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

D5.5 Use Cases Requirements and Scenarios Evaluation Report

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.

Contract number:	101007350
Project acronym:	AIDOaRt
Project title:	AI-augmented automation supporting modelling, coding, testing, monitoring and continuous development in Cyber-Physical Systems
Delivery Date:	March 31, 2022
Authors:	Mehrdad Saadatmand (RISE), Bilal Said (Softeam), AIDOaRt contributing partners
Contributing Partners:	WP1 & WP5 Partners
Date:	March 30, 2022
Version	V1.0
Revision:	05
Abstract:	This deliverable is the first of a series of deliverables, namely D5.5, D5.7, and D5.9, reporting on the evaluation results of AIDOaRt technologies and solutions applied on the industrial case studies and use-case scenarios of the project. For each case study, D5.5 in particular includes the description of target testbeds, validation methodology, experimentation plan, along with the first set of KPIs and metrics to measure the improvements expected from the application of AIDOaRt technologies and solutions.
Status:	Type: R, Dissemination Level: PU

DOCUMENT REVISION LOG

VERSION	REVISION	DATE	DESCRIPTION	AUTHOR
V1.0	01	Feb 10, 2022	Initial draft, expected Table of Content	Mehrdad Saadatmand
V1.0	02	March 14, 2022	Structure and sections set, first draft of contents for deliverable sections provided	Mehrdad Saadatmand, Bilal Said, + all contributing partners
V1.0	03	March 21, 2022	Includes fixes after the initial review	Mehrdad Saadatmand, Bilal Said, + all contributing partners
V1.0	04	March 28, 2022	Includes fixes after the second round of reviews	Mehrdad Saadatmand, Bilal Said, + all contributing partners
V1.0	05	March 30, 2022	Polished for final submission	Mehrdad Saadatmand

Executive Summary

This deliverable is the first of a series of deliverables, namely D5.5, D5.7, and D5.9, reporting on the evaluation results of AIDOaRt technologies and solutions applied on the industrial use cases of the project. This is done as part of Task 5.3 on use-case execution and evaluation, and the aforementioned deliverables are considered outputs of this task. The current deliverable (D5.5), in particular, focuses on: establishing and identifying the baseline of the use cases, description of testbeds and demonstrators on which evaluation of AIDOaRt technologies will be done, description of evaluation and validation methodologies, and most importantly defining the first set of KPIs and metrics to measure the improvements with respect to case study challenges thanks to the application of AIDOaRt technologies and solutions.

Table of Contents

Key Terminology Abbreviations

Abbreviations	Terminology
AI	Artificial Intelligence
AIOps	AI Operations
CPS	Cyber-Physical Systems
CPSoS	Cyber-Physical Systems of Systems
DevOps	Development Operations; a set of practices combining Software Development and IT Operations
FPP	Full project proposal
KPI	Key Performance Indicator
MBE	Model-Based Engineering
MBPLE	Model-Based Product Line Engineering
MBRE	Model-based Requirements Engineering
MDE	Model-Driven Engineering
ML	Machine Learning
SaaS	Software as a Service
SE	Systems and Software Engineering
WP	Work Package

1 Introduction

Deliverable 5.5 (D5.5) is the first output of Task 5.3 in WP5. Task 5.3 in general deals with the execution of AIDOaRt technologies in the context of industrial case studies, and evaluating and measuring the achievements of AIDOaRt in solving the challenges of the use cases. This deliverable, in particular, focuses on: establishing and identifying the baseline of the use cases, description of testbeds and demonstrators on which evaluation of AIDOaRt technologies will be done, description of evaluation and validation methodologies, and most importantly defining the first set of KPIs and metrics to measure the improvements with respect to use-case challenges thanks to the application of AIDOaRt technologies and solutions. The subsequent deliverables from Task 5.3, namely D5.7 and D5.9, will respectively present the results of the evaluation of the AIDOaRt technologies after the first and final use-case development phases.

To facilitate the definition of KPIs and metrics for case studies and industrial use cases, we adopted a top-down approach starting from the project-level KPIs that are defined in the FPP. In other words, for each case study, first a set of relevant project-level KPIs are identified and selected, and then they are refined and described according to the context of that case study. This results in a set of case study KPIs that serve as metrics to measure the expected improvements of AIDOaRt technologies in the context of a case study. As part of this process, a mapping and traceability link between case study KPIs and the individual use-case scenarios of a case study is also established. Therefore, for each case study, a summary table listing its use-case scenarios is also included in this deliverable.

The traceability information is also encoded in the format that is chosen to set the IDs of case study level KPIs based on the following template:

<Use-case scenario ID>_<Relevant project-level KPI>_<Instance number>

or

<Partner ID>_GEN_<Relevant project-level KPI>_<Instance number>

This formatting style enables to easily and quickly identify from a case study KPI ID its related use-case scenario, and the project level KPI that it defines and instantiates while giving the flexibility to reuse and redefine a single project-level KPI multiple times for a specific use-case scenario to capture different aspects of that KPI based on multiple definitions in the context of that scenario. The latter is possible by having an instance number as the last part of a case study KPI ID. Moreover, in few cases where a partner may need to define a KPI that is related to its case study in general, and not specific to a particular use-case scenario, the second format and style that is shown above can be used where the term ‘GEN’ is used instead of a use-case scenario ID.

1.1 Relations to other deliverables

This deliverable mainly builds upon and extends the work that was done in deliverable D1.1 and D1.3 regarding the details of project case studies. In particular, while those deliverables from WP1 contain

information on requirements, detailed use-case specifications, and definitions of use-case scenarios, D5.5 focuses on details related to the evaluation of case studies and use-case scenarios. The information captured in D5.5 will then be of interest for different activities and deliverables in the project related to data collection, and solution deployment and integration. Moreover, as mentioned before, the content of D5.5 will serve as the main input for D5.7 and D5.9 as well where evaluation results of AIDOaRt technologies applied on industrial case studies will be reported.

1.2 Document Structure

In the rest of the deliverable, we first present a recap and overview of project-level objectives and KPIs. These are basically taken from the full project proposal (FPP) and are provided here to help with understanding the mapping between case study level KPIs and the project-level ones. In the next sections, we allocate a dedicated section for each case study in the project where we capture:

1) a brief overview of each case study,
2) description of the current way of working and baseline technology of each case study, basically establishing the situation of each case study at the start of the project,
3) capturing the expected improvements in AIDOaRt with respect to the identified challenges of each case study,
4) defining a set of KPIs relevant for the context of each case study which will be used as a way to measure the level of improvements thanks to the application and adoption of AIDOaRt technologies. Case study level KPIs are instantiated from the project-level KPIs and refined based on the context of each case study,
5) the overall evaluation and validation methodology that each case study will use to collect values for measurement of KPIs and the process to determine the level of success of AIDOaRt technologies in providing the expected improvements,
6) description of the target system that is used as demonstrator and testbed, including e.g., any specific hardware platforms, software subsystems, simulation environment, and so on that constitute the actual demonstrator for a case study in the project against which AIDOaRt solutions will be evaluated, and measurements are performed.

2 AIDOaRt Project-Level Objectives and KPIs

In this section, we present the overall objectives of the AIDOaRt project, and the list of project-level KPIs the are defined to measure the attainment of these objectives.

Objective ID	Description
O1	Holistic integration of system data. To support the integration of different heterogeneous data sources (covering both runtime and design time data), including data pre-processing, aggregation and further access via a global data model.
O2	AIDOaRt global model-based framework for the continuous development of CPS. To develop a scalable framework for efficient handling and management of numerous, heterogeneous and large models potentially covering several functional and non-functional perspectives of the system under development (in a DevOps process) suitable for large distributed cross-functional working teams and allowing to integrate feedback (notably from runtime data) to the system level models.
O3	AIDOaRt AI-augmented toolkit. To employ AI (and notably Machine Learning) techniques in multiple aspects of the system development process (e.g., to support requirements, monitoring, modeling, coding, and testing activities) via a combination of AIOps and MDE and by extending the AIDOaRt core model-based framework.
O4	AIDOaRt demonstrators validation - to develop specific demonstrators showing the application of the AIDOaRt model-based framework and AI-augmented toolkit, and validating these AIDOaRt technologies, through 8 complementary industrial case studies.
O5	AIDOaRt market uptake - to prepare exploitation of the AIDOaRt technology through open source and commercial tools.
O6	Strengthening European excellence in Continuous System Engineering and AIOps. The AIDOaRt project incorporates an innovative research agenda and work plan that will contribute to maintain and further advance the AI-augmented and DevOps world-class research that already exists in Europe.

Table 1 AIDOaRt Objectives

In the following table we present the project-level KPIs.

KPI ID	Definition	Target Value
KPI_1.1	Improvement of the time required for identification of design problems thanks to the analysis of the collected data.	20%
KPI_1.2	Improvement of the early detection of system deviations	20%
KPI_2.1	Reduction of the time/effort required for managing and handling all the involved DevOps models	30%
KPI_2.2	Increase in the number of available data sources to be actually managed and handled in existing engineering practices.	20%
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).	20%
KPI_3.2	Increase the coverage and quality of actionable feedback for the next DevOps iteration.	20%
KPI_4.1	Increase in the percentage of parts of the DevOps process covered in the Use Cases with productivity improvement.	20%
KPI_4.2	Reduction of deviations from the specifications to improve predictability, conformance to specifications and proposal of system design refinements.	20%
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 organized public workshops, hackathons and dissemination events to raise awareness on the opportunities of AIOps for European companies.	3

Table 2 AIDOaRt Project-level KPIs

The above presented KPIs are used to measure the attainment of the AIDOaRt project objectives as shown in the below diagram.

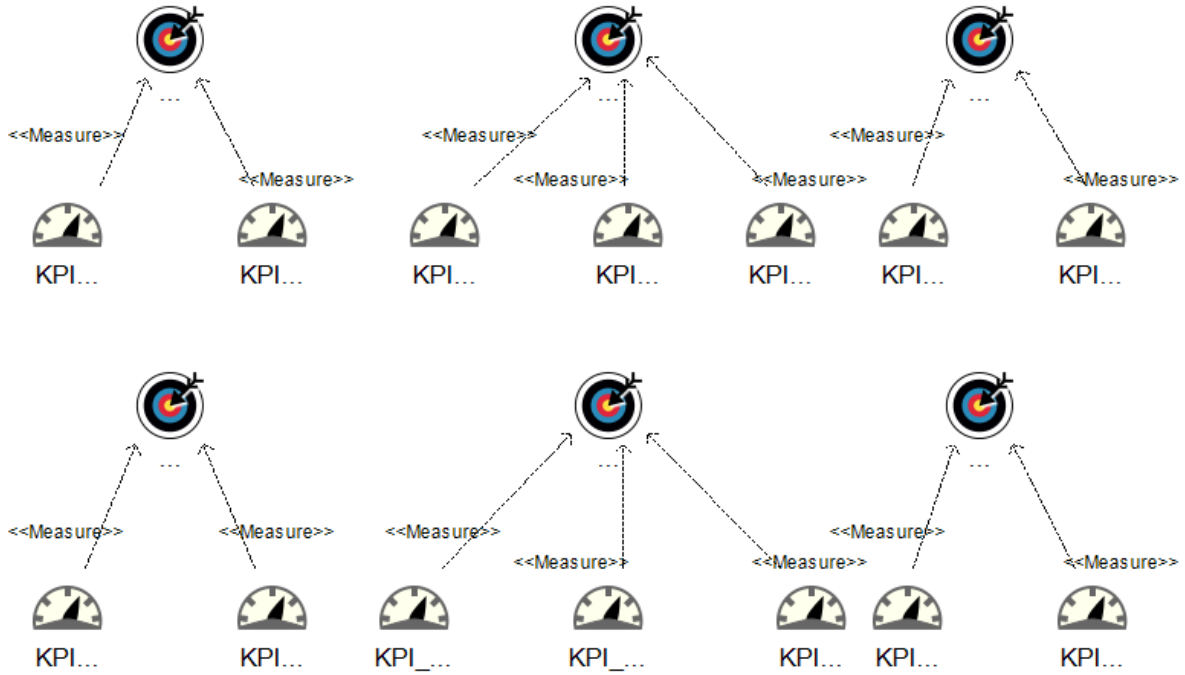


Figure 1 AIDOaRt Objectives Attainment Measurement with Project-level KPIs

3 Case Studies Overview, Evaluation Criteria and Validation Process

In this section, we present an overview on the AIDOaRt partners' case studies, their current way of working and baseline technologies, as well as their expected improvements in AIDOaRt. For each of their case studies, case study providers define their relevant KPIs and their success criteria in terms of target values for each KPI. In addition, they provide their validation methodology and process, demonstrator setup and testbed.

3.1 Case Study Abinsula - Safety critical systems in the automotive domain using disruption technology

3.1.1 Case study overview

Modern cars are connected systems and acquire inputs from the environment, thus they can be considered as Cyber Physical Systems. In this Case Study the sensors are going to be the video that can be used instead of the rear-view mirror. With this new source of information, new challenges in the development process are arising. This is especially true where several stakeholders, such as hardware specialists, software developers and system designers have to work together with safety engineers to ensure a reliable and safe system. In this context, the combination of new and disruptive technology like Artificial Intelligence (AI) and Machine Learning (ML) can enhance the entire development of safety-critical systems and support the prediction of new scenarios that might be considered as safety critical.

The main goal of the Abinsula Case Study is to study and propose an approach, based on AIDOaRt methods, and apply it in an interesting case as the virtual mirrors in cars. In particular, the main technological goals of the Case Study are related to the introduction of AI/ML techniques in the modeling and testing phase of the system development life cycle. Therefore, the Case Study is divided in two main parts: (1) a methodological part and (2) a PoC.

Methodology

Figure 2 illustrates the current overview of the methodology, which is the result of ongoing discussions and collaboration with the University of Sassari (UNISS). The methodology consists of three main parts that correspond to Abinsula use case scenario: (1) Functionality verification, (2) Test definition, (3) Compliance verification.

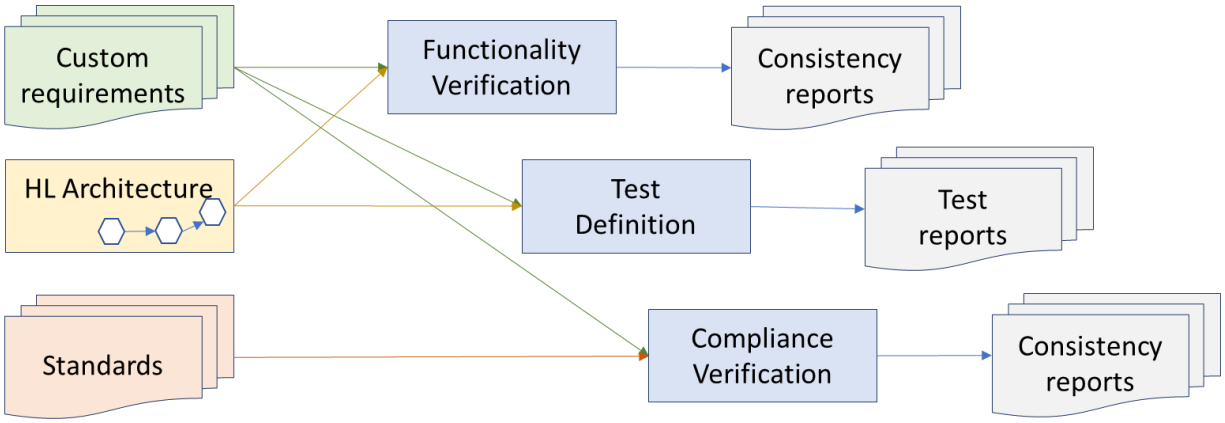


Figure 2 - Current Overview of the Case Study Methodology

The Functionality Verification refers to the automatic verification of requirement and model consistency against properties. The Test Definition refers to the automatic generation of a suite of tests for the system. The Compliance Verification refers to the automatic verification of requirement consistency with respect to a reference guideline. For a more detailed description please refer to deliverable D1.3 in Section 3.1.1 Current System Architecture.

Virtual Rear Mirror PoC

The ABI case study is developed from scratch in AIDOaRt, therefore there is no previous implementation of the PoC. Figure 3 illustrates the current envisioned PoC overview, result of internal decisions and of discussions and collaboration with INTECS.

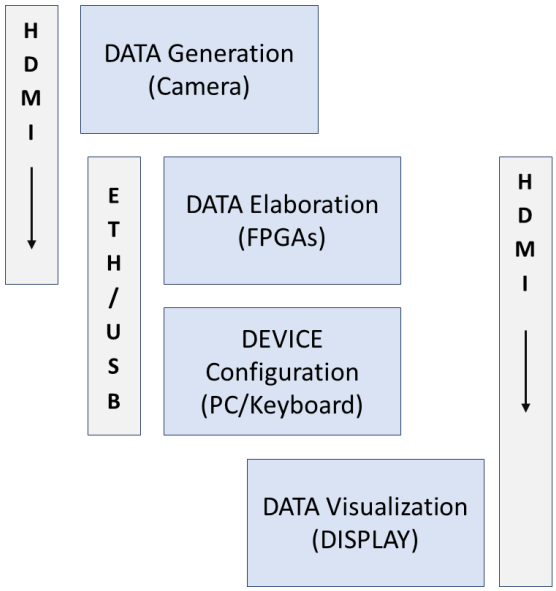


Figure 3 - Virtual Rear Mirror PoC overview

The main data involved in this PoC are images, therefore the main data interface envisioned is the HDMI port. Images from the cameras (DATA Generation Module) pass through two main steps of

elaboration (DATA Elaboration Module) and visualization on the dashboard display (DATA Visualization Module). It is envisioned for the user the possibility of configuring options, therefore a Device Configuration Module is planned to configure the DATA Elaboration Module (the configuration interface is currently under discussion).

This figure does not include details that have not been discussed enough yet (as the possibility of triggering any direct response in the car) and will be modified and updated according to further future discussions with solution providers and implementation choices.

3.1.2 Current way of working and baseline technology

Automotive is adding more and more features that benefit from fast edge computing near the sensors, especially when offering driving assistance. However, a complete replacement of rear mirrors with cameras would require a complete change of car design. Furthermore, having all the rear images from cameras opens a plethora of possibilities (data fusion, correlation of different images from different points of view), and with them also possible hazards in the car, not dangerous as a failure of the brakes but still potentially very relevant issues. Therefore, nowadays such systems are implemented only in concept cars and small production cars that do not apply the same regulation applied in the commercial production.

In this context, it is important to highlight that if it is true that AI is now a recognized as innovative technology, it is far from being applied in real safety-critical applications due to the lack of methodologies, for example for the predictability of the system in domains such as the automotive one.

3.1.3 Expected improvements in AIDOaRt

In the automotive domain, safety represents a critical objective and the emergence of standards, such as ISO 26262 and ISO 16505, has helped the automotive industry to focus on practice to address safety in a systematic and consistent way. In this context, the use of AI and ML is on the rise in order to enhance the automated verification of systems applied in real safety critical applications, such as the ones considered in this Case Study.

The main expected improvements are related to the introduction of artificial intelligence techniques in the modeling and testing phase of the system development life-cycle, as detailed in the following:

- In the modeling phase, the use of automated reasoning and ML techniques for the automated verification of specifications and of the high-level model in order to detect inconsistencies.
- In the modeling/implementation phase, the use of AI techniques for the verification of Deep Neural Network models/implementations.
- In the testing phase, the use of automated reasoning and ML techniques for the generation of optimal test suites.

From the PoC perspective, the main improvements are related to:

- the use of AIDOaRt enhanced explainable object detection modules that enable humanly interpretable reasons for the model decision: its output is enriched by an important hint in what aspects are important to recognize an object in the monitored scene. This gives a measure of the reliability of the model, an aspect of fundamental importance to validate models developed in safety critical contexts, that goes beyond classic performance metrics.

The application of the just mentioned techniques to the Abinsula Case Study does not expect to be the solution to the situation presented in the previous section. However, they can represent a first step in the application of automated verification procedures and tools to this use case in order to increase the correctness, checked at model-level, of the system. The ultimate goal is to provide methods that could inspire how to certificate these systems as safety critical systems.

3.1.4 Relevant KPIs and definition of success

The high-level technological goals of Abinsula Case Study in the AIDOaRt project are mainly related to the introduction of AI/ML techniques in the modeling and testing phase of the system development (ABI_Goal1, ABI_Goal2) and to the investigation of solutions for driving assistance (ABI_Goal3). In the following we report the detail of these goals:

- [ABI_Goal1] Improve and automate the modeling phase of the systems through the use of AI/ML for the verification of specifications and for the verification of the high-level models.
- [ABI_Goal2] Improve the testing phase by using automated reasoning techniques for the generation of optimal test suites.
- [ABI_Goal3] Investigate new solutions for intelligent driving assistance based on ML video processing and deep learning.

The following rationale has been applied in determining the metrics to gauge the success of the Case Study:

- The specifications and properties' model verification will be improved using AI/ML techniques that will automate the process.
- Through the automated verification productivity will improve by reducing the effort of performing the activity manually. Moreover, it will avoid possible human errors.
- Solutions, based on deep learning architectures, for object detection and tracking based and depth perception. These functionalities can provide indications of potential hazardous situations, thus increasing the safety of the vehicles on which they are installed.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
ABI_UCS1	Functionality Verification
ABI_UCS2	Test definition
ABI_UCS3	Compliance verification

Table 3 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
ABI_GEN_4.1_1	Independent	KPI_4.1	Automation of processes improve efficiency in performing the specification and consistency verification.	0%	20%

Table 4 Case Study KPIs

3.1.5 Validation methodology and process

ABI_UCS1 and ABI_UCS3 will be evaluated by comparing the results of the automated processes with results of an expert engineer.

ABI_UCS2 will be evaluated comparing the set of generated tests with tests that would be written by an expert engineer.

The approach to collect KPI values and measure performance will be as follows:

ABI_UCS1

- Assessment that automated consistency verification of specifications and system model report matches the manual consistency report written by an expert engineer.
- Computation of the share of automation versus the total number of process steps.
- Comparison of the time for the automated process to perform the automated consistency verification with the time for an expert engineer to perform it manually.

ABI_UCS2

- Verification that the set of generated tests is complete enough to include the tests that an expert engineer would envision.
- Computation of the share of automation versus the total number of process steps.
- Comparison of the time for the automated process to generate the set of tests with the time for an expert engineer to perform it manually.

- Verification that the automated requirements compliance report matches the manual compliance report written by an expert engineer.
- Computation of the share of automation versus the total number of process steps.
- Comparison of the time for the automated process to perform the compliance verification with the time for an expert engineer to perform it manually.

3.1.6 Demonstrator setup and testbed

Abinsula will evaluate the AIDOaRt solutions for its use case in the following testbed setups:

ABI_UCS1 - Automatic verification of the formal consistency of the technical specifications and of the system design expressed in a given Domain System Language (DSL) by adopting the following UNISS tools UNISS_SOL_1, UNISS_SOL_4, UNISS_SOL_5. The generated documentation is evaluated by an expert engineer.

ABI_UCS2 - The automatic generation of a test suite by adopting the UNISS_SOL_3 UNISS tool. The generated suite of tests is evaluated by an expert engineer.

ABI_UCS3 - Automatic verification of the formal consistency of the technical specifications with respect to a guideline by adopting the following UNISS tools UNISS_SOL_1, UNISS_SOL_4, UNISS_SOL_5. The generated documentation is evaluated by an expert engineer.

Results of the collaborations with solutions providers are going to be exploited in the implementation of the PoC. With reference to the Virtual Rear Mirror overview presented at the beginning of this chapter, the PoC basic elements are implemented as follows:

- Generation module is a camera that captures 1920x1080 images (HD 1080P). Data related to the camera include a up to 60 FPS stream and the camera ID.
- The DATA Elaboration module is the platform in which INTECS (INT_OD solution) algorithms are implemented. Currently, the adoption of an FPGA platform is considered, and feasibility study is being done on a Xilinx Pynq-Z1 FPGA, but other FPGAs are under evaluation. Configuration might be necessary, and currently it is planned to do it from a PC (DEVICE Configuration).
- Finally, images must be individually processed and collectively reconciled and visualized on the dashboard. The DATA Visualization module has not been defined yet. However, we can say that the size of the output

3.2 Case Study - AVL_TCV

3.2.1 Case study overview

The AVL SCENIUS Test Case Generator is an easy means to generate 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.

3.2.2 Current way of working and baseline technology

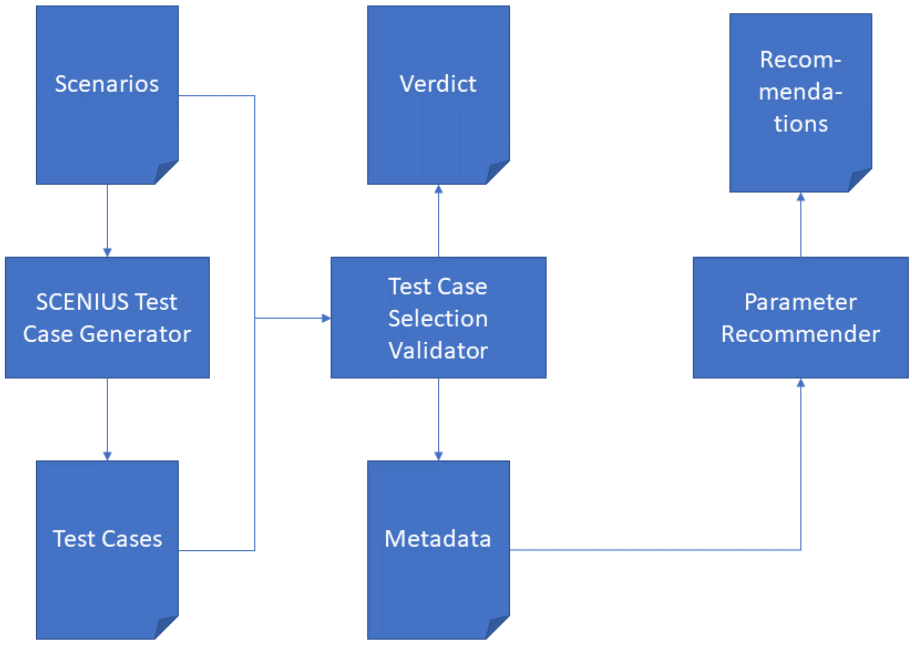


Figure 4 Current System Architecture

The input data for the SCENIUS Test Case Generator describe traffic situations in an abstract way with a set of open parameters (e.g. ego speed, acceleration, distance to other car, etc.). Each parameter has a certain range of possible values. Thus, each Scenario can translate into thousands of Test Cases but not all of them are necessary to cover critical situations. The execution of Test Cases on HIL systems or real vehicles is very expensive. Therefore, a proper selection of parameters is necessary.

3.2.3 Expected improvements in AIDOaRt

The SCENIUS Test Case Selection Validator must determine with a given accuracy, if a given test case selection is adequate. Furthermore, the verdict given by the SCENIUS Test Case Selection Validator must be understandable/explainable by humans. Finally, 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.

3.2.4 Relevant KPIs and definition of success

Definition of success for the 1st case story "CS1" (i.e., quantification of experiment information gain) and 3rd case story "CS3" (i.e., quantification of experiment result maturity): The computed information gain and maturity shall not contradict an expert's assessment and shall show the same trends as the expert's assessment. Experiments with significantly higher/lower information gain and maturity (relative to other experiments directly before or after in the development project) shall be detected.

Definition of success for the 2nd case story "CS2" (i.e., prediction of parameter and product KPI evolution): The forecast for future parameter and KPI values shall be as close as possible to the actual values as they evolve along the development project. Particularly, the forecast shall capture the trend of parameters and KPIs, e.g., if a KPI will get closer to or move away from its target value.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_TCV_UCS1	SCENIUS Test Case Selection Validator
AVL_TCV_UCS2	SCENIUS parameter recommender

Table 5 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_TCV_UCS1_KPI_3.1_1	AVL_TCV_UCS1	KPI_3.1	Automate the processes of examining and providing the explainability of the no-critical and critical test cases.	0%	20%
AVL_TCV_UCS2_KPI_3.1_1	AVL_TCV_UCS2	KPI_3.1	Automate the processes of evaluating whether the selected test	0%	30%

	cases are critical or no-critical.

Table 6 Case Study KPIs

3.2.5 Validation methodology and process

The basic validation methodology for all 3 case stories is to apply the AIDOaRt results (i.e., models) to test data and check the performance with this test data. The test data will be provided by AVL. It will contain one or more datasets. Each dataset represents a whole development project, i.e., a time series of experiments with associated KPI results, parameter values and maturities.

- Validation methodology for the 1st case story "CS1" and 3rd case story "CS3": An AVL domain expert reviews the test data and assesses the information gain and maturities of experiments along the time series of experiments. The expert assesses whether information gain and maturity increase or decrease relative to prior experiments. The expert identifies experiments with significantly lower/higher information gain or maturity, i.e., outliers. This assessment is compared to the output of the model. The comparison is done qualitatively by an AVL expert.
- Validation methodology for the 2nd case story "CS2": Each time series of experiments in the test data is partitioned into 2 segments: Past and future. The model may use the experiments of the "past" segment as input and must forecast the parameters and KPIs of experiments in the "future" segment. The deviation between forecast and values in the "future" segment is measured, e.g., by using a simple sum of squared errors approach. The trend in the forecast and the test data is quantified, e.g., by computing the numerical first order derivative with respect to time for all parameters and KPIs.

3.2.6 Demonstrator setup and testbed

The minimum demonstrator setup basically consists of a repository for the test data, an execution environment for the models and a GUI to visualize and assess the test data and model outputs.

3.3 Case Study - AVL_SEC

3.3.1 Case study overview

One of AVL core business is the development of test beds for vehicle development. Usually these test beds evaluate the functional behaviour of a car like exhaust emission or consumption optimization. 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,

CS1: Design phase;

CS2: In-use phase;

CS3: Remotely via cloud-based service.

3.3.2 Current way of working and baseline technology

The current main methodology of AVL's automated cybersecurity approach is to transfer attack vectors from one system to another. To do so, we abstract a concrete test case and turn it into a generic test scenario by stripping it of all SUT-specific information. Single executable steps of a test case (test scripts) become generic test patterns (see Figure 1).

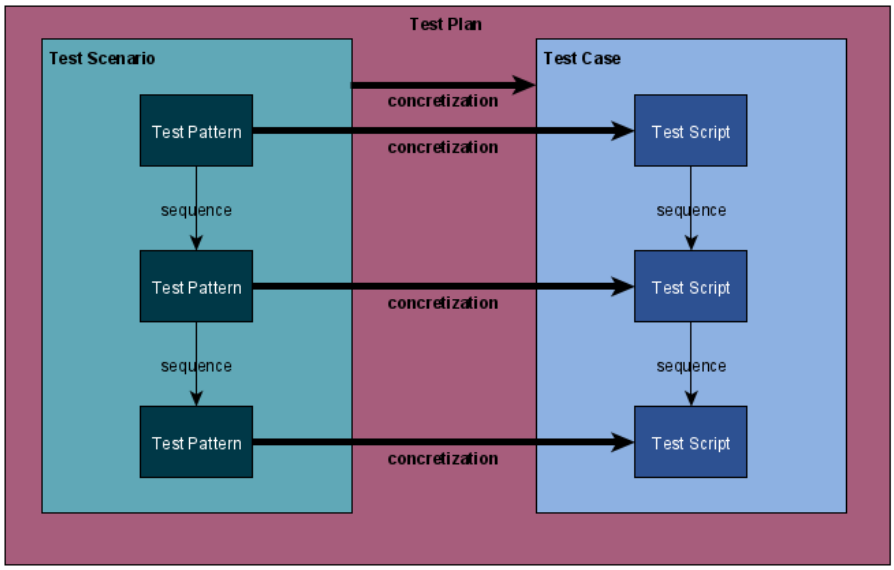


Figure 5 Test Abstraction

For modeling and storing these generic attacks, we developed a domain-specific language (DSL), called Agnostic Language for Implementing Attack (ALIA).

For (a simplified) example, an attack that captures an infotainment head unit and issues a fake speed signal onto a connected CAN bus would not contain any specifics of the SUT, rather a CAN message for the speed signal would generically called CAN_SPD, while a test case generation fuses the script with information about the SUT (in this exemplary case, the concrete CAN message, e.g. 5A1#11.2233.44556677.88). Listing 1 shows an example DSL attack script. The outcome is a semi-executable JSON script that will be interpreted and executed by a dedicated execution engine (see below).

```
PreConditions:
  BT-Scanning: BT_IF
  BT-Exploiting: target
Actions:
  BT-Scanning: target = scan(type:BlueBorne, interface BT_IF)
```

```
  BT-Exploiting: shell = exploit(type: Blueborne, target:target)
  Install Script: attackScript = exploit(type:InstallAndroidCANDosScript, target:target)
  can_attack: exploit(type:ScriptExecution, target:target, shell:shell, file:attackScript)
PostConditions:
  BT-Exploiting: shell
  can_attack: CAN_MESSAGE(CAN_SPD)
```

Listing 1: DSL Attack Script Example

The Test Execution is runs on an Attack Execution Engine (AXE), a platform independent python application.

Input requests for the application contain an array of executable commands, which each consist of the tool to use, its parameters, the environment and a time duration that specifies how long the output collection phase takes. The parameter list for each command may include placeholders that are either determined by the application at runtime or are loaded from the global configuration of the application before execution. The Test Case Generator (TCG) uses scripts that are defined in the attack DSL as a blueprint and outputs corresponding JSON objects that can be directly used for execution and consist of the respective Precondition and Action block, whereas the Postcondition block is forwarded to a Test Oracle.

Each step in the Action block is executed subsequently. Before execution, the application checks if all corresponding preconditions are matched. Depending on the necessary execution environment, commands can be executed in different shells than the initial bash shell as well, for example if an exploit returns a reverse shell, it is stored onto an object and new commands can be piped into that shell as an input. After the execution, the output of each command is collected and stored into the HTTPS response of the application.

Verification of Postconditions is done by the Test Oracle, which is implemented as a rule-based engine that runs on an existing automotive test control solution. The Oracle receives the condition block from the TCG and monitors the SUT and the tool output received from the AXE accordingly. If a condition is met, it reports this back to the Orchestration Software. Through the rules, it asserts whether the SUT has failed or passed a specific test of the complete test case and reports this result to the orchestration software and GUI.

3.3.3 Expected improvements in AIDOaRt

The shortcoming of the current SOTA of AVL's automated testing solution is that one needs a lot of information a priori. Mainly a large database of prefined attacks, as well as a sufficiently large database for System-under-Test (SUT) information to concretize the abstracted attack patterns.

The main improvement of AIDOaRt is to enhance the concept with a) learning algorithms that are capable of learning behavioural models of automotive systems and b) algorithms for anomaly detection to create properly functioning test oracles as well as to develop morphological test generation algorithms. With a) we are able to derive models that can be transferred into a checkable form and generate test cases out of traces that lead to model specification violation, that in the end, might prove to be security-relevant. As modern car architectures include a security gateway, we also need a means to test the shortcomings of these. To do so, in AIDOaRt we apply learning and fuzzing

methods to probe for messages that might go through the gateway and introduce anomalous behavior on the protected side of the gateway.

3.3.4 Relevant KPIs and definition of success

The definitions of the KPIs for case study AVL_Sec are reported case study KPIs table below. As there is no previously working solution for the addressed problem, the baseline is zero for each KPI. Learned protocol implementations, found counterexamples and fuzzed anomaly-inducing inputs are absolute values, while the coverages indicate the percental ratio of found versus overall anomalies in a controlled experiment (deliberately injected anomalies).

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_SEC_UCS1	Learning-based model checking for security testing
AVL_SEC_UCS2	Machine-learning based fuzzing & anomaly detection for automotive systems security testing
AVL_SEC_UCS3	Remote Machine-learning intrusion detection

Table 7 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_SEC_USC1_KPI_1.1_1	AVL_SEC_UCS1	KPI_1.1	Learned protocol implementation models	0%	3%
AVL_SEC_USC1_KPI_1.2_1	AVL_SEC_UCS1	KPI_1.2	Found Counterexamples	0%	9%
AVL_SEC_USC2_KPI_3.2_1	AVL_SEC_UCS2	KPI_3.2	Anomaly Coverage	0%	50%
AVL_SEC_USC2_KPI_4.2_1	AVL_SEC_UCS2	KPI_4.2	Fuzzed anomaly-inducing inputs	0%	3%
AVL_SEC_USC3_KPI_3.1_1	AVL_SEC_UCS3	KPI_3.1	Cloud-based anomaly Coverage	0%	50%

Table 8 Case Study KPIs

3.3.5 Validation methodology and process

The validation of the KPIs regarding modeling, counterexamples and fuzzing (1.1, 1.2 and 2.2) will occur at the system itself, i.e. the system application results will provide all necessary validation evidence. The coverage-based KPIs (2.1 and 3.1) will be evaluated in a controlled experiment, that is using crafted system-under-test situations and/or datasets that provide a clearly defined number of anomalies that should be detected, where the actual detection ration poses the measurement for the goal attainment.

3.3.6 Demonstrator setup and testbed

Each of the three use case scenarios will have a dedicated setup for validation and demonstration. UCS1: Test Case Generation using Automata and Machine Learning with Model Checking will consist of a learner framework with a model checker (the AIDOaRT components to be evaluated) that is to provide sensible test cases, which will be subsequently used in the AVL testing framework as a demonstrator (see Figure 1).

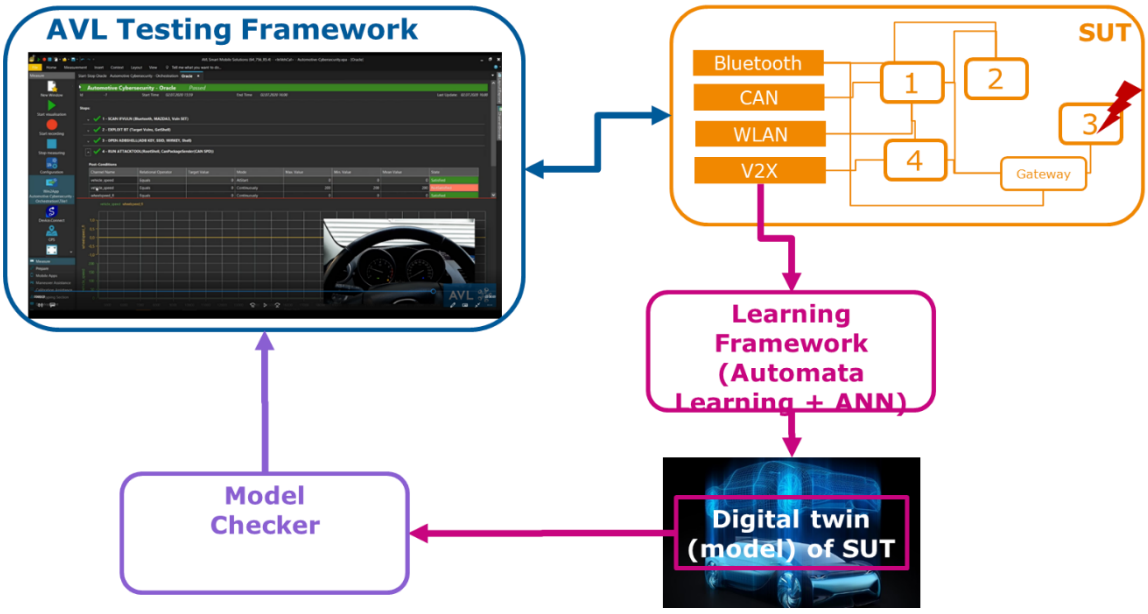


Figure 6 AVL_SEC_UCS1 Evaluation Setup

UCS2: Machine-learning based fuzzing & anomaly detection for automotive systems will consist of an anomaly detection component and a Fuzzer (the AIDOaRT components to be evaluated) that are connected to the system-under-test via the AVL Testing Framwork that will be used to test the efficiency of the anomaly detection under controlled conditions (see Figure 2).

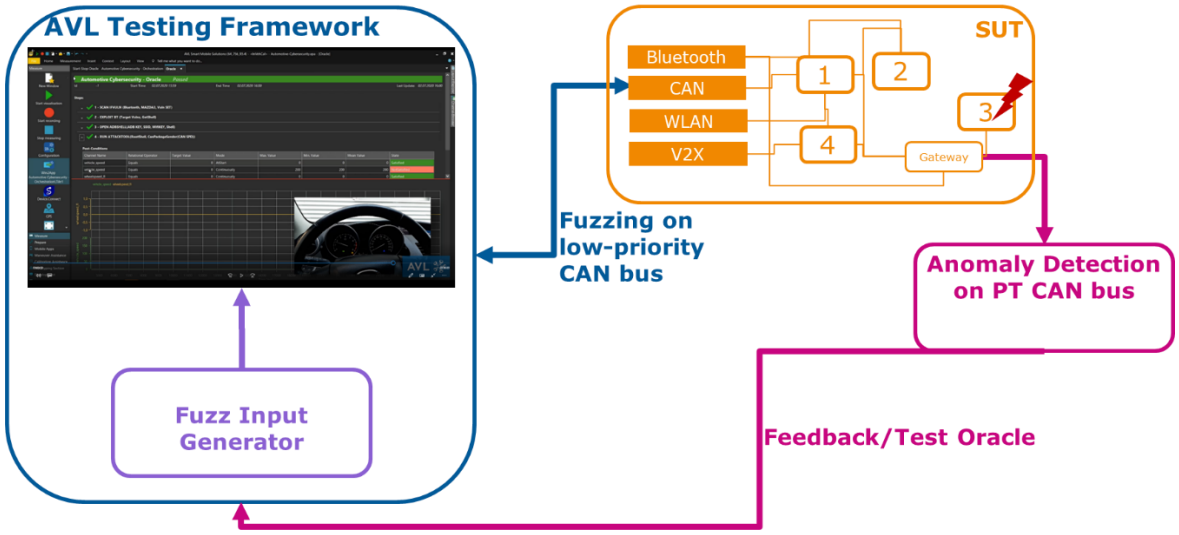


Figure 7 AVL_SEC_UCS2 Evaluation Setup

UCS3: Remote Intrusion Detection using Anomaly Detection will consist of AVL's Device.Connect system that will connect the system-under-test (e.g. a car) to a cloud anomaly detection service (the AIDOaRT component to be evaluated) to demonstrate the detection of anomalies in a vehicle's CAN bus via a remote connection (see Figure 3).

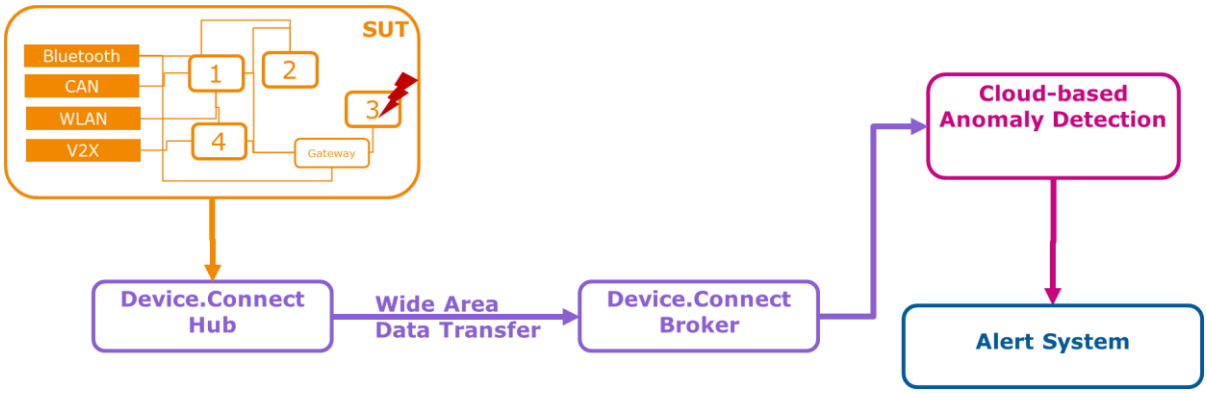


Figure 8 AVL_SEC_UCS3 Evaluation Setup

3.4 Case Study - AVL_TCR

3.4.1 Case study overview

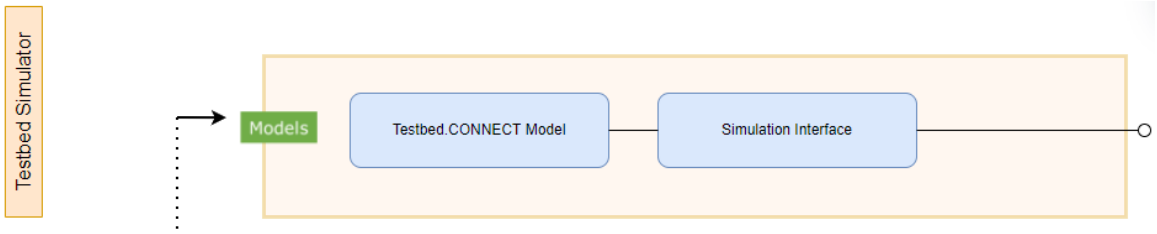
AVL test automation systems are very complex. A reduction of complexity would increase the attractiveness of AVL products for customers.

(1) Detection of an error-prone state of the unit under the test (UUT) during monitoring of the measurements. When an error-prone state is identified, the operator should be informed to secure the UUT, and stop unnecessary testing. Thus, early detection and stopping of error-prone measurements by automated plausibility checks,

(2) The prediction of values that are not available on the test bed or only available with great effort should reduce the complexity of the testing automation system.

(3) To save money and time required for setting up and performing testbed measurements, the informativeness of the measurement experiments can first be assessed on a test simulator. For modeling testbed and under the test (UUT), the PUMA simulator available in AVL is used in the preparation of measurements. Simulations of the experiments performed before going to the test bed should reduce the complexity of the testing automation system.

3.4.2 Current way of working and baseline technology



OR

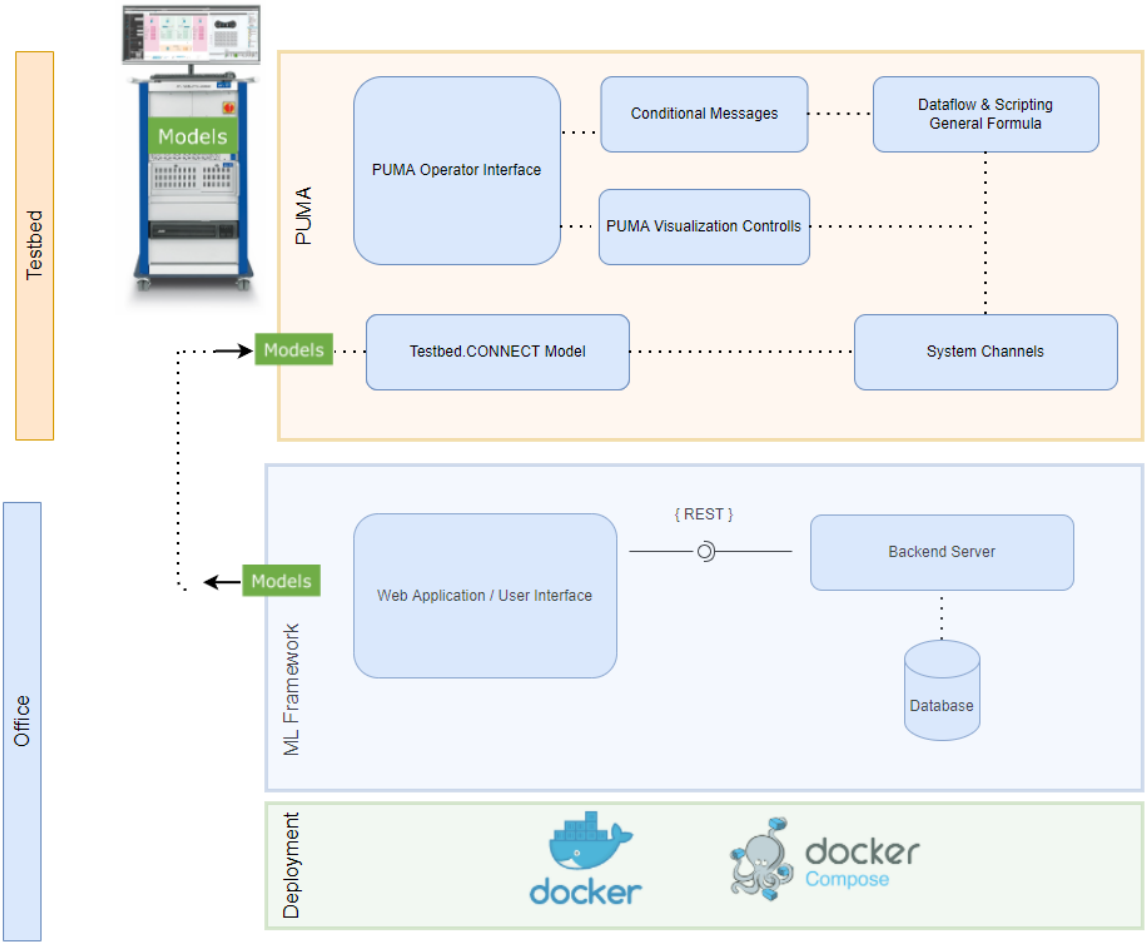


Figure 9 Current System Architecture

The ML Framework architecture is presented in Figure 1. It consists of independent modules, implemented in various technologies, that communicate by invoking REST-HTTP services and with connectors to a tracking server database. The ML Framework follows a client-server architecture and can be installed and used in the office without a dependency to a testbed. For deployment purpose, docker is used. On the testbed standard visualization controls and functional blocks are used from the AVL PUMA ecosystem.

3.4.3 Expected improvements in AIDOaRt

In the AIDOaRt project, we aim to develop a data driven models based on AVL provided testbed data to do online diagnostics and anomaly detection. The generated model will be used when running future tests as a reference model for identifying error prone behaviors. Next, we aim to develop data driven models based on AVL provided testbed data to do simulation. Further, we aim to finding new AI/ML technologies to increase and extend the model efficiency and capabilities of the current modeling solutions. Finally, we aim to deploy any data driven model generated with state-of-the-art frameworks to platform independent C++ code.

3.4.4 Relevant KPIs and definition of success

Definition of success for the 1st case story "CS1" (i.e., quantification of experiment information gain) and 3rd case story "CS3" (i.e., quantification of experiment result maturity): The computed information gain and maturity shall not contradict an expert's assessment and shall show the same trends as the expert's assessment. Experiments with significantly higher/lower information gain and maturity (relative to other experiments directly before or after in the development project) shall be detected.

Definition of success for the 2nd case story "CS2" (i.e., prediction of parameter and product KPI evolution): The forecast for future parameter and KPI values shall be as close as possible to the actual values as they evolve along the development project. Particularly, the forecast shall capture the trend of parameters and KPIs, e.g., if a KPI will get closer to or move away from its target value.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_TCR_UCS1	AI-augmented plausibility analysis of UUTs
AVL_TCR_UCS2	Generation of a data driven model of a equipment hard to measure on the testbed or not freely available.
AVL_TCR_UCS3	Machine Learning based Virtual Sensors

Table 9 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_TCR_UCS1_KPI_1.2_1	AVL_TCR_UCS1	KPI_2.1	Improvement in the early detection of the unit under the test (UUT) deviations.	0%	20%
AVL_TCR_UCS2_KPI_3.2_1	AVL_TCR_UCS2	KPI_3.2	Increase the coverage and quality of the testbed Simulator.	0%	20%
AVL_TCR_UCS3_KPI_3.2_1	AVL_TCR_UCS3	KPI_3.2	Increase the coverage and quality of the virtual sensor.	0%	30%

Table 10 Case Study KPIs

3.4.5 Validation methodology and process

The basic validation methodology for all 3 case stories is to apply the AIDOaRt results (i.e., models) to test data and check the performance with this test data. The test data will be provided by AVL. It will contain one or more datasets. Each dataset represents a whole development project, i.e., a time series of experiments with associated KPI results, parameter values and maturities.

- Validation methodology for the 1st case story "CS1" and 3rd case story "CS3": An AVL domain expert reviews the test data and assesses the information gain and maturities of experiments along the time series of experiments. The expert assesses whether information gain and maturity increase or decrease relative to prior experiments. The expert identifies experiments with significantly lower/higher information gain or maturity, i.e., outliers. This assessment is compared to the output of the model. The comparison is done qualitatively by an AVL expert.
- Validation methodology for the 2nd case story "CS2": Each time series of experiments in the test data is partitioned into 2 segments: Past and future. The model may use the experiments of the "past" segment as input and must forecast the parameters and KPIs of experiments in the "future" segment. The deviation between forecast and values in the "future" segment is measured, e.g., by using a simple sum of squared errors approach. The trend in the forecast and the test data is quantified, e.g., by computing the numerical first order derivative with respect to time for all parameters and KPIs.

3.4.6 Demonstrator setup and testbed

The minimum demonstrator setup basically consists of a repository for the test data, an execution environment for the models and a GUI to visualize and assess the test data and model outputs.

3.5 Case Study - AVL_RDE

3.5.1 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 behavior 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 behavior that is hard to model with a simple algorithm.

Each driver has its own specific behavior that influences the RDX KPI, electrical range, mechanical, electrical and thermal durability thermal behavior, battery aging, 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 behavior 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 behavior.

3.5.2 Current way of working and baseline technology

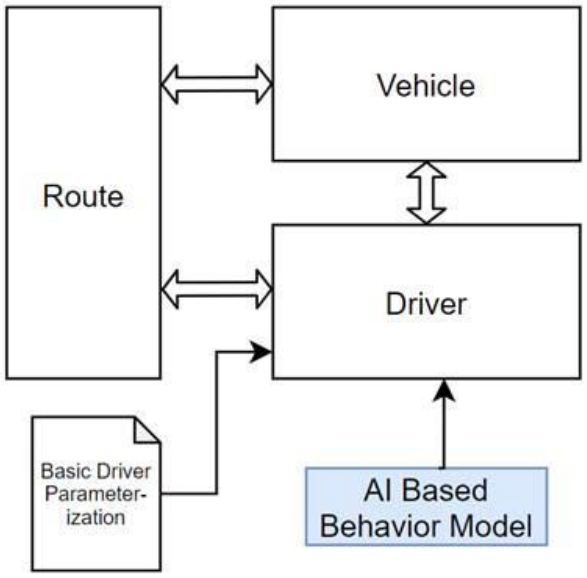


Figure 10 Current System Architecture

AVL Route Studio Simulator is a drive cycle simulation library that is used mainly for RDX based cycle generation. It is composed of route, driver and vehicle models. It runs in fixed time steps and mostly runs below 10Hz. The driver model performs arbitrary look ahead for foreseeing the next *N* steps in the route for slowdowns due to speed limit signs, curvature or stops. The driver contains a humanization model that is a formulated abstraction of typical driver behavior that is significant for RDX KPI.

3.5.3 Expected improvements in AIDOaRt

In this project we aim to improve this humanization model with an AI based one, in which the dynamic models of driver and vehicle perform the physical layer, while humanization factor is augmented into velocity demand values.

3.5.4 Relevant KPIs and definition of success

Definition of success for the 1st case story "CS1" (i.e., quantification of experiment information gain) and 3rd case story "CS3" (i.e., quantification of experiment result maturity): The computed information gain and maturity shall not contradict an expert's assessment and shall show the same trends as the expert's assessment. Experiments with significantly higher/lower information gain and maturity (relative to other experiments directly before or after in the development project) shall be detected.

Definition of success for the 2nd case story "CS2" (i.e., prediction of parameter and product KPI evolution): The forecast for future parameter and KPI values shall be as close as possible to the actual values as they evolve along the development project. Particularly, the forecast shall capture the trend of parameters and KPIs, e.g., if a KPI will get closer to or move away from its target value.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_RDE_UCS1	Vehicle dynamics isolated speed profile data from real world measurements.
AVL_RDE_UCS2	Basic driver behaviour identification
AVL_RDE_UCS3	Model driver behaviour patterns
AVL_RDE_UCS4	Worst Case Emissions Scenario

Table 11 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_RDE_UCS1_KPI_3.1_1	AVL_RDE_UCS1	KPI_3.1	Automate generation of driver profiles which are currently manual.	0%	30%
AVL_RDE_UCS2_KPI_3.1_1	AVL_RDE_UCS2	KPI_3.1	Automate generation of drivers behavior which are currently manual.	0%	20%
AVL_RDE_UCS3_KPI_3.2_1	AVL_RDE_UCS3	KPI_3.2	Increase the coverage and quality of the generated	0%	20%

			measurements of drivers behavior.		
AVL_RDE_UCS4_KPI_3.2_1	AVL_RDE_UCS4	KPI_3.2	Increase quality and accuracy of the critical emission generated by the engine testbed experiment.	0%	20%

Table 12 Case Study KPIs

3.5.5 Validation methodology and process

The basic validation methodology for all 3 case stories is to apply the AIDOaRt results (i.e., models) to test data and check the performance with this test data. The test data will be provided by AVL. It will contain one or more datasets. Each dataset represents a whole development project, i.e., a time series of experiments with associated KPI results, parameter values and maturities.

- Validation methodology for the 1st case story "CS1" and 3rd case story "CS3": An AVL domain expert reviews the test data and assesses the information gain and maturities of experiments along the time series of experiments. The expert assesses whether information gain and maturity increase or decrease relative to prior experiments. The expert identifies experiments with significantly lower/higher information gain or maturity, i.e., outliers. This assessment is compared to the output of the model. The comparison is done qualitatively by an AVL expert.
- Validation methodology for the 2nd case story "CS2": Each time series of experiments in the test data is partitioned into 2 segments: Past and future. The model may use the experiments of the "past" segment as input and must forecast the parameters and KPIs of experiments in the "future" segment. The deviation between forecast and values in the "future" segment is measured, e.g., by using a simple sum of squared errors approach. The trend in the forecast and the test data is quantified, e.g., by computing the numerical first order derivative with respect to time for all parameters and KPIs.

3.5.6 Demonstrator setup and testbed

The minimum demonstrator setup basically consists of a repository for the test data, an execution environment for the models and a GUI to visualize and assess the test data and model outputs.

3.6 Case Study - AVL_MBT

3.6.1 Case study overview

AVL Cameo provides a model based testing component that aims at reducing the testing effort on the testbed. The idea is to reduce the design space, which consists of a relatively low number of variations (<10) to the region that is relevant for the calibration task. For example, the design space consisting of injection patterns to an engine can be reduced by observing the system responses and trying to find those injection patterns that produce acceptable responses - e.g. those emission responses that comply with legislative constraints. However, the application to more complex scenarios or systems bears some limitations on the methodology. For example, in functional testing, larger systems with a lot of variations (e.g. discrete switches) can be difficult to model using standard regression models (with more than 10 variations).

3.6.2 Current way of working and baseline technology

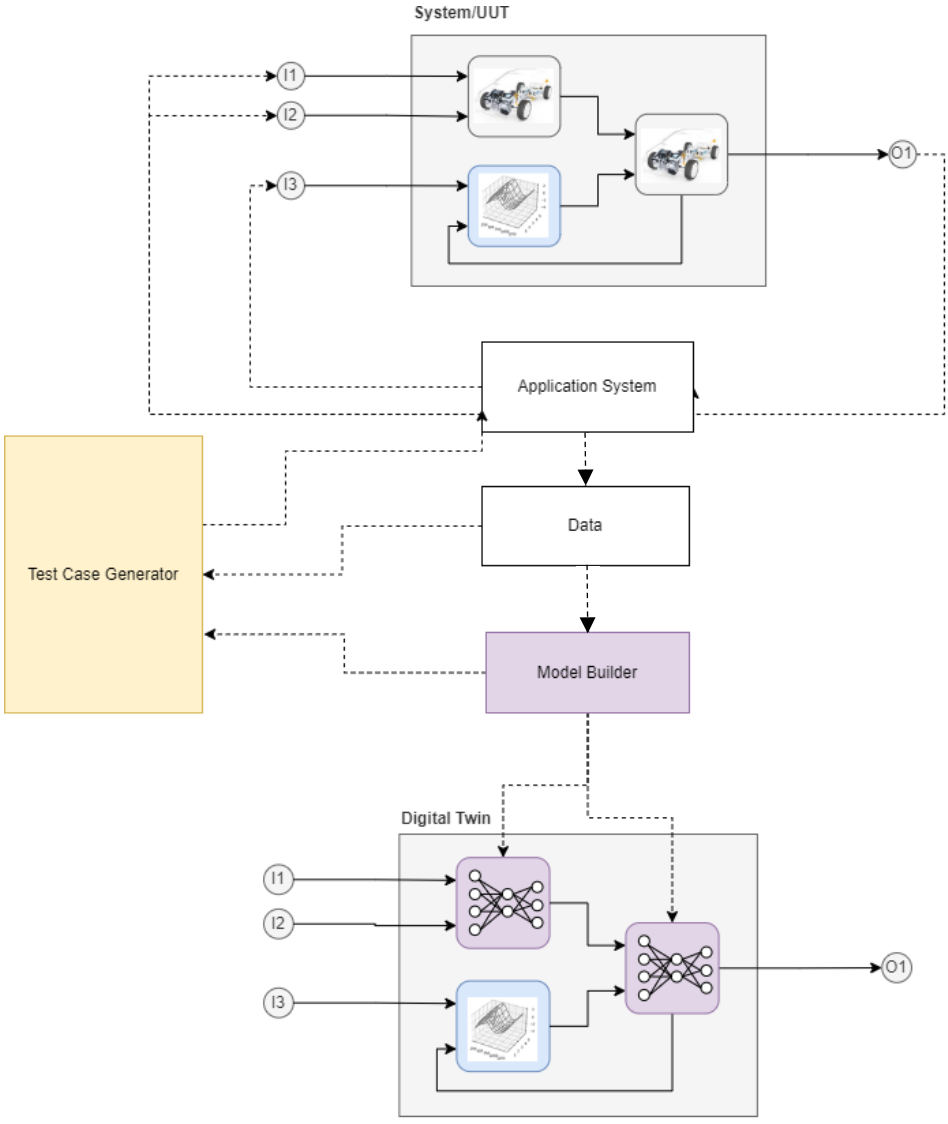


Figure 11 Current System Architecture

The current system setup consists of a unit under test (UUT), as well as a UML description of the UUT. The target is then to generate test cases based on a digital twin/model of the UUT. The goal could be to create test cases that improve predictive performance of the digital twin under a certain scenario.

3.6.3 Expected improvements in AIDOaRt

We are looking for methods to extend the model-based design space reduction to high-dimensional design spaces for functional scenarios. For this use case we need a systematic approach to model large systems and a methodology to reduce the design space for these large system structures. This approach will help to keep the number of measurements of complex systems to a minimum, which is important for costly measurements, tight production schedules and complex calibration tasks.

3.6.4 Relevant KPIs and definition of success

Definition of success for the 1st case story "CS1" (i.e., quantification of experiment information gain) and 3rd case story "CS3" (i.e., quantification of experiment result maturity): The computed information gain and maturity shall not contradict an expert's assessment and shall show the same trends as the expert's assessment. Experiments with significantly higher/lower information gain and maturity (relative to other experiments directly before or after in the development project) shall be detected.

Definition of success for the 2nd case story "CS2" (i.e., prediction of parameter and product KPI evolution): The forecast for future parameter and KPI values shall be as close as possible to the actual values as they evolve along the development project. Particularly, the forecast shall capture the trend of parameters and KPIs, e.g., if a KPI will get closer to or move away from its target value.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_MBT_UCS1	Model-Based Testing of Functional Scenarios

Table 13 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_MBT_UCS1_KPI_3.2_1	AVL_MBT_UCS1	KPI_3.2	Improvement of the ML methodology will lead to increase of the coverage and quality of the processes currently performed.	0%	30%

Table 14 Case Study KPIs

3.6.5 Validation methodology and process

The basic validation methodology for all 3 case stories is to apply the AIDOaRt results (i.e., models) to test data and check the performance with this test data. The test data will be provided by AVL. It will contain one or more datasets. Each dataset represents a whole development project, i.e., a time series of experiments with associated KPI results, parameter values and maturities.

- Validation methodology for the 1st case story "CS1" and 3rd case story "CS3": An AVL domain expert reviews the test data and assesses the information gain and maturities of experiments along the time series of experiments. The expert assesses whether information gain and maturity increase or decrease relative to prior experiments. The expert identifies experiments with significantly lower/higher information gain or maturity, i.e., outliers. This assessment is compared to the output of the model. The comparison is done qualitatively by an AVL expert.
- Validation methodology for the 2nd case story "CS2": Each time series of experiments in the test data is partitioned into 2 segments: Past and future. The model may use the experiments of the "past" segment as input and must forecast the parameters and KPIs of experiments in the "future" segment. The deviation between forecast and values in the "future" segment is measured, e.g., by using a simple sum of squared errors approach. The trend in the forecast and the test data is quantified, e.g., by computing the numerical first order derivative with respect to time for all parameters and KPIs.

3.6.6 Demonstrator setup and testbed

A possible setup consists of

- A simulation tool, e.g. Carmaker
- A Co-Simulation platform running the simulation tool in combination with ADAS controllers
- A testcase generation tool sending test cases to the simulation tool, e.g. AVL CAMEO

3.7 Case Study - AVL_ODP

3.7.1 Case study overview

The vision is to provide continuous verification and validation (V&V) in vehicle development processes based on systematic linking and management of involved information entities (simulations, physical tests etc.). Continuous V&V means to be able to check at any point in time whether any key performance indicator (KPI) of the current state of development can meet the target. Due to systematic linking, highly structured data emerges that accurately represents the system under development and its evolution along the development process. The logical next step is to develop solutions for gaining additional value from this structured data: Apply data analytics and create models to optimize vehicle development processes.

3.7.2 Current way of working and baseline technology

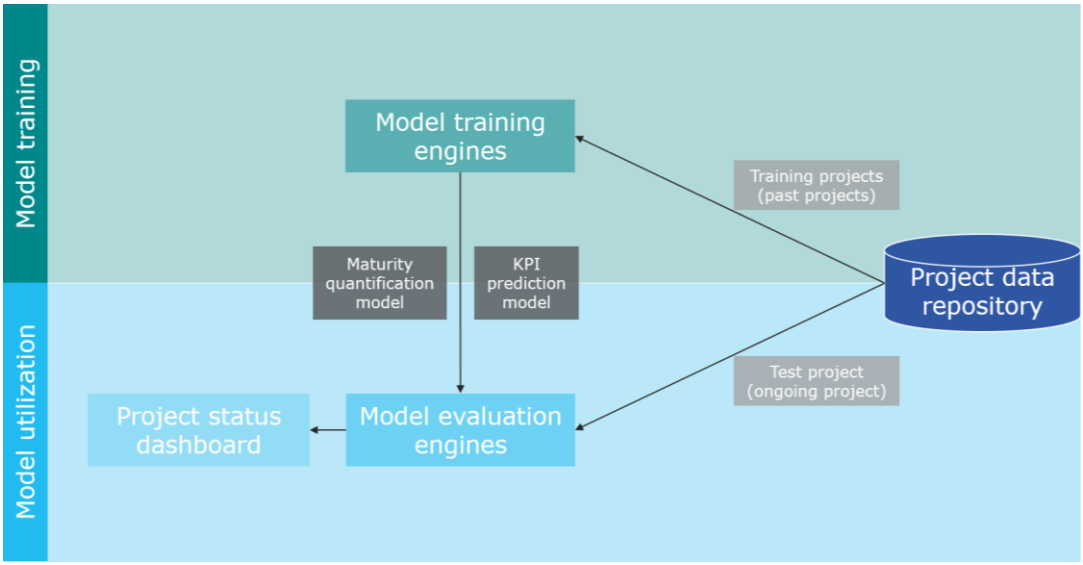


Figure 12 Current System Architecture

The current system architecture is a first proof-of-concept for this case study, i.e., applying ML methods to the optimization of vehicle development processes. The architecture is divided into two distinct blocks: Model training and model utilization. This architecture is a first prototype. It is not yet capable of addressing all use cases and requirements.

3.7.3 Expected improvements in AIDOaRt

Use the highly structured data in an ML-augmented approach for the optimization of development processes, i.e., apply data analytics and make inferences in to optimize the vehicle development process. Of particular interest are the prediction of KPI evolution and of KPI maturity.

3.7.4 Relevant KPIs and definition of success

Definition of success for the 1st case story "CS1" (i.e., quantification of experiment information gain) and 3rd case story "CS3" (i.e., quantification of experiment result maturity): The computed information gain and maturity shall not contradict an expert's assessment and shall show the same trends as the expert's assessment. Experiments with significantly higher/lower information gain and maturity (relative to other experiments directly before or after in the development project) shall be detected.

Definition of success for the 2nd case story "CS2" (i.e., prediction of parameter and product KPI evolution): The forecast for future parameter and KPI values shall be as close as possible to the actual values as they evolve along the development project. Particularly, the forecast shall capture the trend of parameters and KPIs, e.g., if a KPI will get closer to or move away from its target value.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
AVL_ODP_UCS1	Information gain quantification
AVL_ODP_UCS2	ML-based KPI prediction
AVL_ODP_UCS3	ML-based maturity determination

Table 15 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
AVL_ODP_UCS1_KPI_1.1_1	AVL_ODP_UCS1	KPI_1.1	Improvement of the time required to quantify the information gain of each experiment thanks to the analysis of the collected data.	0%	20%
AVL_ODP_UCS1_KPI_2.3_1	AVL_ODP_UCS1	KPI_2.3	Reduction time and effort required for integrating new experiments into a DevOps pipeline.	0%	20%

AVL_ODP_UCS2_KPI_3.1_1	AVL_ODP_UCS2	KPI_3.1	Automate of the processes of predicting project's KPIs which are currently manual.	0%	30%
AVL_ODP_UCS3_KPI_3.1_1	AVL_ODP_UCS3	KPI_3.1	Automate of the processes of predicting project's maturity which are currently manual.	0%	20%

Table 16 Case Study KPIs

3.7.5 Validation methodology and process

The basic validation methodology for all 3 case stories is to apply the AIDOaRt results (i.e., models) to test data and check the performance with this test data. The test data will be provided by AVL. It will contain one or more datasets. Each dataset represents a whole development project, i.e., a time series of experiments with associated KPI results, parameter values and maturities.

- Validation methodology for the 1st case story "CS1" and 3rd case story "CS3": An AVL domain expert reviews the test data and assesses the information gain and maturities of experiments along the time series of experiments. The expert assesses whether information gain and maturity increase or decrease relative to prior experiments. The expert identifies experiments with significantly lower/higher information gain or maturity, i.e., outliers. This assessment is compared to the output of the model. The comparison is done qualitatively by an AVL expert.
- Validation methodology for the 2nd case story "CS2": Each time series of experiments in the test data is partitioned into 2 segments: Past and future. The model may use the experiments of the "past" segment as input and must forecast the parameters and KPIs of experiments in the "future" segment. The deviation between forecast and values in the "future" segment is measured, e.g., by using a simple sum of squared errors approach. The trend in the forecast and the test data is quantified, e.g., by computing the numerical first order derivative with respect to time for all parameters and KPIs.

3.7.6 Demonstrator setup and testbed

The minimum demonstrator setup basically consists of a repository for the test data, an execution environment for the models and a GUI to visualize and assess the test data and model outputs.

3.8 Case Study BT (Alstom) - Case Study Railway Electric Propulsion System Development

3.8.1 Case study overview

BT (Alstom) UC: Solutions to automate data processing and transfer between different stages of the development process, multi-physics modelling and test data correlation. The potential outcomes of such an endeavor are improved design and development chain resulting in a more effective process, optimized solution customization, simulation and V&V test facilities, standards certification support, reduction of the overall costs, i.e. time to market, maintenance and life-cycle costs, monitored operations, energy efficiency, etc.

3.8.2 Current way of working and baseline technology

The current system architecture for the BT (Alstom) Use Case is presented in Figure 13. It consists of the following modules:

- Module 1: Requirements Management Tool.
- Module 2: Bid Engineer: Manual allocation of/response to requirements.
- Module 3: Propulsion controller.
- Module 4: Control SW Engineer: Manual analysis of test data and update of control model parameters.

These are connected using the following interfaces for data exchange.

- Interface 1: Customer requirements.
- Interface 2: Requirement's allocation / response data.
- Interface 3: Control model parameters.
- Interface 4: System test measurement data.

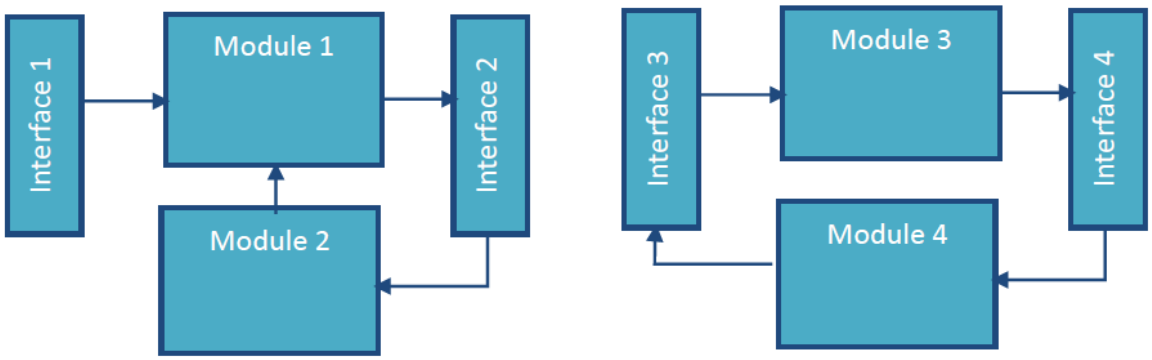


Figure 13Diagram capturing the current case study architecture

3.8.3 Expected improvements in AIDOaRt

Within the ADIOaRt project, Alstom aims to automate and improve their requirements engineering process using an ML solution that would analyze requirements and add adequate responses to each requirement.

Moreover, by using ML algorithms Alstom aims to automate the parametrization of specific models in the propulsion control system that depend on the physical properties of the corresponding hardware components. Today this is done by manually iterating parameter settings against measured data during systems testing.

3.8.4 Relevant KPIs and definition of success

The high-level goals of Alstom (BT) in the AIDOaRt project are:

- [CS_BT_Goal1] Improve the analysis of a customer specification for critical requirements and use AI to get recommendations for suitable actions.
- [CS_BT_Goal2] Automate the parametrization of control models of the propulsion system during physical system testing using AI/ML methods.

The following rationale has been applied in determining the metrics to gauge the success in each of the Use Case Scenarios:

Use Case Scenario 1 (BT_USC1):

- The requirements analysis which is essentially manual today will include automated process steps.
- Through the automated process steps productivity will improve by shortening the time for analysis.
- Through the automated incorporation of data from previous customer specification responses, the number of data sources actively managed and used in the process will be increased.

Use Case Scenario 2 (BT_USC2):

- The control model parameter tuning which is essentially manual today will include automated process steps.
- Through the automated process steps productivity will improve by shortening the time for tuning.
- By applying AI to process measured data the model accuracy will increase reducing the deviations to improve the predictability of the control model.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
BT_UCS1	Requirements Engineering recommender system
BT_UCS2	Automated model parametrization

Table 17 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
BT_UCS_1_KPI_3.1_1	BT_UCS1	KPI_3.1	Share of automated process steps in customer requirements analysis	0%	20%
BT_UCS_1_KPI_4.1_1	BT_UCS1	KPI_4.1	Relative time consumed for customer requirements analysis	100%	80%
BT_UCS_1_KPI_2.2_1	BT_UCS1	KPI_2.2	Share of data sources accessed and managed automatically for use in the customer requirements analysis process	0%	20%
BT_UCS_2_KPI_3.1_1	BT_UCS2	KPI_3.1	Share of automated process steps for control parameter tuning	0%	20%
BT_UCS_2_KPI_4.1_1	BT_UCS2	KPI_4.1	Relative time consumed for parameter tuning	100%	80%
BT_UCS_2_KPI_4.2_1	BT_UCS2	KPI_4.2	Model response deviation reduction with AI tuned parameters vs initial parameter set	100%	80%

Table 18 Case Study KPIs

3.8.5 Validation methodology and process

BT_UCS1 will be evaluated by comparing the actions recommended by the AI system with the actions recommended by an expert bid/advance engineer.

BT_UCS2 will be evaluated by comparing the real response of the propulsion system with the simulated response using the parameters generated automatically by the AI/ML system.

The approach to collect KPI values and measure performance will be as follows:

BT_UCS1:

- The function of the AI system will first be qualified by verifying that the actions recommended by the system match those recommended by an expert bid/advance engineer.
- The share of automated versus the total number of process steps will be computed.
- The time for the AI system to recommend actions will be compared to the time for a non-expert engineer to perform the same task.
- A demonstrated integration of the AI system with the current requirements management tool and the database will constitute a measure of the increased number of data sources used in the process.

BT_UCS2:

- The function of the AI system will first be qualified by verifying that the responses are within plausible parameter boundaries for a given propulsion application.
- The share of automated versus the total number of process steps will be computed.
- The time for the AI system to provide a set of tuned parameters will be compared to the time for the tuning analysis to be done manually for a given propulsion application.
- The deviation between the control response from the trained AI system determined parameter set and the real response from the propulsion system for a given application will be measured and compared to the corresponding response deviation from the initial parameter set.

3.8.6 Demonstrator setup and testbed

Alstom (BT) will be evaluating the AIDOaRt solutions for its use case in the following testbed setups:

BT_USC1

- IBM Rational DOORS requirements management system where the evaluated AI solution will be applied as a plug in or the equivalent thereof in a PC environment.
- Training data will be accessed from the DOORS system.
- The comparison data for evaluation will be generated in a PC environment with access to the DOORS system.

BT_USC2

- PC based control software environment where the evaluated AI solution will be applied as a plug in or the equivalent thereof.
- Training and evaluation data will be accessed from a high order physics-based simulation model in the PC environment.

3.9 Case Study CAMEA - AI for Traffic Monitoring Systems

3.9.1 Case study overview

Traffic monitoring system is usually 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. We are investigating possibility of enhancing current Traffic monitoring systems (within Transport and Smart Mobility domain) using AI. Thus, post-processing of currently extracted data can bring more information and thus make such systems more reliable or introduce new features. To maintain the reliability, senor needs to be properly calibrated. For the best possible user experience, auto-calibration (sensor position, elevation and azimuth angles, …) techniques should be employed. Within the UC, we are mainly targeting to low-power requirements using some of following proposed techniques - suitable (embedded) platform, pre-processing of data, balancing between load and power consumption, and AI guided configuration and setup. Such system can be then deployed to the field with possibility of autonomous operation with e.g. battery supply or solar power.

The current system architecture for the CAMEA radar sensors consists of the following modules:

- RCA – radar module with embedded processing and serial input/output interface
- ETH3RS – MCU for translation serial to TCP and for auto radar configuration

These are connected using CLI interface – TCP port, that is for RCA module translated to serial. It is intended for sending for text commands (sensor configuration and start) and for text messages sent back. There is also data stream interface – TCP port, that is for RCA module translated to serial. This is high data rate interface for point cloud data stream with given (configured) framerate.

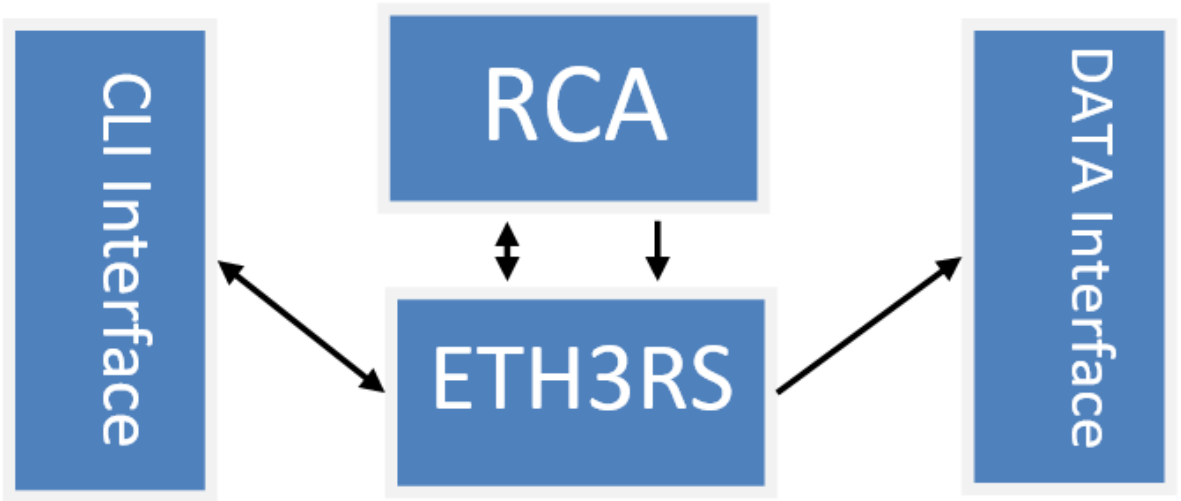


Figure 14 Radar system with configuration and data stream interface

3.9.2 Current way of working and baseline technology

Currently, there is no AI employed at the moment. As traffic monitoring systems are mostly video-based or alternatively radar-based, and they can serve for applications as travel time estimation or vehicle detection and classification. Using conventional methods for camera/radar data processing and classification, the system does not meet defined accuracy requirements. Without AI post-processing there is often not enough information and thus system is not as reliable as demanded. To maintain the reliability, sensor needs to be properly calibrated (setup of position and angles) which is done manually now. There is also 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.

3.9.3 Expected improvements in AIDOaRt

For the best possible user experience, auto-calibration techniques can be employed. Without sensor auto-calibration, any traffic monitoring system can be considered as credible. Manual calibration takes a lot of time and should be periodically checked. Considering low-power operation, AI can play its 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 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 given application keeping the lowest power consumption possible.

3.9.4 Relevant KPIs and definition of success

The focus of CAMEA use case 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 constrains for such parameters needs to be carefully done. The we are aiming to measuring both power consumption of RCA module and junction temperature of radar chip (part of RCA module). Power consumption and temperature with stock configuration will be measured and then we start with iterations of AI-tuned configurations. Reasonable reduction 20% in power consumption and 5 degrees Celsius in temperature is expected to be measured.

Use-case Scenario ID	Complete Name of the use-case scenario
CAMEA_UCS1	Enhanced Traffic Monitoring System
CAMEA_UCS2	Enable Initial Auto-calibration
CAMEA_UCS3	Enable Low-power Device Configuration

Table 19 Recap of use-case scenarios

For our use case, following KPIs are relevant:

Use-case Scenario	Project KPI ID	KPI instantiation and Definition	Base Value	Target Value	Use-case KPI ID
CAMEA_UCS3	KPI3.1	Automating radar configuration generation and best candidate selection which is currently manual.	0%	10%	CAMEA_UCS3_KPI3.1

Table 20 Relevant KPIs

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
CAMEA_UCS1	Enhanced Traffic Monitoring System
CAMEA_UCS2	Enable initial device autocalibration
CAMEA_UCS3	Enable low-power device configuration

Table 21 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
CAMEA_UCS3_KPI3.1	CAMEA_UCS3	KPI_3.1	Automating radar configuration generation and best candidate selection which is currently manual.	0%	10%

Table 22 Case Study KPIs

3.9.5 Validation methodology and process

Using our experimental radar sensor, we are planning to measure two values:

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

In case of temperature, stabilization of the value should be done. It will be necessary to wait minute or two to warm up a bit or cool down a bit. This is very necessary especially in case of 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 stabilization, and it can be measured immediately after configuration. However, this quantity will differ with load, radar should be fed with artificial/recorded data. although I 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 device with such data is not very easy. Thus, after some testing of the approach on the table, we should move radar outside. We will also use CAMEA infrastructure and make radar accessible via internet. There we can get some real vehicle passes so 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.

3.9.6 Demonstrator setup and testbed

Main part of the planned testbed is radar sensor itself. 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 own capabilities for junction temperature measurement. As there are no power meters, 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, 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 PC with defined period. Values will be collected and evaluated for each iteration. Individual tuned configurations will be generated using tools from project partner ABO.

3.10 Case Study AtelierB- Case Study CSY

3.10.1 Case study overview

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 (formal software). 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 modified to adapt to the change of

hypothesis.

Therefore we believe that machine learning can help engineers in some aspects of the B development, being those interactive proving or adapting proof to change of hypothesis. The interactive prover relies on automatic theorem proving but needs to be driven by a developer who builds a proof by assembling elementary bricks, addition of hypotheses, proof by contradiction, proof by cases, and calls to inductive or deductive rules (a formula A => B).

Within a system, there are proofs that solve several proof obligations and it is quite easy to automatically check if a proof solves more than the proof obligation for which it was designed; nonetheless, there can be proof obligations that would be solved with a small modification of the proof and those are impossible to detect automatically.

We would like to develop a module within the Interactive prover that can learn interactive theorem proving from the developer and adapt it to unsolved proofs of the system. In this respect, ClearSy can provide a system written in B, with specifications and proof, and develop solutions in Atelier B; ClearSy can also bring expertise in formal development and the specificity of the interactive prover.

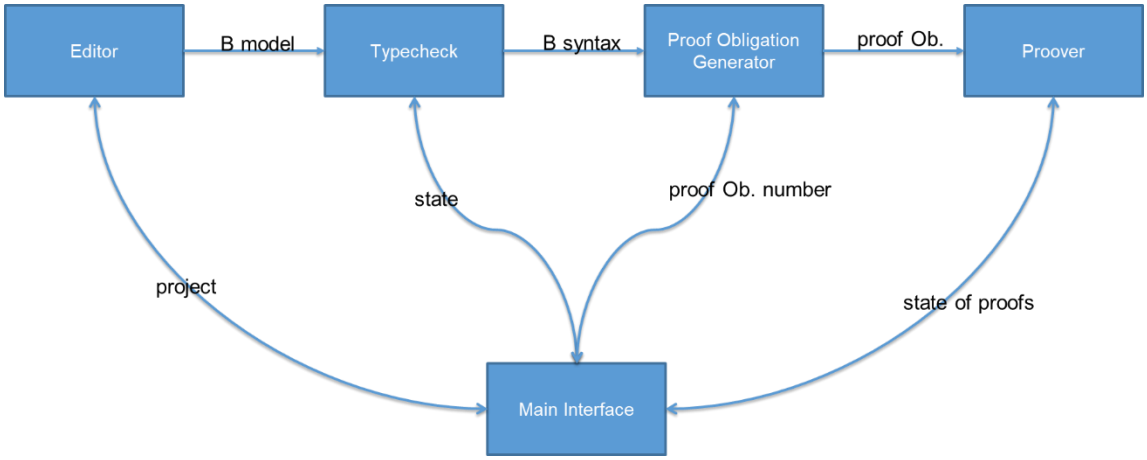


Figure 15 Diagram capturing the current architecture of the case study

The architecture of the AtelierB is represented in Figure 15. It consists of an editor and several tools managed by a same interface.

- The Main Interface gives the user information about the project he is working on, and the current state of the components. It can monitor the tool activity like type checking, proof obligation generation, automatic proving.
- The Editor is a text/code editor with some code dedicated feature like syntax checking and coloration, navigation between linked components …
- The type checker simply validates the components and their interdependencies
- The proof obligation generator creates for each operation of each component the associated proof obligations

The prover is a tool that can demonstrate the correctness of the proof obligations. For about 75% of them, the prover can automatically do the work, with human only assisting with parametrization. For the 25% remaining, a human prover is required who will manipulate the proof obligation with deduction, induction, syntactic equivalence in order to make the prover able to solve the proof obligation.

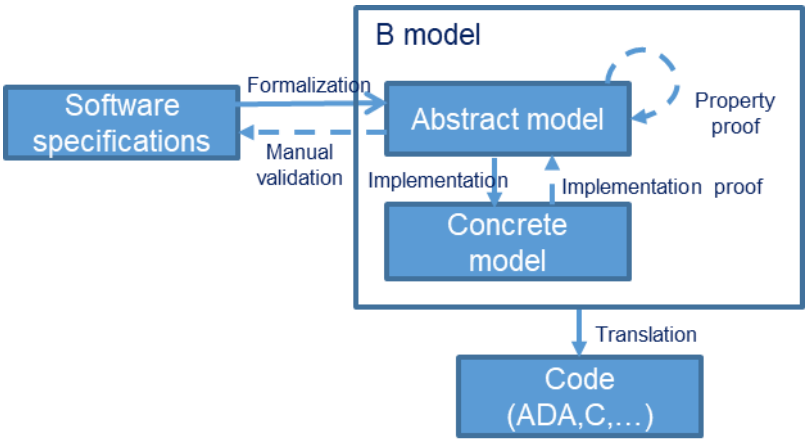


Figure 16 Diagram capturing the current case study flow

The current system flow for generating code in AtelierB is presented in Figure 16. It consists of the transformation of the following models (interfaces for data exchange):

- Software specifications: often written as plain text, it embeds the needs of the customer, in the form of sentences or more formal requirements
- B model as 2 kinds of representations:
  - The abstract model is a formal reflect of the software specification. It should stay close to the former in order to be recognized and validated by the designer.
  - The concrete model is another representation of the same software with implementation constraints. For example, it should be determinist and based on finite types.
- Destination Code: depending of the project it can be C, ADA or any kind of machine language as long as a generator exists. It is semantically close to the concrete model, with only the adapted changed syntax.

3.10.2 Current way of working and baseline technology

The Clearsy Case study is made of 6 independent scenarios for which the baseline differs.

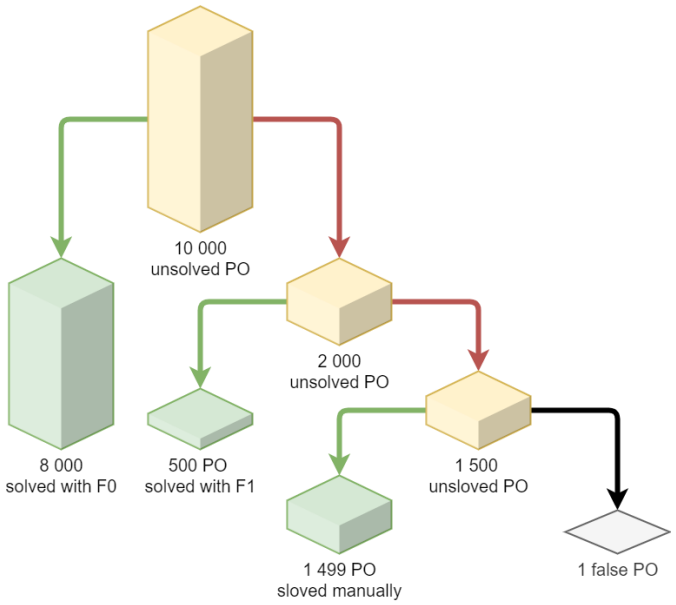


Figure 17, PO solving process in baseline (initial solving)

UCS1 is about classifying proof obligations (PO) in several categories. Depending on the complexity, PO can be solved automatically, and by different provers, solved manually, or not solved. In baseline, there is no classification, the total batch of PO, 10000 for a standard project, is tried on the weakest force solver (the fastest). This process lasts 20 minutes and treats 80% of the PO. Then, we try the second prover, in 2h it treats 25% of the 2000 remaining PO, 500 PO. The provers spend more time on a PO that cannot be solved than on a PO that can be solved.

For the last 1500 PO we can try if an existing proof can solve it or try to solve it manually at 16 PO per developer day. If errors exist in the model, some PO will not be provable, the project will be modified and a fraction of the PO will change and need to be reproved.

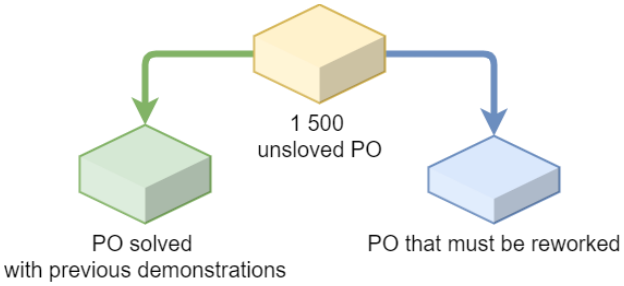


Figure 18, manual solving process in baseline

UCS2 is about the part after the automatic proving, when the developer is supposed to manually solve the PO. When a model has already be worked on, there are manual demonstration 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's just one command that must be added or removed. This has to be done manually and takes time.

UCS3 and UCS4 work also on the manual proving. 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 requires variation of the same demonstration.

UCS5 and UCS6 take a completely different path, they are 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 come from the obligation for the implementation to refine the specification.

3.10.3 Expected improvements in AIDOaRt

In UCS1, with help of AIDOaRt, we expect to be able to classify PO before trying the automatic prover, if we can do this, we will use provers only on PO that can effectively be solved, we estimate that it would reduce the total automatic proving time for a project from 2h20 to less than 30 minutes with the same result.

If we can also detect wrong PO, we could immediately warn the developer, who will be able to modify the model before trying to solve the PO and therefor saving a lot of time.

In UCS2, we expect that AIDOaRt could help to better interpret the demonstration to a finer level and to reuse or modify old demonstration or part of them to solved modified PO.

In UCS3 and UCS4, we hope to find a way to use the experience from the solved part of the project to help solve the rest. In UCS3 we imagine solving completely new PO, in UCS4 we more modestly expect the tool to suggest command to the user during the solving process.

UCS5 and UCS6, if we could automatically derive one model from the other as we expect to do with AIDOaRt, it could save modelling time. Because of their generated origins, we could also expect the PO created by an automatic refinement or abstraction to be particular, and maybe good candidate for automatic solving.

3.10.4 Relevant KPIs and definition of success

Here we will present metrics to measure AIDOaRt improvements on our case study. We have 6 Use-case scenario that are summarized in the following table.

Use-case Scenario ID	Complete Name of the use-case scenario
UCS1	PO classification
UCS2	Solve a PO with a variant of existing proofs
UCS3	Solve a branch of a PO using Reinforcement Learning

UCS4	Suggest a command in interactive proving
UCS5	Automatic refinement of specification
UCS6	Automatic specification (abstraction) of machine code

CSY use-case scenarios

In the table below, we have the KPI defined for this case study.

Use-case Scenario	Project KPI ID	KPI instantiation and Definition	Base Value	Target value	Use-case KPI ID
CSY_USC_1	KPI 1.1	Reduction of the proving time due to classification of the PO and reduction of unnecessary prover attempts	0%	50%	CSY_USC_1_KPI_1.1_1
CSY_USC_2	KPI 3.1	Use Machine Learning to solve X% of the manual PO of a new project	0%	10%	CSY_USC_2_KPI_3.1_1
CSY_USC_3	KPI 3.1	Use Machine Learning to solve X% of the manual PO of a project after a modification, when old proof existed	0%	25%	CSY_USC_3_KPI_3.1_1
CSY_USC_4	KPI 4.1	Reduction in time/effort required for proving using the interactive prover	0%	10%	CSY_USC_4_KPI_4.1_1
CSY_USC_5	KPI 2.1	Reduction in time/effort required for creating a model	0%	10%	CSY_USC_4_KPI_2.1_1
CSY_USC_5	KPI 2.1	Reduction in time/effort required for creating a model	0%	10%	CSY_USC_5_KPI_2.1_1

Case-study KPI definition

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
CSY_UCS1	PO classification
CSY_UCS2	Solve a PO with a variant of existing proofs
CSY_UCS3	Solve a branch of a PO using Reinforcement Learning
CSY_UCS4	Suggest a command in interactive proving
CSY_UCS5	Automatic refinement of specification
CSY_UCS6	Automatic specification (abstraction) of machine code

Table 23 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
CSY_USC_1_KPI_1.1_1	CSY_UCS1	KPI_1.1	Reduction of the proving time due to classification of the PO and reduction of unnecessary prover attempts	0%	50%
CSY_USC_2_KPI_3.1_1	CSY_UCS2	KPI_3.1	Use Machine Learning to solve X% of the manual PO of a new project	0%	10%
CSY_USC_3_KPI_3.1_1	CSY_UCS3	KPI_3.1	Use Machine Learning to solve X% of the manual PO of a project after a modification, when old proof existed	0%	25%
CSY_USC_4_KPI_4.1_1	CSY_UCS4	KPI_4.1	Reduction in time/effort required for proving using the interactive prover	0%	10%
CSY_USC_5_KPI_2.1_1	CSY_UCS5	KPI_2.1	Reduction in time/effort required for creating a model	0%	10%
CSY_USC_6_KPI_2.1_1	CSY_UCS6	KPI_2.1	Reduction in time/effort required for creating a model	0%	10%

Table 24 Case Study KPIs

3.10.5 Validation methodology and process

The previous KPI will be evaluated as such:

Use-case Scenario	Validation methodology
CSY_USC_1	On a representative selection of B projects, we will measure the time required for the automatic proving with and without classification of the PO. Unclassified or wrongly classified PO will be treated as before. The time required for classification will be taken in account.
CSY_USC_2	On a representative selection of B projects, we will count the number of manual PO that can be treated by machine learning without knowledge of the particular project.
CSY_USC_3	On a representative selection of B projects, we will count the number of manual PO that can be treated by machine learning after modification of the project.

CSY_USC_4	A developer will evaluate how much effort can be saved by using suggestions from the prover tool and how much they are relevant.
CSY_USC_5	Automatic refinement will be applied on the specification of several models, a developer will then evaluate how much effort can be saved by starting with the generated implementation compared to starting from nothing.
CSY_USC_6	Automatic specification will be applied on the implementation of several models, a developer will then evaluate how much effort can be saved by starting with the generated specification compared to starting from nothing.

3.10.6 Demonstrator setup and testbed

The setup of the demonstrator includes the AtelierB tool and its dependencies and a large collection of existing models from previous Clearsy projects. Those models are anonymized to preserve IP but can be considered as representative of usual railway developments. This collection will be part in two, one pool for training, the other for evaluation. To this setup will be added script and tools that are being created during the time of the AIDOaRt project.

3.11 Case Study HIB - Restaurants

3.11.1 Case study overview

The TAMUS system (previously called HIPPOS as reflected in some of the screenshots in the sections that follow) is an integrated point-of-sale system for restaurants that is offered by HI Iberia and deployed in 40+ restaurants across Spain.

It uses a complex architecture that meshes a number of small Android devices (for table orders and waiter usage), a per-restaurant local server (that unifies the ordering system, is in charge of updates in the location and also in charge of interacting with the secure credit card payments system and the cash box) as well as a cloud segment (that is in charge of backups, unification of all of the deployments and also provides managerial access to some aspects of the system via web).

All of this complexity is managed using a simple solution and until recently a small dedicated team could handle all of the operations in the DevOps cycle manually, from requirements for new features to development, to installation in the locations and then managing day-to-day issues. But the success of the restaurant chains using the system means that the development team is spread ever thinner. Thus, in AIDOaRt we intend as a general goal to alleviate some of the pain points in this system.

We present in the following sections a short overview of the current system operation and a list of the intended features that will be implemented using AIDOaRt approaches. This will lead naturally to

a dedicated proposal of an evaluation testbed for the system that can measure the benefits of AIDOaRt according to KPIs that quantify these improvements.

3.11.2 Current way of working and baseline technology

The current TAMUS system corresponds to the following architecture already presented in [D1.3].

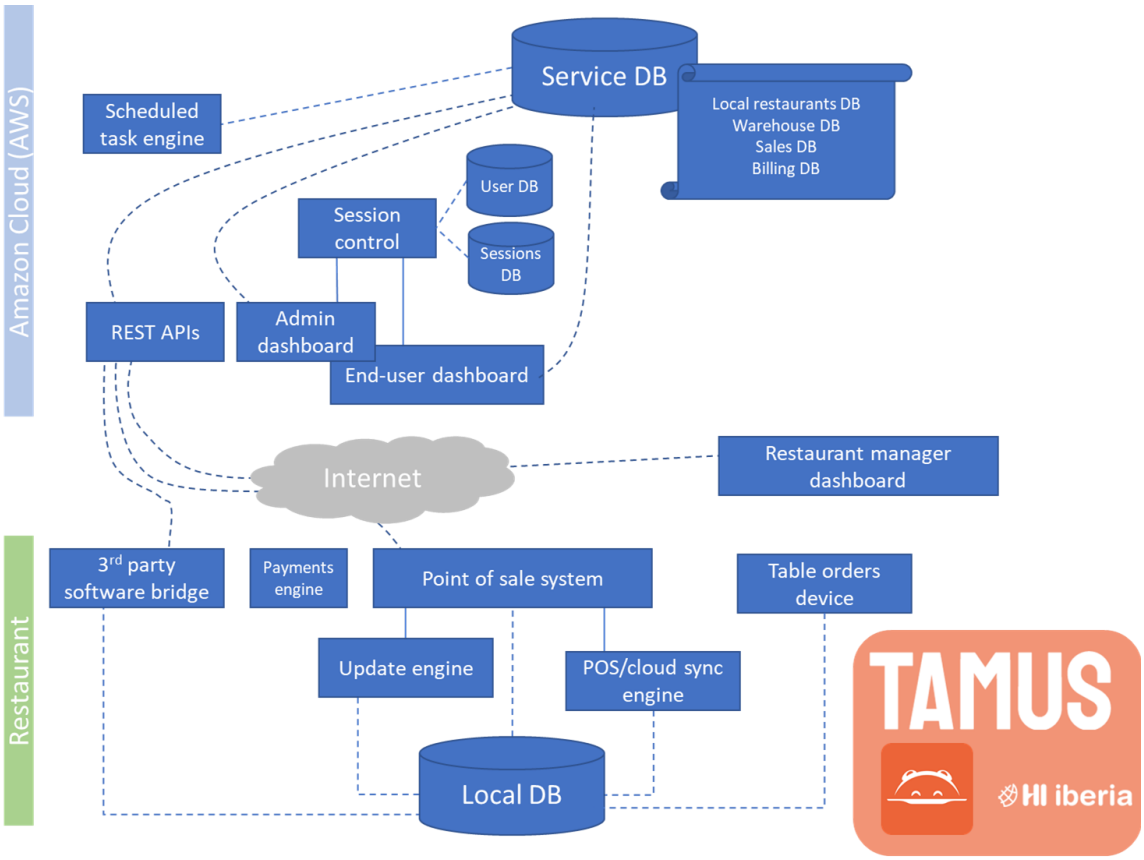


Figure 19 Restaurants UC baseline architecture

The elements most critical for the implementation of AIDOaRt improvements are the Scheduled task Engine, Update Engine (both for HIB_R03 and HIB_R04) while the technology baseline for HIB_R01 and HIB_R02 falls both outside of this architecture and involves all of the architecture as it deals with systemic aspects.

- For addressing HIB_R01 all of the modules above are important as the collection of all available logs is the essential aspect. The current baseline is the collection of logs from the operating system that runs the software (Linux for the POS system and the cloud server at Amazon and Android for the Table orders device).
    - There is no current analysis technology on these logs. They are manually inspected for issues by the HI Iberia developers.

```
INFORMACIÓN: [PRODUCTADD] Producto CROQUETAS CREMOSAS DE PALETILLA IBERICA añadido
jun 17, 2021 1:39:34 PM com.hi.utils.LogUtils log
INFORMACIÓN: f6849f22-2571-43f8-b7a7-4cb91f636910
========================== CIERRE TICKET (5585) -  MESA: bar1 COMENSALES: 2-0 ESCOTE ====================
TicketLines - |Producto: TERCIO ESTRELLA GALICIA    |Cantidad: 1.0  |Precio: 2,73 € | Total +(IVA): 3,00 €
TicketLines - |Producto: COCA COLA  |Cantidad: 1.0  |Precio: 2,55 € | Total +(IVA): 2,80 €
TicketLines - |Producto: CROQUETAS CREMOSAS DE PALETILLA IBERICA    |Cantidad: 1.0  |Precio: 10,00 €    | Total +(IVA): 11,00 €
--- TOTAL 16,80 € ---
-------- Formas de pago --------
Forma de pago: cash 16,80 €
================================================================================================================
jun 17, 2021 1:39:38 PM com.hi.utils.LogUtils log
INFORMACIÓN: Resultado del cierre de ticket (resultok) true
jun 17, 2021 2:39:27 PM com.hi.utils.LogUtils log
INFORMACIÓN: [ORDERPRINT] Imprimiendo en mesa TER5: Crema de Orujox1.0-3.5
jun 17, 2021 2:50:19 PM com.hi.utils.LogUtils log
INFORMACIÓN: [TABLEOPEN] Entrando en la mesa TER5
jun 17, 2021 2:50:22 PM com.hi.utils.LogUtils log
INFORMACIÓN: c79a8fa1-8dad-4ae8-915f-d440cefcb8f7
========================== CIERRE TICKET (5586) -  MESA: TER5 COMENSALES: 1-0 ESCOTE ====================
TicketLines - |Producto: Crema de Orujo     |Cantidad: 1.0  |Precio: 3,18 € | Total +(IVA): 3,50 €
--- TOTAL 3,50 € ---
-------- Formas de pago --------
Forma de pago: VISA 3,50 €
================================================================================================================
jun 17, 2021 2:50:23 PM com.hi.utils.LogUtils log
INFORMACIÓN: Resultado del cierre de ticket (resultok) true
jun 17, 2021 2:58:06 PM com.hi.utils.LogUtils log
INFORMACIÓN: [ORDERPRINT] Imprimiendo en mesa TER5: HUEVOS FRITOS, IBERICO, PATATAS Y PIMIENTOSx1.0-13.0||Copa DO Ruedax1.0-3.0
jun 17, 2021 3:04:02 PM com.hi.utils.LogUtils log
```

Figure 20 Example Logs from the Point-of-Sale system including an order and sales details.

- To address the requirements management, the current solution uses a common Trello board shared between the developers and the product manager. The Product Manager inserts cards with new features, classifies these in categories and assigns it to particular developers based on his insight. The expected goal of AIDOaRt is to automate these functions of the Product Manager.

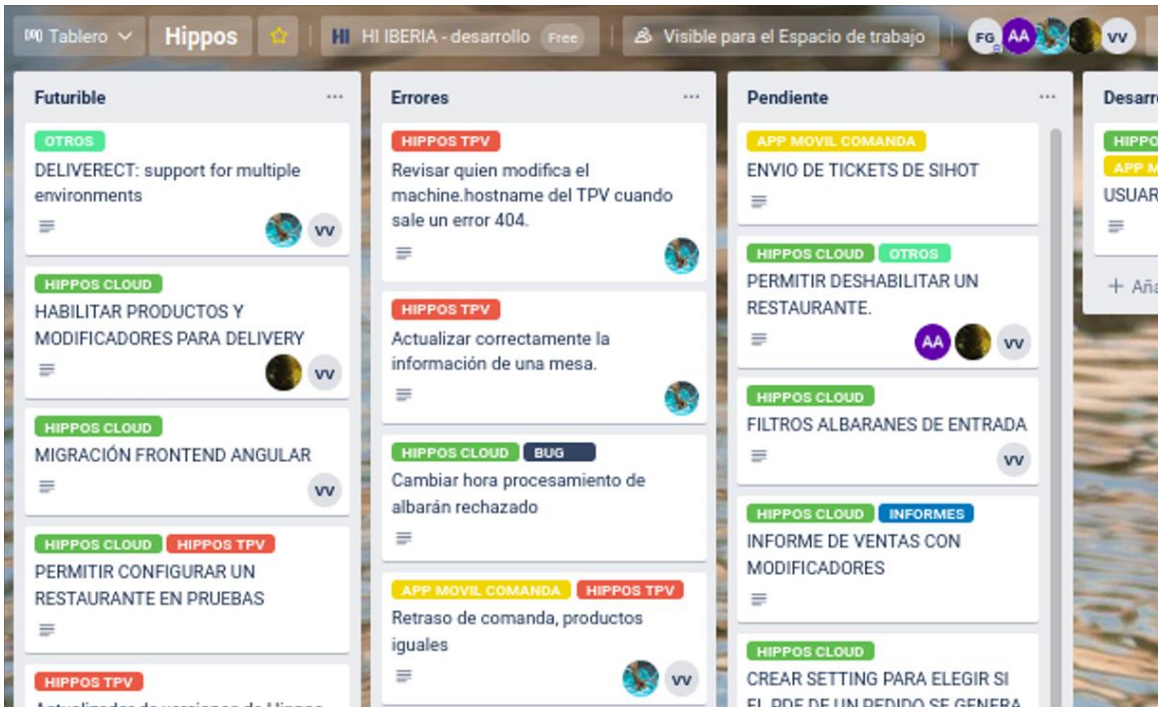


Figure 21 Sample of the Trello board for TAMUS. Tags mean topics of development and columns mention different phases of the development pipeline.

- For the HIB_R03 and HIB_R04 the key aspect is that new AI methods must interact with the existing structure of the assets used in producing new versions of the TAMUS system and

deploying it to the restaurants and cloud. The core element is the “TAMUS Bundle” which consists of a new version of the system (as a .jar archive), new needed assets such as images or other data plus an installation script and a copy of the updater program (also as a .jar file). The receiving end-system unzips the file, makes the changes mentioned in the installation script (typically moving files and deleting old ones) and executes the updater file. The objective of AIDOaRt here is to automate this process and add layers of testing for QA and storing traces for accountability of the process.

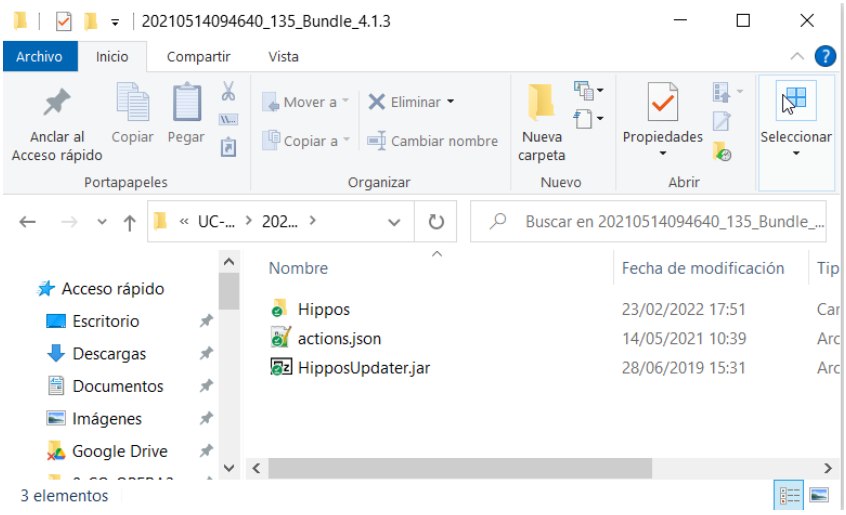


Figure 22 Contents of the bundle file for version 4.1.3 of TAMUS (labelled HIPPOS here)

```
{
    updateName: "Bundle_4.1.3",
    updateId: 135,
    updateSize: 28700000,
    timeRange: "00:00-23:59",
    checkForClose: false,
    actions: [
        {
            type: "file",
            action: "copy",
            source: "Hippos\\Hippos.jar",
            target: "%PROGRAMFILES%\\Hippos\\Hippos.jar"
        },
        {
            type: "file",
            action: "copy",
            source: "Hippos\\Readme.version",
            target: "%PROGRAMFILES%\\Hippos\\Readme.version"
        }
    ]
}
```

Figure 23 Detail of the actions.json file that contains the installation script for version 4.1.3 of TAMUS (labelled HIPPOS here)

3.11.3 Expected improvements in AIDOaRt

The improvements that HI Iberia intends to build into the TAMUS based restaurant Case Study are as follows:

- Improved capabilities to understand the stored logs of all of the elements in the system. This includes logs generated by the ordering Android devices, the local gateway and Point of Sale system that is installed at each restaurant and the cloud backend that synchronizes all of the locations' data. This will be done using:
    - Analysis of the regular logfiles produced by the underlying technologies such as Linux and Android operating systems, Nginx web servers, databases and point of sale terminals. These logs contain very low-level information but are essential to track the general health of the system. This will be done using regular log analysis tools such as Logstash or Graylog.
    - Analysis of higher-level information encoded in Natural Text such as the notes of the orders taken at each table, comments from the restaurant administrators. For this, a Natural Language log analyzer tool (HIB_logAnalyzer) will be built during the project using common NLP and text analytics libraries such as the Stanford NLP toolkit.
    - The goal is to combine both approaches above into a unified HIB_logAnalyzer that can build reports from collections of daily logfiles.
- Better management of system requirements without disrupting the current system based on Trello boards. This will entail interacting with the Trello API to achieve:
    - New cards created in the system are analyzed and automatically classified (priority and technical topic) using an AI engine that posts them with the appropriate Trello tags and in the appropriate sections on the board.
    - In addition, these new cards are pre-assigned automatically to a developer in the team based on an AI algorithm that takes into account inputs such as development skills and already assigned workload.
- Better management of the update process for the TAMUS system. New versions of the system will be subject to a range of tests selected by AI in order to maximize QA of the new version. Traceability of all this process will be also automatically maintained.
- Finally, better management of the deployment process for the new versions of the TAMUS system. This will automate leveraging on extensions to the current system of "TAMUS Bundles" and the deployment of the new versions will be subject to the tests to maximize quality and automatically keep the process accountable.

3.11.4 Relevant KPIs and definition of success

For the Case Study Restaurants we defined in the work leading up to the deliverables in M9 the following Use Case Scenarios that mirror the four main requirements of the Case Study. For each of

them the full definition and step-by-step explanation of inputs, outputs and outcomes can be found in the deliverable D1.3. In summary, they correspond to the following goals and definition of success:

- **HIB_UCS1:** we expect that by analyzing the logs of the different parts of the system with AI techniques (specific for the analysis of logs but also using NLP tools) we can find better the issues in a given installation of TAMUS and perform predictive maintenance tasks better at the same level or better that the human operators.
- **HIB_UCS2:** the requirements and support tickets are now collected using a manually managed Trello board that the product manager oversees and uses to signal priorities to tasks and assign them to particular developers. This process should be automatized in AIDOaRt using AI to read the contents of the Trello cards and providing at least a first classification that enables the product manager to quickly sift through the system suggestions and adjusting accordingly.
- **HIB_UCS3**: our current system generates versions of TAMUS on a manual basis. Only whenever the product manager finds that a new version must be issued, the process of packaging and launching the version is performed. We expect that this is now automatized according to DevOps principles.
- **HIB_UCS4**: the current system updates are performed semi-automatically, with some aspects such as distribution of the updates to the different restaurants being done on a manual basis. We expect that these can be fully automatized with AIDOaRt. In addition, we expect to collect valuable insight from the process: from telemetry (when it was installed, how long did it take) to other more novel aspects (e.g., collecting natural language comments from support tickets opened for each version and looking for patterns).

We propose the following KPIs to measure progress in the implementation of the AIDOaRt features in the Restaurants Case Study.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
HIB_UCS1	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.
HIB_UCS2	AI augmented environment for the management and verification of the requirements
HIB_UCS3	AI enabled deployment for continuous integration enforcement.
HIB_UCS4	Automatic updates to the code for the POS system including AI for compatibility and issue tracking.

Table 25 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
HIB_UCS_1_KPI_1.2_1	HIB_UCS1	KPI_1.2	Reducing time spent on issues detection base on logs	0%	96%
HIB_UCS_1_KPI_1.1_1	HIB_UCS1	KPI_1.1	Improve the system performance on detection of specific issues	0%	95%
HIB_UCS_2_KPI_3.1_1	HIB_UCS2	KPI_3.1	Coverage of the automatic system reading the Trello board in terms of cards fully processed (input into the system for analysis)	0%	75%
HIB_UCS_2_KPI_1.1_2	HIB_UCS2	KPI_1.1	Precision of the assigning of tags to cards as compared to human operator	0%	75%
HIB_UCS_2_KPI_1.1_3	HIB_UCS2	KPI_1.1	Precision of the assigning of tags to developers as compared to human operator	0%	75%
HIB_UCS_3_KPI_1.2_2	HIB_UCS3	KPI_1.2	Success in generating valid versions of the system compared to product manager approach	0%	50%
HIB_UCS_4_KPI_1.1_4	HIB_UCS4	KPI_1.1	Volume of generation of insights from the update process for a given version	0%	500%
HIB_UCS_4_KPI_3.1_2	HIB_UCS4	KPI_3.1	Reduction on the time spent on detecting missing software updates	0%	75%

Table 26 Case Study KPIs

3.11.5 Validation methodology and process

Some of the proposed KPIs can be directly measured by test tools as the values can be calculated from objective tests. But for many of the proposed KPIs they will require a comparison with the opinion of the human product manager. This will be done using checklists in which the product manager validates or rejects the proposed actions by the automation system.

A more thorough, qualitative approach will be proposed for subsequent iterations to collect the general satisfaction of the product manager as compared to the previous state of practice (little automation in the case of TAMUS). This will not be used to feed the general project KPIs since no such qualitative aspects are observed but it will be a good measurement of overall progress from the

developers and product manager point of view that can be incorporated in e.g., sales pitches for AIDOaRt enabled solutions.

For the validation, a data collecting campaign will be launched when the first integrated version of the Case Study is delivered as part of deliverable D5.2.

3.11.6 Demonstrator setup and testbed

The demonstrator of the Use Case Restaurants will be deployed in a realistic testbed that includes of the required software components and hardware devices in a manner similar to what's used in a regular TAMUS installation in a restaurant:

- A network environment using wired network for the local server and wireless for the ordering devices.
- x86 PC running Windows to simulate the Point of Sale system as the center hub for the restaurants. This will be connected to appropriate hardware such as a cash register and card reader and connected to the provided network.
- Android devices with the TAMUS ordering application installed and connected to the provided network via Wi-Fi.
- A replica of the TAMUS cloud system installed in a similar deployment in the Amazon AWS cloud.

This environment is supposed to be flexible enough to incorporate other elements not mentioned in this description if eventually needed (e.g., a printer for receipts). The initial description is valid for the first iteration of the system evaluation but will be updated as needed for future evaluations.

3.12 Case Study 8 - SPMP- Smart Port Monitoring Platform

3.12.1 Case study overview

The Smart Port platform is an Industry 4.0- Big Data solution in charge of monitoring the activities of a port in real time, through the analysis of data coming from sensors (IoT) installed in port Cranes/Truck, vessels and information systems (legacy and external systems).

Thanks to this solution, the port/terminal operators can better control in real time, the productivity and problems of all the operations being carried at ports. It also allows you to analyze historical data to review past productivity and problems as a forensic review. Another major objective is to detect specific or recurring bottlenecks in the port, so the port will know which problems are causing a greater loss of productivity and consequently a greater economic loss. For example, thanks to this solution the port will be able to determine on the basis of the data which option is better to operate more ships or containers, whether to reorganize the work groups or to acquire new machinery.

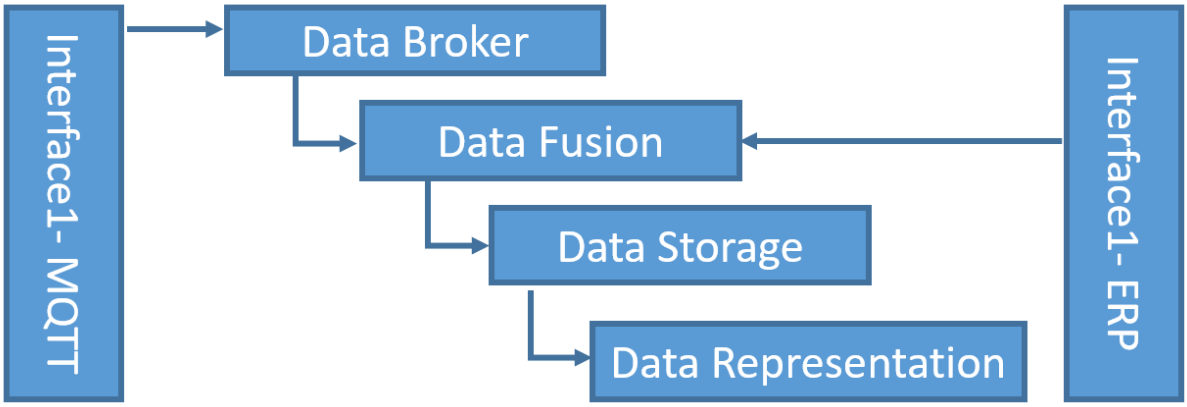


Figure 24: SPMP case study architecture

The current system architecture for the SPPM consists of the following modules:

- Module 1: Data Broker. Receives data from IoT Sensors
- Module 2: Data Fusion. Pre-process all data (IoT data and Port operation data)
- Module 3: Data Storage. No SQL database that stores all the processed data
- Module 4: Data Representation. Dashboards

These are connected using the following interfaces for data exchange.

- Interface 1.MQTT – Obtains data from all the cranes and vehicles
- Interface 2- ERP – port operations information

The solution can be deployed either in the cloud or on-premises.

Thanks to the AIDOaRt Project, it is expected to improve the development, deployment and monitoring phases of the platform.

3.12.2 Current way of working and baseline technology

The main challenges of SPMP are the definition of the architecture, the process of the large amount of information produced (Big data platform), and the customization needed for each port and terminal.

The sizing of SPMP is not trivial. In fact, a typical problem is the correct dimensioning of an infrastructure for given demands, avoiding over-dimensioning and, thus, higher monthly operational costs. Moreover, each client could have a different cloud provider or a specific on-premise infrastructure. Currently the deployment of the solution is made manually. Having a deployment of the infrastructure in an automatic way would be a plus since there are so many nodes and software components involved and it is not easy to carry out the deployment.

On the other hand, monitoring the running platform is another aspect to control since both the nodes and the software components can have temporary or permanent failures. In addition to this, monitoring the flow of data and ensuring that the information is processed correctly without loss of data is another desired improvement. This functionality aspect affects the system recommendations that can vary considerably if not all the data is considered and in the correct order, resulting in a great negative impact on productivity.

Monitoring is one of the aspects that has a lot of potential for improvement, as currently the infrastructure is partially monitored using elasticsearch and kibana to monitor the status of the nodes. Many of the problems are beyond the scope of the monitoring system.

Some of the current problems escape the monitoring system and may take time to be detected. The aim is to improve the monitoring system so that it detects most problems in both the infrastructure and the deployed solution. In addition, all the information gathered by the monitoring system could be used to determine anomalies and patterns of behavior, which can help us to predict future problems, and may even suggest infrastructure resizing to improve the quality and availability of the solution.

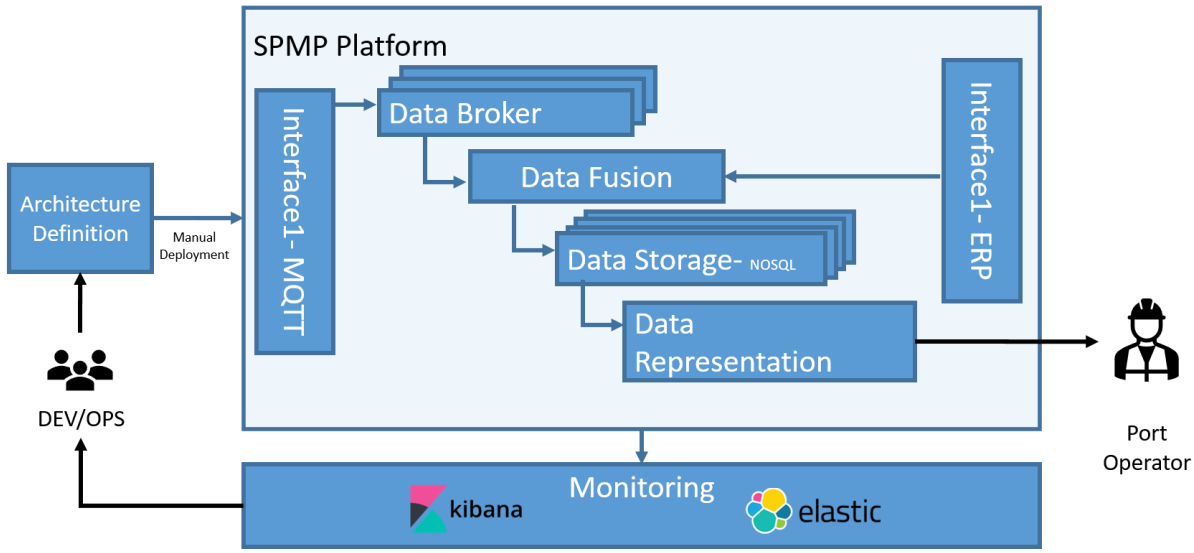


Figure 25: SPMP Current Situation

3.12.3 Expected improvements in AIDOaRt

Thanks to Prodevelop's participation in the AIDOaRt project, the following improvements are expected to be achieved:

- Use of Infrastructure as Code (IaC) solutions to have automatic deployments independent from the cloud provider.

- Redefine the current motoring system using Prometheus + Grafana. This new system will monitor both the status of the nodes and the status of the applications. All the information collected will be used by AIDO-aRt AI-services/component to improve to improve the availability of the system (objective 24/7) and to improve the recommendations of the system.
- Use the monitoring Information to detect anomalies and behaviour patterns in the solution.
- Use the monitoring information to predict future problems (Self-learning) ant try to automatically correct them (self-healing), being able to create new nodes and migrate or run software components between nodes
- To have a dynamic platform able to adapt to the system needs using reconfiguration techniques.

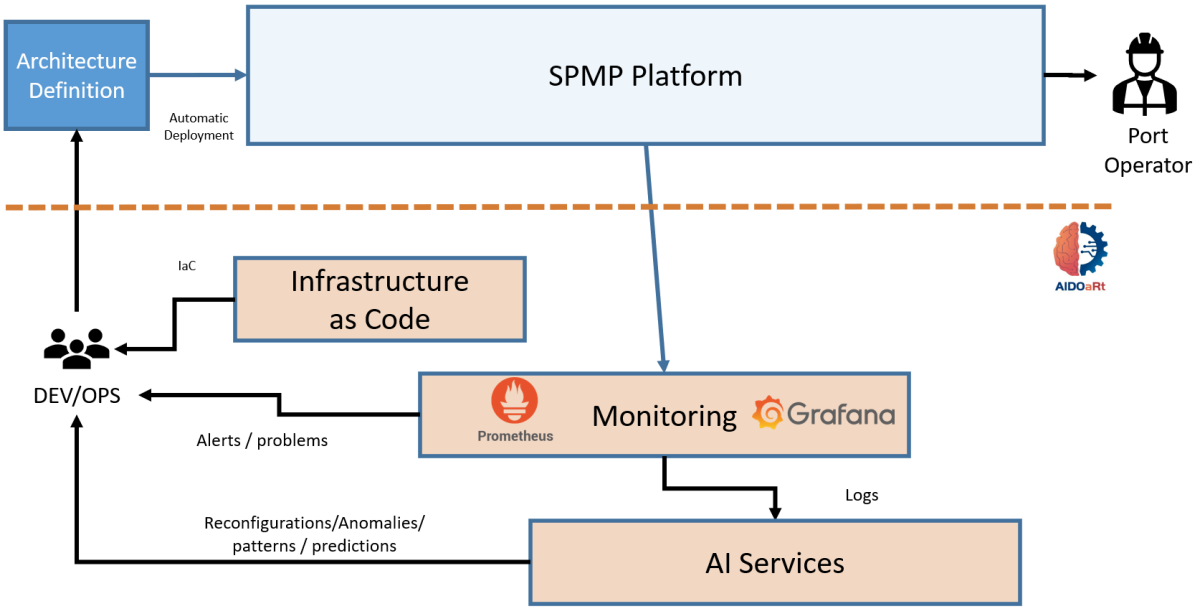


Figure 26: SPMP Improvements

3.12.4 Relevant KPIs and definition of success

The following table describe the relevant AIDOaRt KPI for the Prodevelop use case. For each relevant KPI, a definition of how each KPI will be measured in this particular UC is provided, with the base lain and the expected improvement at the end of the project after finalizing the UC.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
PRO_UCS1	Automatic deployment of the solution based on Infrastructure as code

PRO_UCS2	Automatic validation of the platform provisioning
PRO_UCS3	Detection of anomalies of the platform performance
PRO_UCS4	Detection and correction of service interruptions
PRO_UCS5	Resizing of resources based on current workload

Table 27 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
PRO_UCS_2_KPI_3.1_1	PRO_UCS2	KPI_3.1	Automate the validation of the infrastructure after each redeployment. Currently the validation is made manually. For this KPI, the % of automated test out of the total number of tests will be measured.	0%	20%
PRO_UCS_3_KPI_3.1_1	PRO_UCS3	KPI_3.1	Automate the detection of anomalies in the infrastructure. For this KPI, the % on anomalies automatically detected out of the total number of anomalies will be measured.	0%	20%
PRO_UCS_4_KPI_1.1_1	PRO_UCS4	KPI_1.1	Detects platform design problems thanks to an infrastructure monitoring platform. For this KPI, the % of design problems detected by the system automatically will be measured.	0%	50%
PRO_UCS_4_KPI_3.1_1	PRO_UCS4	KPI_3.1	Be able to automatically detect and correct system crashes base on Self-leaning and Self-healing techniques. For this KPI, the % of recurrent system crashes automatically solved will be measured	0%	20%

Table 28 Case Study KPIs

3.12.5 Validation methodology and process

In order to evaluate the improvements obtained after finalizing the use case, metrics and a methodology for their evaluation must be established. This section describes the evaluation methodology to be applied to evaluate the Prodevelop Use Case.

The evaluation will consist of a quantitative validation and a qualitative validation, in order to assess the performance improvements obtained after the implementation of each use case scenario.

Some of these improvements may be due to the application of a single Use Case scenario, while others may be due to the application of several use case scenarios together.

A quantitative approach will be taken so to examine Key Performance Indicators [KPI] that will be measured systematically on repeated occasions to observe their evolution. To do this, a new platform monitoring system will be implemented and most of the data needed to measure the KPI will be registered in that platform, for the rest of needed data an alternative mechanism bases on log files or excel files will be provided

On the other hand, a qualitative approach will be taken by gathering and analysing statistical metrics obtained with questionnaires, and with feedbacks from the partners involved in the development of the Use Case Scenarios. These qualitative metrics will yield valuable insights on the subjective perceptions of the project´s stakeholders regarding its resulting status along several dimensions, by supplying concrete, specific statistical measurements on the aforementioned subjective perceptions. Thanks to this qualitative approach, it would be possible to evaluate other facets of the Use Cases that are difficult to measure only with the quantitative approach.

The qualitative approach and the quantitative approach will finally be combined and synthesized so to arrive at a logical, complete, and conclusive verdict that unequivocally serves to evaluate PIACERE project´s results collectively and objectively.

3.12.6 Demonstrator setup and testbed

In order to develop the use case, the hardest part is to simulate the real workload that the system has to cope with by processing a large amount of data, so that we can replicate the problems in the test environment.

For this reason, Prodevelop will provide real anonymized logs of the system, so that we can perform analysis on them.

In other cases, controlled tests can be carried out in development environments and once tested in pre-production. These tests will be carried out by Prodevelop.

3.13 TEK_EEA — Agile process and Electric/Electronic Architecture of a vehicle for professional applications

3.13.1 Case study overview

TEK (Tekne SRL https://en.tekne.it/) is an Italian medium enterprise that is active in the automotive sector. TEK produces special vehicles for professional applications whose realization, according to customer needs and market opportunities, can vary from the customization up to the development of a new vehicle.

In the use case “Agile process and Electric/Electronic Architecture of a vehicle for professional applications”, TEK develops the demonstrator of a Prognostics and Health Management (PHM) system for monitoring, diagnostics, and preventive maintenance of the on-board electronic, which is applied to an electric minivan prototype.

The use case focuses on three aspects among those that TEK considers prominent for improving its industrial development process: (1) 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—AI/ML based components of the PHM demonstrator in the AIDOaRt case.

3.13.2 Current way of working and baseline technology

TEK technological baseline — The TEK development process is mildly model-based. Class diagrams are used for defining the structure of objects and their relationships, State Machine for their internal details, and Sequence diagrams for their interactions. But it does not happen that the core of the specifications lies in the models; moreover, we do not apply any unified method.

For example, requirements are created and modified in a Microsoft Word file that is the specification document. The textual parts of the specification that correspond to requirements attributes are tagged (such as the name, the statement, the level of compliance). Tags enable a Visual Basic for Application program to extract textual parts and to enter them in a Microsoft Excel workbook. This is enriched with the description of the test cases, collects the test results, and is used for requirements tracing. Test design, execution, and results interpretation are manual.

Regarding the AI/ML technology, TEK carried out some experimentations oriented to detect and classify mobile radiofrequency sources and, currently, is in a preliminary phase of experimentation of the use case.

TEK use case baseline — Figure 27 shows the functional blocks and the data flow of the system that TEK currently uses for the use case preliminary experimentation.

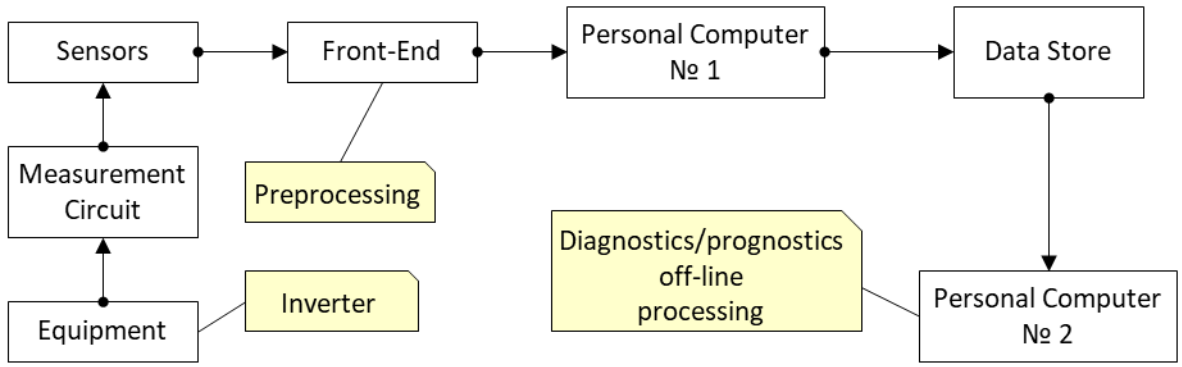


Figure 27 System currently used for the use case preliminary experimentation

The use case deals with control, diagnostics/prognostics, and maintenance of the on-board power electronics of electric vehicles—the preliminary experimentation considers the inverter. The quantities to be measured are voltages, currents, and temperatures, some of them require an interface circuit for adapting their ranges to those of the sensors. The measured data are pre-processed by a front-end (microcontroller and FPGA) whose output is collected and stored by a personal computer. Currently, the elaboration for diagnostics/prognostics runs off-line on another personal computer.

The case study aims at a system where the diagnostics/prognostics processing runs near real-time on a vehicular computing platform with cloud connection capability (Figure 28).

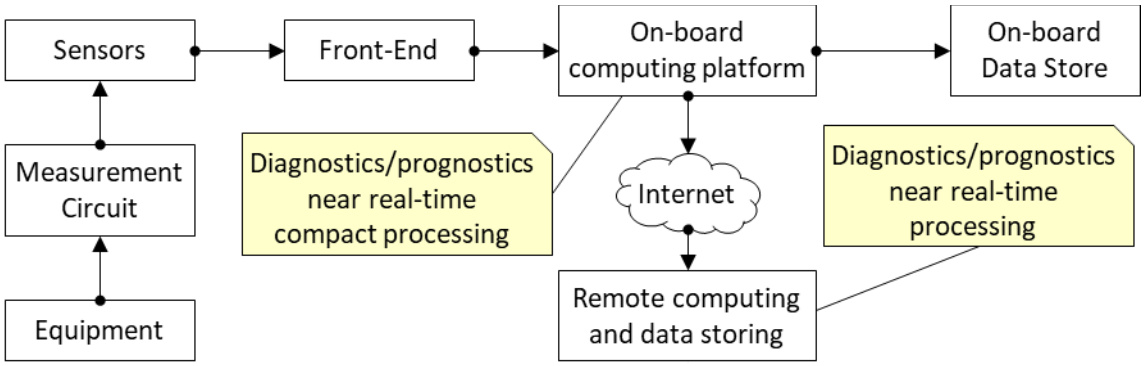


Figure 28 System the TEK use case aims at

The on-board processing is said compact because it has computational limitations due to the resources that are available, the remote computing can provide full capabilities.

3.13.3 Expected improvements in AIDOaRt

Among the improvement aspects of its industrial development cycle, both as technological competence and tools in use, in AIDOaRt TEK considers the following ones that are regarded as prominent:

a) **support to the design-time test**, of the models as well as of the target component that the design selects for the implementation (this can require simulation/emulation or rapid prototyping);
b) **run-time test automation**, e.g. semi-automatic ways for synthesizing enhanced models for defining the tests, for carrying out the tests, and for interpreting the tests results;
c) **AI/ML based components** in the PHM demonstrator, e.g. to predict imminent failures of the physical system which the software system acts on.

In the following, we explain these aspects through the relation that each of them has with one of the three *use case scenarios* that constitute the TEK use case (namely "Design choices verification", "Run-time verification", and "Operating life monitoring") and through the functionalities that are employed in the use case scenario. The AIDOaRt Framework is expected to provide these and other functionalities, each of which is described by its *case story*.

TEK_UCS_01 (support to the design-time test) — Figure 29 represents the *use case scenario* TEK_UCS_01 *as a UML use case diagram* and the *case stories* CS_01 ⋯ CS_06 *as UML use cases*.

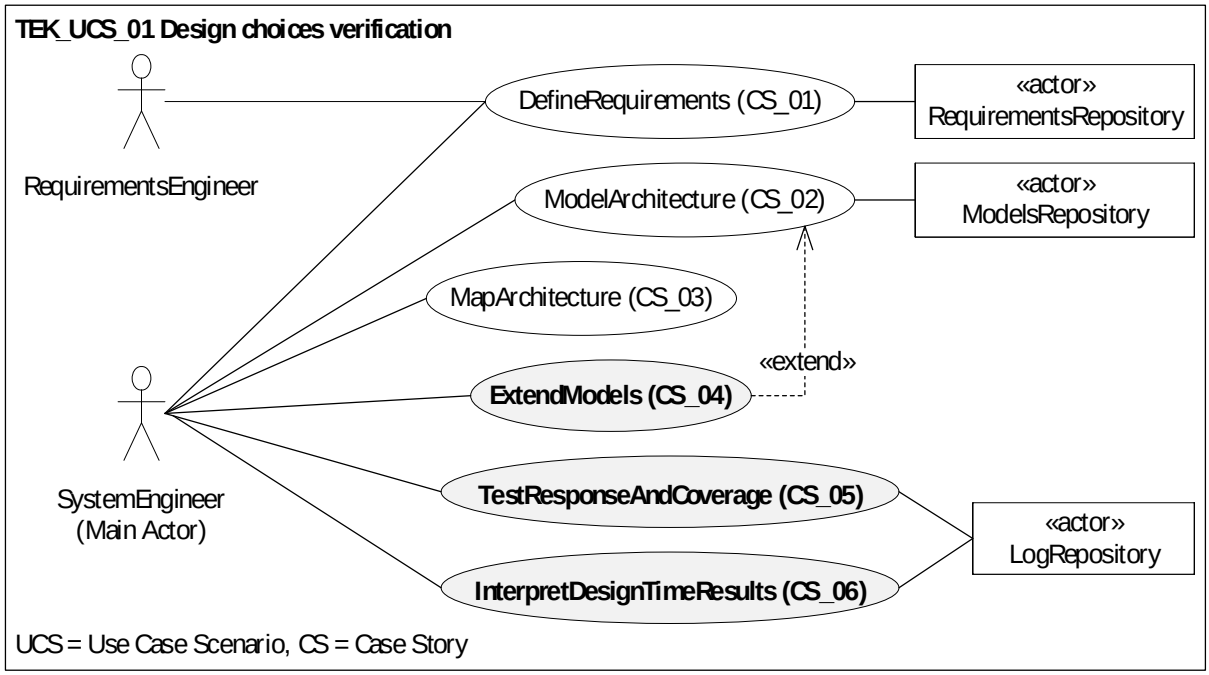


Figure 29 Use case scenario TEK_UCS_01 "Design choices verification"

TEK_UCS_01 focuses on the cases stories CS_04, CS_05, and CS_06.

- **CS_05** verifies in a semi-automatic manner, at design time, (1) if the models *cover* the requirements and if they *work*, as well as (2) the *adequacy* of the *target components* (that will be implemented, or reused, or provided in some way) on-which/with-which the system engineer has in mind to map/realize the architecture. We consider the second part essential to confirm the design choices (it can require some work to emulate/simulate the target components or even rapid prototyping).

- **CS_06** interprets the test results in semi-automatic manner and supports the successive choices of the system engineer.
- **CS_04** synthesizes in a semi-automatic manner the *extended models*, these describe the test, as well as are the basis for executing the test and interpreting the results—both CS_05 and CS_06 need them.

TEK_UCS_02 (run-time test automation) — The use case scenario TEK_UCS_02 is depicted in Figure 30. Among all its case stories CS_07 ··· CS_11, it focuses on CS_09, CS_10, and CS_11.

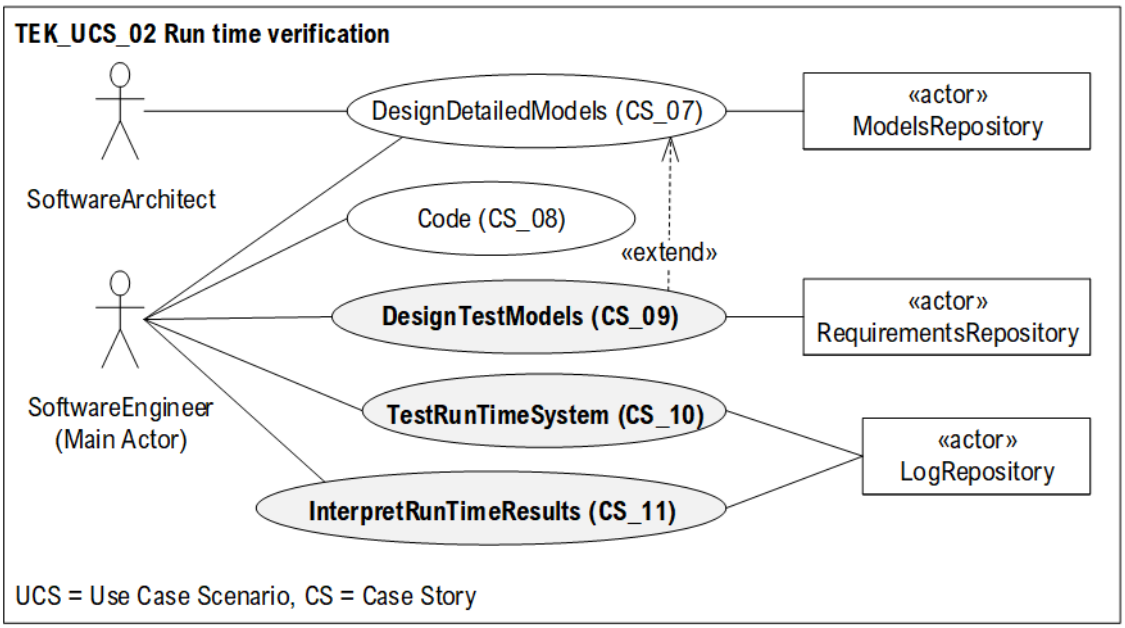


Figure 30 Use case scenario TEK_UCS_02 "Run-time verification"

In semi-automatic manner:

- **CS_09** synthesizes the extended models that describe the test;
- **CS_10** carries out the test;
- **CS_11** interprets the test results.

TEK_UCS_02 is similar to TEK_CS_01, but it applies to all the necessary run-time verifications of the real components after they have been implemented, while TEK_CS_01 applies to the confirmation of the design choice.

TEK_UCS_03 (AI/ML based components) — The use case scenario TEK_UCS_03 is depicted in Figure 30. Its case stories are CS_12, CS_13, and CS_14, and they are all relevant.

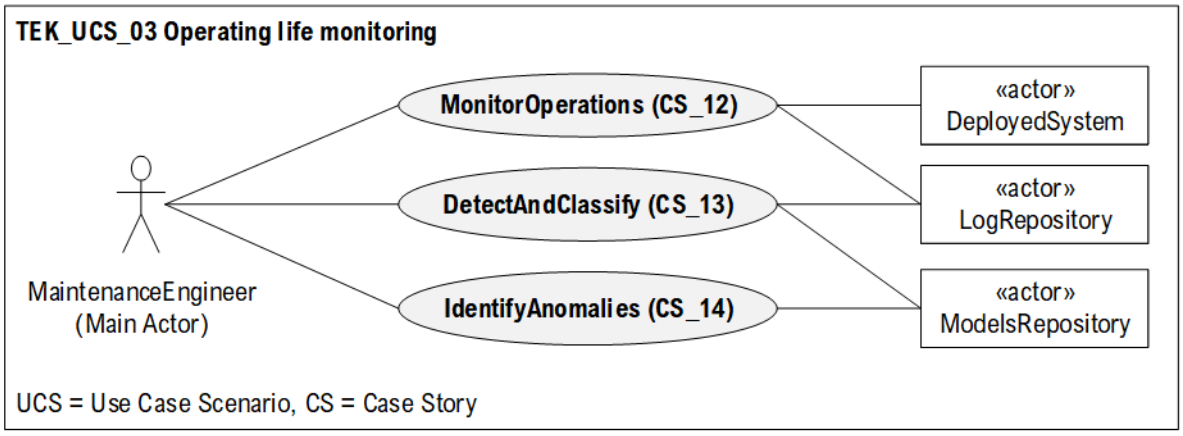


Figure 31 Use case scenario TEK_UCS_03 "Operating life monitoring"

- **CS_12** collects data from the deployed system that operates in the real environment and provides its services.
- **CS_13** analyses the collected data, for detecting anomalies that already occurred and persist, as well as anomalies that probably will occur. After a detection it classifies the anomaly and raises an alarm.
- **CS_14** identifies the anomaly (causes analysis) after it has been detected and classified, as well as gives support to the maintenance engineer who decides about the maintenance intervention.

AI/ML technologies are supposed to underlie the functionalities of the demonstrator components required in TEK_UCS_03 (detection, classification, and identification), as well as the complex capabilities of the development tools required in TEK_UCS_01 and TEK_UCS_02.

3.13.4 Relevant KPIs and definition of success

The previous section "Expected improvements in AIDOaRt" depicts how TEK experiments the AIDOaRt Framework functionalities: these are employed for carrying out three groups of development activities (ultimately three use case scenarios) *to evaluate* how they contribute to achieve the goals that TEK considers in AIDOaRt:

- TEK_Goal_01: support to the design-time test.
- TEK_Goal_02: run-time test automation.
- TEK_Goal_03: AI/ML based components.

Three KPI's, which specialize the AIDOaRt project level KPI's, allow *to quantify* the achievement of each goal at the end of the project. Two of them measure the efforts spent in the design verification (TEK_Goal_01) and in the run-time test (TEK_Goal_02) that are expected to decrease. The third one measures the performance of the AI/ML based components that is expected to increase.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
TEK_UCS_01	Design choices verification — Before the implementation; on models and/or with rapid prototyping; environment: simulated/emulated.
TEK_UCS_02	Run time verification — On the implemented module/component/sub-system/system; environment: high fidelity near real, real.
TEK_UCS_03	Operating life monitoring — On the product that carries out its services; environment: operating.

Table 29 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
TEK_UCS_01_KPI_1.1_1	TEK_UCS_01	KPI_1.1	Reduction of effort for design verification	0%	25%
TEK_UCS_02_KPI_3.1_1	TEK_UCS_02	KPI_3.1	Reduction of testing effort	0%	30%
TEK_UCS_03_KPI_3.1_1	TEK_UCS_03	KPI_3.1	Improvement of capability of detecting and classifying defects during the operating life	0%	30%

Table 30 Case Study KPIs

3.13.5 Validation methodology and process

Plan — The verification is quantitative: measurement of each KPI and computation of the improvement as percentage of the baseline value.

Moreover, the assessment of the project outcome can be completed by a qualitative validation of certain aspects of the tools—such as training, usability, integration in the current industrial development cycle—that, through a discrete scale of values, the technicians express based on their experience.

Procedures — Table 31 describes the proposed procedures.

KPI	Proposed procedure
TEK_UCS_01_KPI_1.1_1: Effort for design verification.	Measurement of the time taken for verifying the design. Baseline value: time taken by manual execution.

TEK_UCS_02_KPI_3.1_1: Testing effort.	Measurement of the time taken for designing and preparing the tests, and for interpretating the results thereof. Baseline value: time taken by manual execution.
TEK_UCS_03_KPI_3.1_1: Capability of detecting and classifying defects during the operating life.	Creation of a test environment, emulation (injection) of defects. Figures of merit of the detection and classification functionalities are measured.

Table 31 Proposed procedures for measuring the improvement of the KPI's

The procedures can be refined during the project prosecution, because the tools coverage may not include some parts, e.g. of some system component or of some sub-phase of the development.

3.13.6 Demonstrator setup and testbed

TEK_UCS_01_KPI_1.1_1 "Design choices verification" ↔ Use case scenario TEK_UCS_01

The requirements are defined, then there is the model-based design and, at the end, the design verification that consider if the models are consistent regarding the requirements and if they "work". The time to carry out this last activity is measured.

According to the procedure for assessing the KPI improvement, the time taken by the manual execution of the activity above is compared to the time taken by the same activity when this uses the Framework. A weight can be introduced in the comparison, in the case the two test sets have different coverage.

TEK_UCS_02_KPI_3.1_1 "Run-time verification" ↔ Use case scenario TEK_UCS_02

For selected components of the demonstrator, after these have been developed, the tests are designed and prepared. The total time to carry out these activities is measured. The preparation time includes time required for special testing software but not for physical preparation; test execution time is excluded at all; time for interpreting the results of the tests can be added to the total. Test design and results interpretation are based on requirements and on design models and can reuse the latter.

The same considerations as in the use case scenario TEK_UCS_01 case apply to the procedure for assessing the KPI improvement.

TEK_UCS_03_KPI_3.1_1 "Operating life monitoring" ↔ Use case scenario TEK_UCS_03

The testbed includes the hardware and the software of the demonstrator of the Prognostics and Health Management system. The power electronics which the system acts on is in part real, e.g. battery, converter, and motor, and in part emulated, as well as the environment. The AI based component of the system monitors the power electronics and analyses the collected data, for detecting anomalies that already occurred and persist, as well as anomalies that probably will occur. Faults are injected, or condition for which faults will happen are created. The figures of merit of detection and classification functionalities are measured.

3.14 Case Study Volvo CE - VCE_MDE (VCE)

3.14.1 Case study overview

Volvo CE is undergoing transformation in its product development context from traditional complex mechanical systems with embedded software towards more advanced software and technology intensive units (component systems) integrated within larger SoS (Systems of Systems). For a given constituent system, there are multiple development cycles, e.g. product, platform, application, service developments form longer to shorter durations. That is, Volvo CE is looking forward to using model-based frameworks, methods, and tools to support the above developments cycles within a common unified framework. The VCE use case involves customizing standards, model-based frameworks and languages, e.g. OMG standard (UAF, SysML, etc) to model systems, software, information architectures. Additionally, the VCE use case involves integrating AI based data analytics to enhance configurations of software, as well as support overall architecture decision process towards a model-based DevOps framework.

3.14.2 Current way of working and baseline technology

VCE is a company that works with large scale product lines and has a large variety in the products. To enable working on such a large scale an extensive workflow is require, which is described in Figure 32.

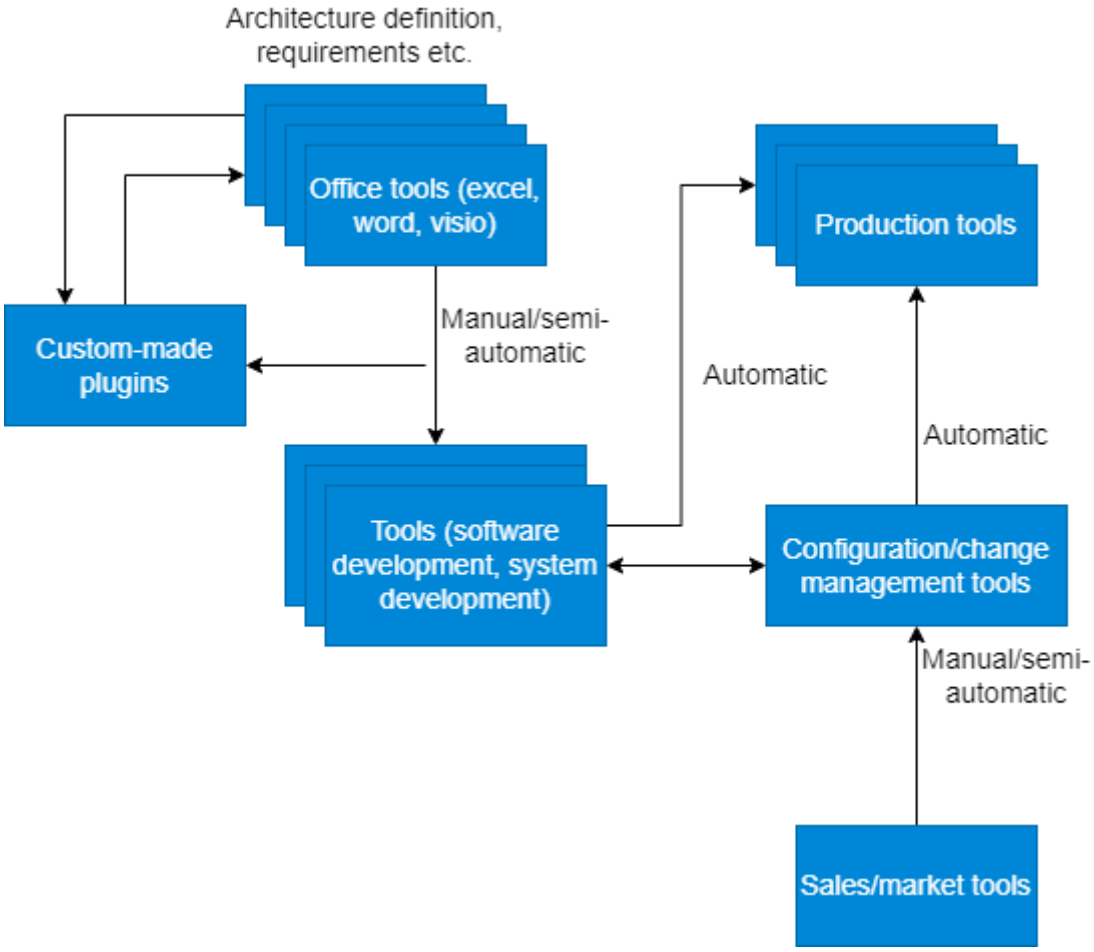


Figure 32 – Current VCE architecture

In the current way of working the baseline relies on methods and tools that are heavily reliant on documents and various non-model-based tools. The baseline way of working limits the understandability of the products being developed. This in turn limits the amount of re-use that can be done with the current artefacts, making the development less efficient.

With AIDOaRt the aim is to move towards a model-based framework to facilitate modularity, re-useability, scalability, and understandability. Due to the heavy integration with legacy tools and third-party tools there is a large importance on interoperability as well. In a similar sense there is a need for automation, where DevOps and AI principles, methods, and techniques will be evaluated and applied to improve the various workflows and pipelines.

3.14.3 Expected improvements in AIDOaRt

Previously the current baseline of VCE was described. The expected results of AIDOaRt aims to improve the baseline and start the transition towards the architecture seen in Figure 33.

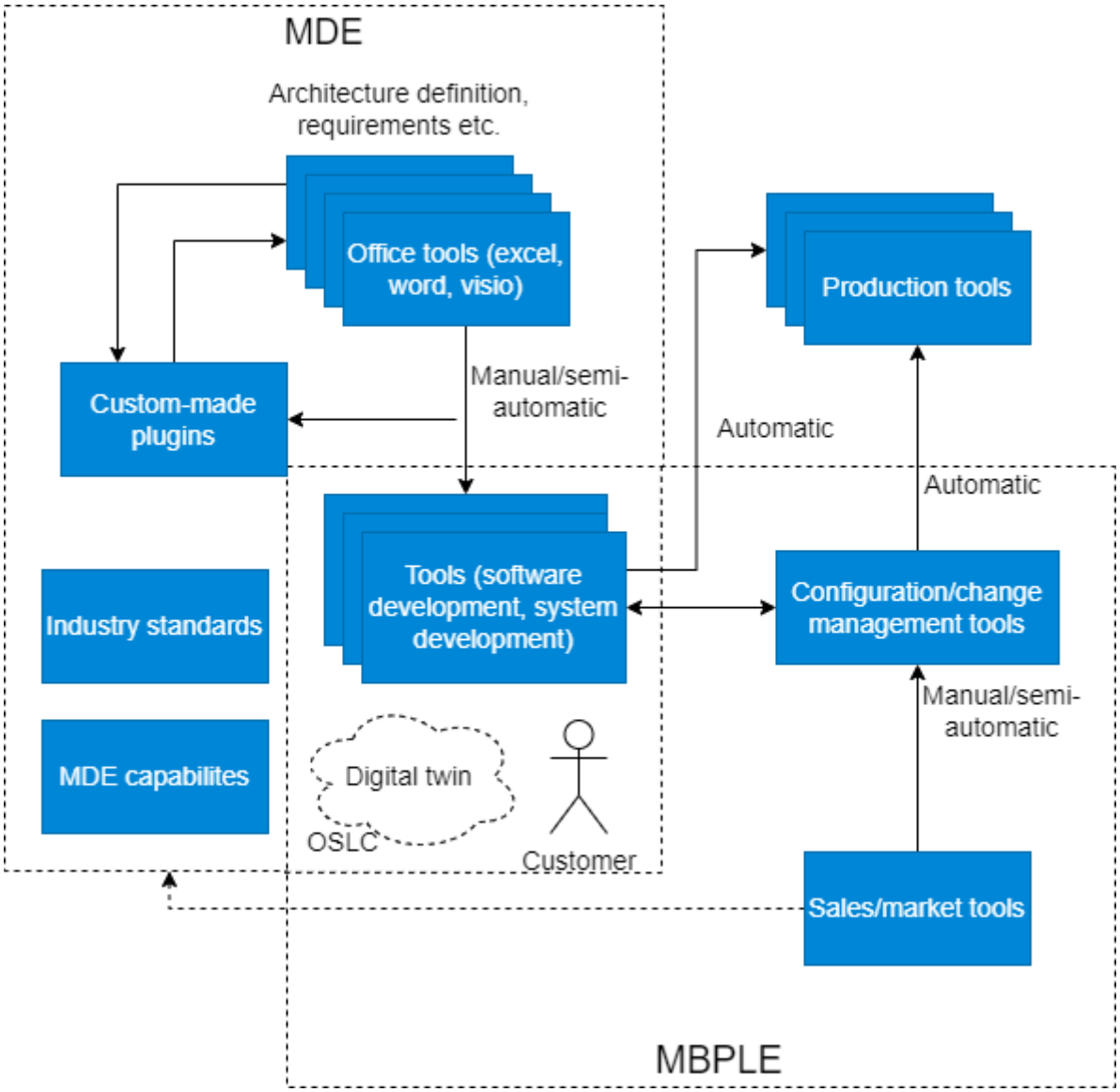


Figure 33 target architecture for VCE

The envisioned architecture is an extension of the baseline seen in the previous section. Specifically, the aim is to introduce MDE principles and capabilities into the current baseline. The use case scenarios VCE_UCS_1 and VCE_UCS_2 describe the needs of VCE to integrate MDE to support the architecture modelling activities and the corresponding V&V. To support MDE, VCE aims to integrate industrial standards, such as OSLC, to facilitate interoperability and integration with various tools, both legacy and new. In addition, VCE_UCS_3 describes the need of VCE to introduce AI and DevOps tools, methodologies, and techniques. This is realized via the digital twin approach, situated in both the MDE and MBPLE contexts.

By introducing the methods and tools developed in the AIDOaRt framework in the VCE context and evaluating the use case scenarios the improvements are expected to several. Increasing re-use of architecture descriptions via models is expected and a key aspect of the model-based approach. Further the use of architecture models currently at VCE are descriptive, utilizing MDE the aim is to

create analytical architecture models. However, a key part of MDE is to further increase the descriptive capabilities of the architecture models, increasing the understandability. From the product line perspective, the use of MBPLE will be used to improve capabilities related to variability. Overall, the workflow at VCE could be improved regarding automation, where AI and DevOps methods are a potential solution. Additionally, the digital twin principles will be evaluated and applied to improve analysis in the production pipeline.

3.14.4 Relevant KPIs and definition of success

VCE has three use case scenarios in the AIDOaRt project. For each scenario the following rationale has been applied when determining the KPI's and relation to AIDOaRt KPI's:

Use case Scenario 1 (VCE_UCS_01)

- The current process needs to be improved and extended via MDE capabilities
- VCE requires patterns and methods to facilitate re-use of architecture models
- Currently many activities are manual which should be automatic
- Integration with standards and third part tools are required

Use case Scenario 2 (VCE_UCS_02)

- The engineer should be able to validate and verify high level models towards architecture requirements and consistency
- The validation and verification process should be automated wherever possible
- The validation and verification criteria should be customizable

Use case Scenario 3 (VCE_UCS_03)

- VCE requires AI capabilities
- Automation with AI methods should improve current processes
- DevOps and AI capabilities are a key part of automation of current manual tasks

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
VCE_UCS_01	Modeling system, software, data architectures
VCE_UCS_02	Validation and verification of architecture models.
VCE_UCS_03	AI augmented DevOps workflow and data analytics

Table 32 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
VCE_UCS_1_KPI_1.1_1	VCE_UCS_01	KPI_1.1	Increase in development velocity, utilizing MDE	0%	10%
VCE_UCS_1_KPI_2.2_1	VCE_UCS_01	KPI_2.2	Re-use of architectural models in future projects	0%	20%
VCE_UCS_1_KPI_3.1_1	VCE_UCS_01	KPI_3.1	Automation of modeling activities	0%	20%
VCE_UCS_2_KPI_1.2_1	VCE_UCS_02	KPI_1.2	Inconsistency detection in models	0%	10%
VCE_UCS_2_KPI_3.1_1	VCE_UCS_02	KPI_3.1	Automate V&V on architecture models	0%	20%
VCE_UCS_3_KPI_3.1_1	VCE_UCS_03	KPI_3.1	Automate manual processes	0%	20%
VCE_UCS_3_KPI_3.2_1	VCE_UCS_03	KPI_3.2	Improve coverage of data gathering	0%	20%

Table 33 Case Study KPIs

3.14.5 Validation methodology and process

The validation of the defined KPI's of VCE will be evaluated internally on VCE processes and projects currently ongoing. Since VCE works with large product lines maintaining variability the different KPI's will be evaluated by introducing the AIDOaRt principles and methods in projects utilizing the previous workflows and methods. More concretely the use-case scenarios VCE_UCS_1 and VCE_UCS_2 and the corresponding KPI's will be evaluated via creating patterns and methods of modelling VCE artifacts and internal components. As such the activities currently performed will be migrated into the AIDOaRt framework and performed by engineers at VCE. The different KPI's will be measured and compared to the baseline processes and interviews with engineers will provide additional evidence to support the findings and evaluate the KPI success and viability of the proposed framework and methods. For VCE_UCS_3 the evaluation will consist of mapping the extent of automation and increase in potential data coverage of VCE projects where the AIDOaRt framework is applied.

3.14.6 Demonstrator setup and testbed

VCE will evaluate the KPI's in the AIDOaRt project on the architecture models used to describe the VCE systems, such as haulers. The systems will be described using SysML from OMG and follow industrial standards such as OLSC. Existing means of measuring and evaluating measurements will be used for the KPI evaluation to have a uniform structure of the data gathered that will be compared.

3.15 Case Study Westermo - Embedded systems for data communication

3.15.1 Case study overview

Westermo develops embedded systems for robust industrial communication applications such as onboard rail, track-side rail, power distribution etc. The software in these devices is developed in an agile feature-driven process with high demands on quality, and over the years Westermo has invested heavily in their system-level automated testing. In the AIDOaRt project Westermo desires to improve all aspects of the development and continuous integration and to (i) increase the flow in the development process (e.g. by automating steps in the process), as well as (ii) increase the product quality (e.g. by improving the quality processes). The "user" could be (a) the development teams during development, and (b) project management that inspects if quality is in shape for release. (Westermo has limited experience of AI, and almost all flavors of AI could be considered in the AIDOaRt project.)

3.15.2 Current way of working and baseline technology

The current flow at Westermo is an automated build system with nightly testing. First, humans (A) locally alter code, or create other artifacts. Then, when they submit code changes (B), the source code is analyzed with static code analysis and compiled in a series of steps (C). Intermediate results are stored in many individual subsystems that typically have a storage (say database or log file repository), a backend and a frontend (D). After passing through these steps new software has been compiled (E) that is an important part of the embedded systems (F). Each night, many test cases are executed on test systems buildup of physical devices or virtualized hardware. The results are stored in a sub-system and can be thought of as having a storage, backend, and frontend.

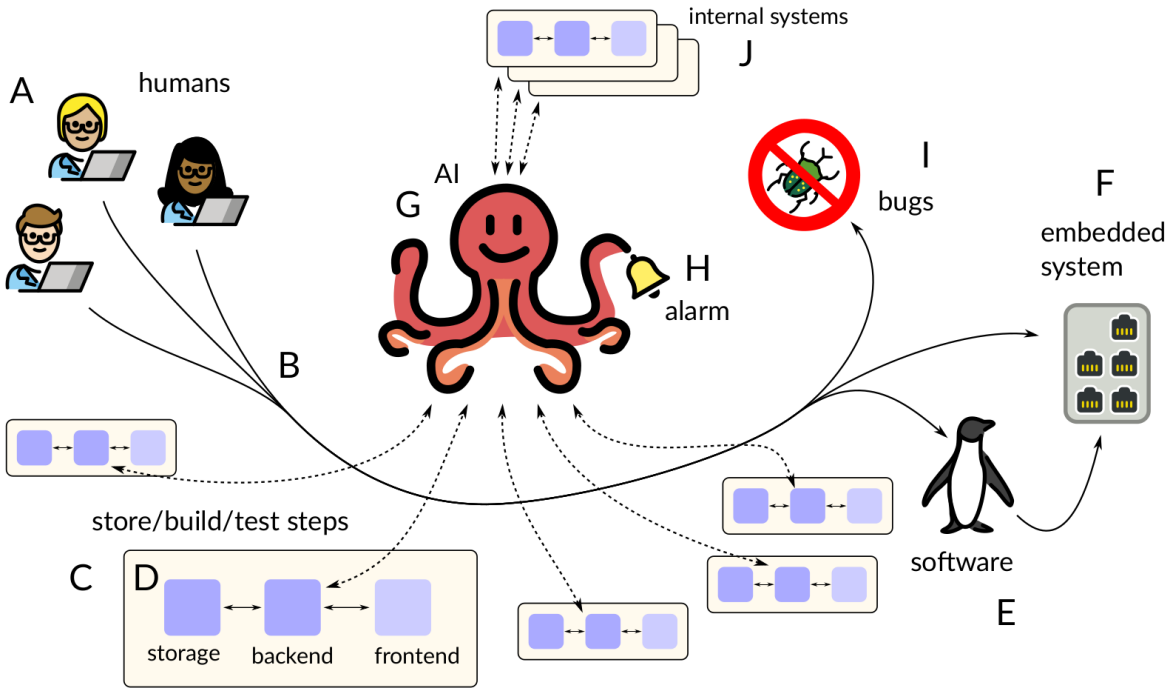


Figure 34 Simplified Architecture of the Westermo Use Case

The goal in AIDOaRt is to, on top of, or parallel to, the existing flow, add one or several AI's (G) that analyze data, raises alarms is flaws are identified (H) such that issues can be eliminated (I) or that helps humans to find the issues faster. It is likely that this AI would need internal (J) systems for internal representation of data -- these internal systems are likely of a similar structure as the ones already in place.

In particular, the current baseline has very limited supervision, anomaly detection, and sometimes overwhelms staff with information such as log data.

3.15.3 Expected improvements in AIDOaRt

As previously described, Westermo has three case stories in AIDOaRt. First, Westermo expects automated continuous decision making and increased flow in the development process (W_CS_1). Second, Westermo expects prediction/monitoring of reliability or root causes of failures on development artifacts (W_CS_2). Third, Westermo expects prediction/monitoring of non-functional quality attributes of the products (W_CS_3).

In practice, this could mean that some AI system or systems (G in the above figure), would have data available for a holistic integration and analysis, or more concretely: (i) at least twice as many data sources are available for analysis by an AI as before AIDOaRt, and (ii) these data sources are used with already collected data sources for analyses. This obviously enables an AI to (iii) find unknown

dependencies between data sources, such as (iii.a) identifying triggers in one data source, that leads to (iii.b) observable consequences in other data sources.

Another track in which AIDOaRt is expected to bring improvements, is for the AI to (iv) monitor functional or quality characteristics (e.g. tests that pass or fail, or memory usage) and warn when deviations are found. If an AI can learn to monitor, then with enough data, perhaps (v) root causes to these issues can be identified (e.g. test case Z fails because of previous activities X and Y). Finally, if root causes can be identified, then (vi) predictions could perhaps be taught (e.g. by observing activity X and Y, an AI estimates a 75% likelihood for failure Z).

3.15.4 Relevant KPIs and definition of success

For Westermo, the most important aspect of AIDOaRt is the expected improvement in terms of human performance. Westermo expects staff to find defects earlier, to mitigate problems faster, and to get support in monitoring and observing phenomena in DevOps. In order to support these improvements, enhancements on data collection and collaborations within AIDOaRt are important.

KPIs of particular importance are KPIs 1.1 on improvements in early detection and reduction in time needed for mitigating problems, KPI 2.2 on increases in data collection, and KPI 3.1 on increased automation and actionable feedback.

In the following table, we list the use case scenarios of the case study.

Use Case Scenario ID	Complete Name of Use Case Scenario
W_USC_1	AI-Augmented devops development process
W_USC_2	AI-powered root cause analysis w.r.t. functional issues
W_USC_3	AI-powered root cause analysis w.r.t. non-functional quality
W_USC_4	AI-powered log analysis/root cause analysis

Table 34 Recap of the Use Case Scenarios

In the following table, we list the KPIs that are relevant to the case study. For each use case scenario of the case study, we specify which project KPIs are relevant, what they mean in the context of the use case scenario, and what is the current baseline value and the expected target value for each KPI by the end of the project.

Case Study KPI ID	Use Case Scenario ID	Project KPI	KPI Instantiation and Definition	Base Value	Target Value
W_UCS_1_KPI_2.2_1	W_USC_1	KPI_2.2	Increase of 20% in the number of available data sources for enabling AI-Powered monitoring, root causing, prediction, etc.	0%	20%

W_UCS_2_KPI_1.2_1	W_USC_2	KPI_1.2	Improvement by 20% the time needed to root cause functional issues.	0%	20%
W_UCS_3_KPI_1.2_1	W_USC_3	KPI_1.2	Reduction of 33% of the person time needed to be invested in root cause investigations of non-functional quality issues.	0%	33%
W_UCS_4_KPI_3.1_1	W_USC_4	KPI_3.1	Implementation of useful log clustering for 20% of available log types.	0%	20%

Table 35 Case Study KPIs

3.15.5 Validation methodology and process

Some improvements can be expected to be almost purely quantitative such as increases in the number of data sources collected, from a certain number to an increased number. Some can be derived from existing or planned data collection in nightly testing if combined with human judgment, knowledge or understanding, such as hours of nightly testing lost due to poor maintenance (a number expected to be reduced). As mentioned above, the most important aspect of AIDOaRt is the improvements in human performance, which motivates involving humans in the validation. To summarize, Westermo expects to validate the improvements from AIDOaRt quantitatively, by analyzing collected data, and with focus group and/or reference group meetings which involve staff working with analyzing artifacts produced in the DevOps process, and/or managers of such staff.

3.15.6 Demonstrator setup and testbed

Three important tools already in use at Westermo are (i) an internal GitLab instance with support systems, (ii) a tool for exploring and visualizing test results and logs called Tim[1], as well as (iii) a set of other standard open source tools and services for managing DevOps etc. Any improvements coming from AIDOaRt are to be expected to be integrated or evaluated with GitLab, Tim or other related tools.

One way to evaluate, e.g. log clustering (W_USC_4), would be to set up an isolated clone of the environment needed for running a Tim instance on a laptop, copy a subset of the available data onto that laptop, and use this isolated environment for exploration, prototyping and evaluation. In fact, this approach has been done in a final thesis during the fall of 2021 and is being done again for another thesis project during the spring of 2022.

[1] See Strandberg, Afzal, & Sundmark. Software test results exploration and visualization with continuous integration and nightly testing. *Int J Softw Tools Technol Transfer* (2022). doi.org/10.1007/s10009-022-00647-1

4 Conclusion

This deliverable is the first output from Task 5.3 in WP5 regarding the evaluation of AIDOaRt technologies in the context of each industrial case study of the project. Among other things, the main focus in D5.5 has been to capture the details of: case study baseline technologies and state at the start of the project; expected improvements from the application of AIDOaRt technologies and solutions; case study KPIs and metrics to measure and determine the level of improvements and success in solving the challenges of case studies; and description of case study testbeds and demonstrators for evaluation of project solutions, along with the validation methodology and evaluation process. Deliverables 5.7 and 5.9 as follow-ups of D5.5 will capture detailed evaluation results and report on concrete measurements based on the set of KPIs that are defined in this deliverable for each case study and use-case scenario.