results exploration. Typically, Westermo staff looks at test results from nightly testing in the mornings to understand the state of the software under test, or if there are problems with the test framework. In this scenario, we expect that the LogGrouper

and the Bug-Inducing Commit tool would both illustrate improvements from AIDOaRt. If possible, we would like to also demonstrate improvements based on findings and/or anomalies with respect to test correlations and performance metrics in the same scenario. However, we do not expect to integrate tools into the Westermo DevOps process, instead these tools will likely be off-line tools.

- Second, during AIDOaRt, we have collaborated on tools that we expect to be less mature, or would have a longer distance to integrate into Westermo. These will probably be demonstrated off-line. We can expect three sub-groups:
 - Improvements in CRT (Copado). We can expect that some but not all of the work on CRT during AIDOaRt is related to Westermo.
 - The model for optimising value in test resource allocation could be demonstrated with off-line data, and compared to the manual process currently in place.
 - On-line test prioritisation could be demonstrated using the test results data set, and compared to random ordering (and the original ordering in the data set).

12. Conclusions

This deliverable reports the status, at the project month M30, of evaluation of development within individual case studies of the AIDOaRt.

In the deliverable, the “case study provider” partners report on the evaluation of progress made in the development of each case study. In particular, they describe (and try to quantify where applicable) how use of AIDOaRt tools have contributed to solving some of the challenges they face in each development scenario, as well as the plans to solve the remaining ones. The “solution provider” partners contribute to this description and recall intermediate evaluation of the progress of their work. The deliverable is a starting point for the next development phase.

KPIs — this deliverable updates the measurements based on specification of the *case studies KPIs* (Key Performance Indicators), or adding baseline for the remaining KPIs. All these indicators were derived from the *project KPIs*, through specialisation and adaptation of the latter to the system developed in the case study, to the partner interests and processes, and to the used tools.

In this deliverable, each partner reported baseline or updated measurements of multiple KPIs. The number of indicators per partner is shown in Figure 12.1.

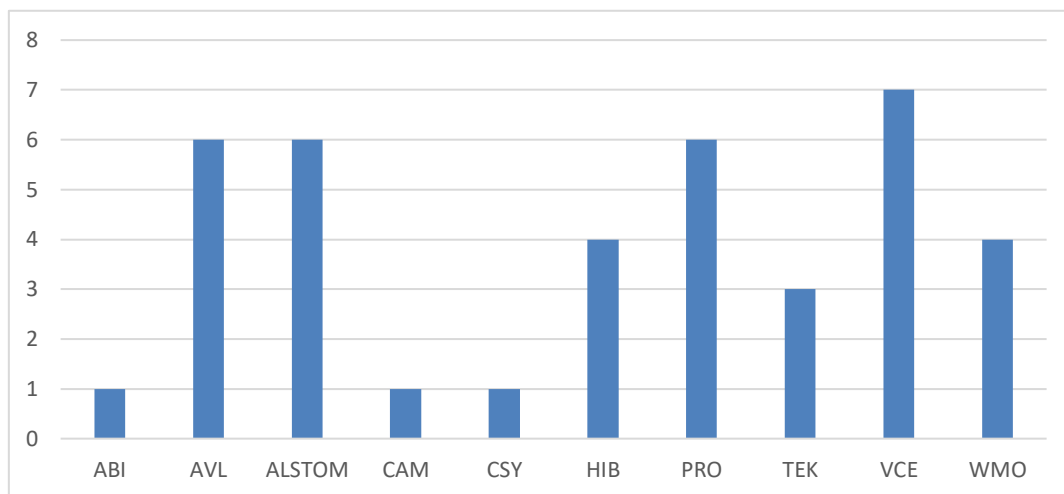


Figure 12.1 Number of KPIs per project partner

For each project level KPI, Figure 12.2 shows the number of measurements of the case study KPIs that derive from the former. As well, number of already measured KPIs is added to the chart.

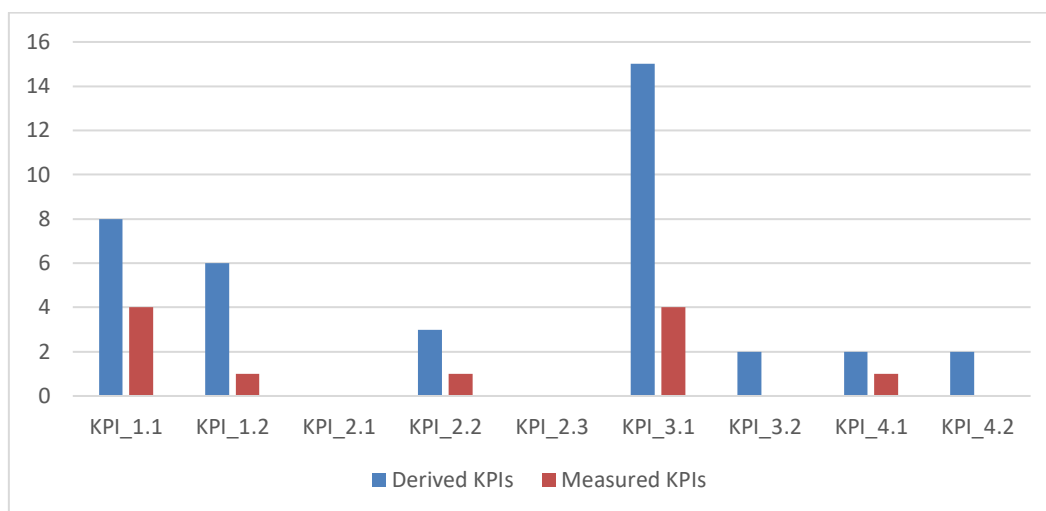


Figure 12.2 Project KPIs statistics

It can be seen that there is no measurement for KPI_2.1 and KPI_2.3 at the moment. On the other side, 4 KPIs are already fulfilled meeting or overcoming the target. These were covered by HIB, PRO and WMO. The rest seems to be on good track – another 8 measurements were done including explanation of proposed improvements for the next development phase. On the other hand, the rest of KPIs with baseline explains in detail status and progress and it is obvious that most of them will be finished during the next development phase and measured during the next evaluation phase. For almost all KPIs, the total number of measurements is higher than the number grouped by case study, because there are case studies that address the same KPI in more than one scenario.

As well as in D5.6 [9], KPI_3.1 is still the most used since it is related to the automation of the global development process: “Increase in the percentage of the automated parts of the processes which are currently manual (e.g. predictive maintenance, generation of test cases).” KPI_1.1 and KPI_1.2 are the second most used. The first one is defined as “Improvement of the time required for identification of design problems thanks to the analysis of the collected data.” The second one is defined as “Improvement of the early detection of system deviations.”

Planned improvements— In the next months, by using the improvements of the tools available in the meantime, there will be further improved development of the case studies that allow final evaluation, with the support of the updated measurements of KPIs. These results will be reported in D5.8 [10] to be released before the end of the project (M34).

Planned demonstrations — In the next months, by using the improvements of the tools available in the meantime, the development of the case studies will be brought to the final state that allows evaluation of the project final outcomes, with the support of the last measurements of KPIs. These results might be reported in D5.9 [11] to be released at the end of the project (M36).

13. Referenced documents

AIDOaRt deliverables

- [1] D1.1 Use Cases Requirements Specification.
- [2] D1.3 Use Cases Scenarios and Evaluation Criteria Definition.
- [3] D1.2 Architecture Specification Initial Version.
- [4] D1.4 Architecture Specification Final Version.
- [5] D2.3 Data Collection and Representation - Final Version
- [6] D4.1 AIDOaRt AI-Augmented Toolkit Initial Version.
- [7] D5.2 AIDOaRt Integrated Framework - Initial Version.
- [8] D5.5 Use case Requirements and Scenarios Evaluation Report.
- [9] D5.6 Use Case Development Report – 1.
- [10] D5.8 Use Case Development Report – 2 (will be released at the project month 34 = Jan 2024).
- [11] D5.9 Use Case Evaluation Report – 2 (will be released at the project month 36 = Mar 2024).

Other sources

- [12] X. Zheng., L.K. John, & A. Gerstlauer, LACross: Learning-Based Analytical Cross-Platform Performance and Power Prediction. *Int J Parallel Prog* 45, 1488–1514 (2017). <https://doi.org/10.1007/s10766-017-0487-0>
- [13] O. Bringmann, W. Ecker, A. Gerstlauer, A. Goyal, D. Mueller-Gritschneider, P. Sasidharan and S. Sing: “The next generation of virtual prototyping: Ultra-fast yet accurate simulation of HW/SW systems”, *proc. of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015.
- [14] J. Redmon, and A. Farhadi, “YOLOv3: An Incremental Improvement”, 2018. <https://arxiv.org/abs/1804.02767>
- [15] A. Geiger, P. Lenz, C. Stiller, R. Urtasun. (2013). “Vision meets robotics: The kitti dataset”. *The International Journal of Robotics Research*, 32(11), 1231-1237.
- [16] H. Spieker, A. Gotlieb, D. Marijan, & M. Mossige (2017, July). Reinforcement learning for automatic test case prioritization and selection in continuous integration. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 12-22).
- [17] A. Andrieux et al., "Web Services Agreement Specification (WS-Agreement) (v. gfd-r.192)," 2011. OGF - Grid Resource Allocation Agreement Protocol WG.