AIDOaRt

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

D5.2 - AIDOaRt Integrated Framework - Initial Version

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	Month 21 (December 2022)
Author(s):	Raúl Santos de la Cámara (HIB), Inmaculada Luengo (HIB) Internal reviewers: Claudia Keinrath (AVL), Smrutirekha Sahoo (Alstom) External reviewers: Benjamin von Berg (TUG), Bernhard Aichernig (TUG), Andrea Pferscher (TUG), Johan Bergelin (MDU/VCE)
Partners contributed:	BT, MDH, AIT, AST, AVL, DT, TUG, JKU Linz, CAMEA, BUT, ABO, AND, QEN, CSY, IMTA, PIO, ABI, INT, ROTECH, TEK, UNIVAQ, UNISS, ACO, UOC, HIB, ITI, PRO, UCAN, VCE, RISE, WESTMO
Date:	21 December 2022
Version	1.00
Revision:	01
Abstract:	This document presents the initial status of integration in the AIDOaRt project. It implements the methodology put forward in D5.1 in which integration is detailed across different axes and according to several aspects of interest. This is cross-related with the modelling approach followed in other areas of the project to generate a comprehensive integration approach that will be completed in deliverables D5.3 and D5.4.
Status:	Release candidate v1.0

DOCUMENT REVISION LOG

VERSION	REVISION	DATE	DESCRIPTION	AUTHOR
V0.1	01	01 Jul 2022	Table of contents	Raúl Santos de la Cámara (HIB)
V0.2	01	03 Aug 2022	Changes to TOC	Raúl Santos de la Cámara (HIB), Claudia Keinrath (AVL)
V0.3	01	23 Aug 2022	Changes to TOC, definition of fist microtasks	Raúl Santos de la Cámara (HIB)
V0.4	01	30 Aug 2022	Final ToC	Raúl Santos de la Cámara (HIB)
V0.55	01	13 Oct 2022	Updates to section 3	All authors
V0.8	01	28 Nov 2022	Draft for review	Raúl Santos de la Cámara (HIB)
V0.9	01	07 Dec 2022	Review updates to section 3, Introduction, Conclusions.	All authors
V0.96	01	14 Dec 2022	Draft for pre-final review.	Raúl Santos de la Cámara (HIB)
V0.97	01	16 Dec 2022	Draft after pre-final review.	Raúl Santos de la Cámara (HIB)
V0.98	01	19 Dec 2022	Integration of fixes following pre-final review. Consolidation of section order in Chapter 3	Raúl Santos de la Cámara (HIB)
V1.00	01	21 Dec 2022	Release candidate version	Raúl Santos de la Cámara (HIB)

Table of Contents

1 Introduction

This deliverable presents the first release of the AIDOaRt integration framework. Based on the methodology that emerged from Task 5.1 (see AIDOaRt deliverable D5.1[1], see description of the task in the starting paragraphs of Chapter 2), we outline the concrete integrations that have been made up to the publication of this document.

Here the focus was on establishing the so-called *confirmed* relations as explained in D5.1. These correspond to relations of two or more elements that require an integration to work together in the context of an AIDOaRt framework/platform development. They are confirmed in that they have been explicitly identified by their owners as related and mature for integration. This is in contrast with so-called *potential* relations (see D5.1.), whose elements do not correspond to explicit relationships, but can be derived from the mutual relationships expressed in the AIDOaRt Modelio model environment. For example, a solution and a use case that share a common AIDOaRt Generic Requirement are potentially suitable for integration, even if this wasn't explicitly signaled by their owners.

This is considered one of the benefits of the modeling strategy followed under the AIDOaRt framework: for large inventories of solutions, requirements and use cases, some potential integrations might be overlooked even by the very owners of the elements. By using the *Integration Mediator Pattern* described in D5.1, the achieved benefit is that all of the available relationships can be exhaustively analyzed and examined to achieve the best possible integration.

When applying the above-mentioned pattern, some *Integration Aspects* are defined that correspond to facets of the integration process that can be documented in detail. This makes the process traceable and the technical aspects of the integration well defined. In this document, we outline a detailed analysis of the 21 confirmed relations outlined earlier. Further on, the current integration status of those collaborations is presented. The work will be continued in future deliverables of task T5.1 such as deliverable *D5.3 AIDOaRt Integrated framework – intermediate vers.(M24)* and *D5.4 AIDOaRt Integrated framework – final vers.(M36)*. These documents will further detail the integration aspects, expand on subsequent confirmed collaborations, and project results.

[1] Throughout this document the following abbreviations for project items are used:

- WPX → AIDOaRt work package X,
- TX.Y → project task Y corresponding to work package X,
- DX.Y → project deliverable Y corresponding to work package X.

1.1 Scope and motivation

The purpose of this document is to present the status of integration aspects in the AIDOaRt framework at the mid-time of the project. The overall integration is currently work-in-progress: most of the required building blocks from the use cases and the proposed work methodology for AIDOaRt are in place, but integrations are complete to varying degrees. Therefore, this document briefly presents the underlying integration methodology (Chapter 2), followed by a snapshot of current integration work, divided into 21 active collaborations. This will allow insights into the benefits of the AIDOaRt integration methodology, which proposes core elements of an all-encompassing methodology to produce CPS solutions using AI, MDE and DevOps techniques.

1.2 Relations to other deliverables

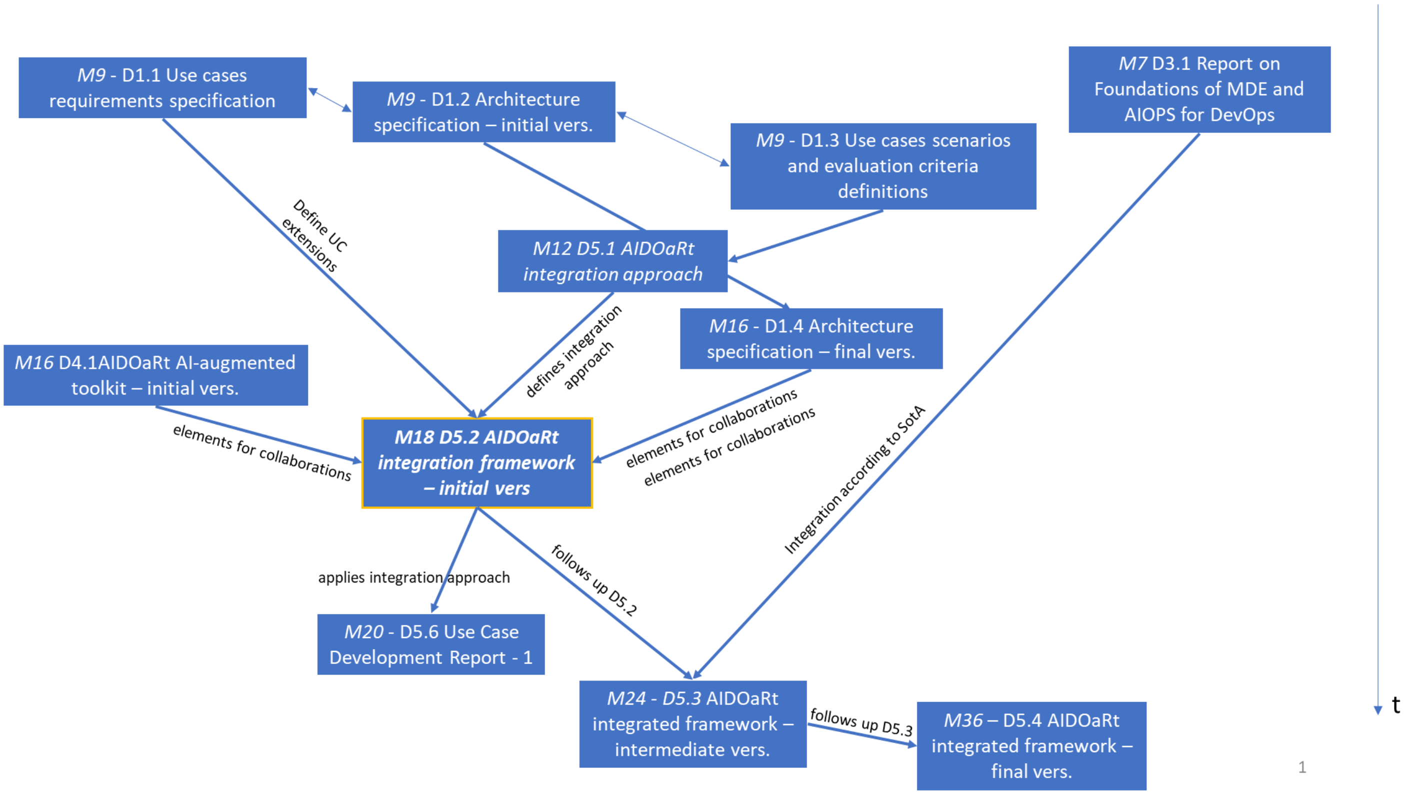

M9 - D1.1 Use cases requirements specification
M9 - D1.2 Architecture specification – initial vers.
M7 D3.1 Report on Foundations of MDE and AIOPS for DevOps
M9 - D1.3 Use cases scenarios and evaluation criteria definitions
Define UC extensions
M12 D5.1 AIDOaRt integration approach
M16 - D1.4 Architecture specification – final vers.
M16 D4.1AIDOaRt AI-augmented toolkit – initial vers.
defines integration approach
M18 D5.2 AIDOaRt integration framework – initial vers
elements for collaborations
elements for collaborations
elements for collaborations
Integration according to SotA
applies integration approach
follows up D5.2
M20 - D5.6 Use Case Development Report - 1
M24 - D5.3 AIDOaRt integrated framework – intermediate vers.
follows up D5.3
M36 – D5.4 AIDOaRt integrated framework – final vers.
t
1

Figure 1 Relationships of D5.2 with other deliverables

Figure 1 gives a pictorial summary of how D5.2 relates to other deliverables in the project. Given that integration work is presented in this report, it is indirectly related to most of the work in the project, however, closer relationships can be identified with the following deliverables:

- From the initial round of descriptive deliverables for the project (such as *D1.1 Use cases requirements specification*, *D1.2 Architecture Specification – Initial version* and its follow-up *D1.4 Architecture specification – final version* and *D1.3 Use Cases Scenarios and evaluation criteria specifications*) the basic elements were obtained that make up the AIDOaRt integration framework. This corresponds, on one side, to the Use Cases and their context (i.e., the results to be used in real world scenarios that integrate AIDOaRt technology) and the basic framework upon which AIDOaRt operates, with a primary list of components of the framework, solutions that conform to the framework and which are applicable to solve needs of the use cases, and an overarching architecture that enables all of them to work together. From the work in WP3 we also obtained a thorough understanding of the current state of the art for the related fields of study for AIDOaRt (e.g., MDE, CPS, DevOps and AI), which we defined in *D3.1 Report on Foundations for MDE and AIOps for DevOps*. Thus, these are the fundamental ingredients of the integrated framework.
- On top of these initial results, which outlined the basics of the AIDOaRt mode of operation, we were able to build more specific results that are AIDOaRt specific, and which can help us specify the integrated framework. These include diverse deliverables such as D4.1 AIDOaRt AI-augmented toolkit – initial vers. and D5.1 AIDOaRt Integration Approach, which further outline the different building blocks in the AIDOaRt toolbox (in D4.1) and a method proposal for integrating elements in AIDOaRt (in D5.1). These second-stage results form the basis of the work that we are presenting in D5.2, in which we document the integration process by implementing the D5.1 methodology on top of the tools and other results presented in deliverable D4.1.
- Finally, work on the integration in the AIDOaRt framework does not end with this document. Following this, the *D5.3 AIDOaRt integrated framework – intermediate version* document will capture more integration descriptions later on in the project. Furthermore, these integrated elements will be used for the production or development of the test pieces for evaluation in *D5.6 Use Case Development report*. In D5.3, it is aimed to use the AIDOaRt methodology (described in D5.1) to its full extent, the benefits of which upon the production of use case implementations will be shown in D5.6.

1.3 Terms used in this document

In creating the methodology outlined in D5.1 and other areas of the project that are related to this deliverable, a number of terms have been used to signify concrete aspects of our work. In this section we therefore provide a short summary of the most relevant terms for a better understanding of D5.2.

- **Use Case Provider** – partner, which provides a *Use Case Scenario* related to a dedicated end product or service in form of a *Case Study*
- **Case Study** – is about the intended end product in the form of a demonstrator and/or prototype of a system

- **Case Story** – describes what the product should offer the end user as concrete product features
- **Use Case Scenario** – describes *which* requested/required AI-augmented technical solution/approach is intended to be applied within a concrete Use Case to realize the related Case Story
- **Use Case Requirements** – describes *what must be fulfilled* to enable the Use Case Scenario in the context of the intended AI-augmented technical solutions/approaches manifested by Component Capabilities (see below)
- **Use Case Data Requirements** – are derived requirements for each Use Case Scenario but is not focusing on the AI-augmented technical solutions/approaches as such, but on the involved data sets with which these solution/approaches must deal with in the context of AI-augmented topics. In this sense, Use Case Data Requirements are driven not only by the Use Case Providers, but also by the Solution Providers, since their tools must deal with the data related to the Use Cases.
- **Solution Provider** – a partner providing a corresponding solution for one or more *Use Case Providers* by a dedicated method, tool or algorithm manifested in concrete *Component Capabilities*.
- **Solution Components (Capabilities)** – are concrete tool features, methods or other solutions provided/developed by the Solution Providers. They should fulfill Use Case (Data) Requirements.
- **Generic Requirements** – based on the set of Use Case Requirements, Use Case-independent Generic Requirements have been derived and potentially assigned to dedicated Solutions and Component Capabilities
- **AIDOaRt Integration Approach** – the overall integration approach presented by this document, which is continuously evolving during T5.1 activities and is reported by a series of deliverables (D5.1-4).
- **AIDOaRt Integration Mediator Pattern** – a generic pattern-based approach, which is intended to detect potential and actual integration relationships between use case and solution providers. This was first explained in deliverable D5.1 and it is summarized in this document in Chapter 2.
- **AIDOaRt Integration Mediator Instance** – a concrete instance of the AIDOaRt Integration Mediator Pattern usually refers to a dedicated integration aspect and can be referred by use case and solution providers to derive potential indirect relationships. What we present in Chapter 3. are a number of AIDOaRt Integration instances that use the Integration Mediator pattern.
- **Integration Aspect** – a concrete topic for an AIDOaRt Integration Mediator Instance, which associates a (potential) relationship between use case and solution providers (or use case and solution aspect) with a specific content (e.g., generic requirement, applied state-of-the-art,

involved AIDOaRt architecture component, etc.). For all the 21 instances of integration presented in Chapter 3, description of many of the aspects are presented.

- **AIDOaRt Integration Mediator Process** – a dedicated process associated with the AIDOaRt Integration Mediator Pattern, which evaluates potential horizontal and vertical relationships between use case and solution providers to be confirmed or rejected. It comes along with a gap analysis for the rejected ones and an integration status for the accepted ones. The process was used to generate intermediate results used in the integration presented in Chapter 3. and to generate microtasks so that partners can work piecewise in this process.
- **AIDOaRt Integration Status** – refers to the confirmed (actual) relationships (horizontal and vertical) between use case and solution providers. It further refers to concrete involved components related to the corresponding integration aspect (e.g., relevant papers, AIDOaRt architecture elements or toolsets, etc.)
- **Integration Strategy** – concrete report how specific components and elements referred by the AIDOaRt integration status will be integrated- The different subsections of Chapter 3. Are good examples of particular integration strategies for the selected integrations.
- **Vertical Integration Relations** – integration relations relevant between use case and solution providers.
- **Horizontal Integration Relations** – integration relations relevant either between one or more use case providers or between one or more solution providers.
- **AIDOaRt Architecture** – overall architecture of the AIDOaRt project as elaborated by task T1.3 and reported by the deliverables D1.2 and D1.4
- **AIDOaRt Architecture Component** – concrete component defined by the AIDOaRt architecture reflecting a certain aspect of the AIDOaRt project
- **AIDOaRt Architecture Component Interface** – generic interface description of a concrete AIDOaRt architecture component.
- **AIDOaRt Modelio model-based approach** – model-based approach to define a project-wide single source of truth of the AIDOaRt architecture, use cases, solutions and the relationships among them.
- **Systematic Mapping Study (SMS)** – extensive literature study performed by T3.1 to find AIDOaRt relevant literature as basis for the aimed AIDOaRt goals and research activities. One of the aspects of the Integration-Mediator Pattern refers to how integration is related to the SMS.

1.4 Document structure

The organization of this document is as follows. After this initial **Introduction**, **Chapter 2** contains a summary of the methodology originally presented in the D5.1 (the *Integration-Mediator Pattern*) and its relationship with Task 5.1, presenting the initial version of integrations with respect to the AIDOaRt collaborations. These will be presented with the necessary context of the project's organization, concretely, the organization of periodic *hackathons* to boost the collaborations between the partners. These hackathons provide the unique opportunities of identifying *confirmed* collaborations (e.g., collaborations worked on during a hackathon are confirmed by definition), which have been chosen as a starting point for the analysis in this document. Furthermore, in **C**hapter 2, a detailed analysis on several integration aspects proposed to analyze generic collaborations is outlined. This is the basis for the method defined in deliverable D5.1 and the results so far are compiled in **Chapter 3** Integration Framework: Vertical Integrations. Finally, **Chapter 4** presents the conclusions of the work progress so far in the T5.1 and the potential next steps that will lead to the upcoming deliverables D5.3 and D5.4 as well as indirect continuations such as the Use Case integrations in deliverable D5.6.

2 Methodology summary and analysis of results

The description of the task T5.1 of AIDOaRt reads as follows:

> *[…] the main interaction between WP5 and WP2-WP4 will be performed and coordinated by this task. Appropriate use case-driven adaptations required for the AIDOaRt framework are evaluated and communicated on the one hand. On the other hand, the use case environment required for T5.2 and T5.3 is set up in a use case-specific manner.”*

Thus, the core goal of the task is to define ways of interaction and coordination between the different elements that have been developed in work packages WP2-WP4 in terms of integration between components and then to transfer these learnings to the use case specific applications.

For this to be applicable across AIDOaRt elements and beyond, this needs to be outlined at a high abstract level, that can then be applied in the project and eventually in other areas that need DevOps and AI inspired solutions for other domains.

This abstracted view was presented in deliverable *D5.1 AIDOaRt integration approach*, which was delivered in month M12 of the project. In that deliverable a thorough explanation of the methodology is presented. Said methodology is centered around what is called the Integration-Mediator Pattern and is used to document the integration results in this deliverable D5.2.

To get as much detail as possible, we encourage the reader to consult the full text of D5.1 as it provides many details and examples to follow the suggested process. In this chapter, we present a summarized view of this methodology that can be used as a guide to understand the subsequent analysis that is presented in Chapter 3.

We start to summarize this by analyzing some basic concepts. We define integration as the full set of extensions needed in two or more entities to collaborate on a given task. Integration can range from very trivial things such as formats used in collaboration tools to very advanced functional patterns required for successful coordination.

In the AIDOaRt perspective, we can see the project integrating use cases and solutions through several such discrete integrations (see Figure 2).

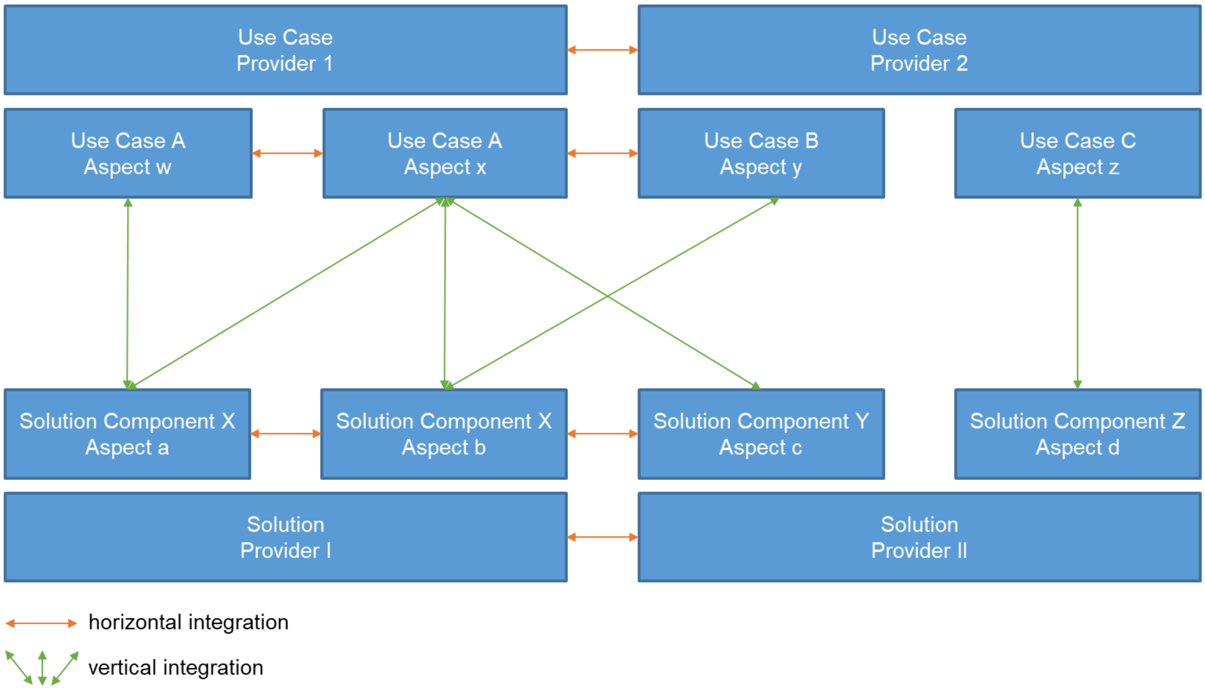


Figure 2 Vertical vs. horizontal integration aspects in AIDOaRt

For example, Use Case A, that features aspect x, is related to Solution Component X in two ways, through separate connections to aspects a and b. These connections are depicted as mostly vertical lines in the diagram (and using green arrows as connectors) and labeled as “Vertical Integrations”. Vertical integrations, are relationships between use cases and solution providers that are established through connections between aspects of use cases and components of a solution.

In contrast, some lines in Figure 2 are Horizontal, depicted as red arrows and connecting elements beyond the use cases and their immediate solution providers: for example, there could be a relationship between Use Case A, aspect x and Use Case B, aspect y. These are referred to as Horizontal Integration Relations.

Within the frame of AIDOaRt, Vertical Relations can be seen as required for a Use Case implementation to be integrated, while Horizontal Relations are those that, even if they are not required for the integration of the Use Case, enrich the overall project and have a certain degree of cohesiveness and cross-compatibility.

In real-world terms, the Vertical integrations are usually easier to find by mere inspection by the providers of Use Cases and Solutions, while for the horizontal a more systematic search process might be needed. Another dimension upon integrations are classified on is the status: some integrations are ‘Confirmed’ (i.e., existing and already being worked on by the respective owners) while some others are ‘Potential’ (i.e., detected manually or via automated processes as related, but maybe not being worked on at a given time).

Confirmed relations are rarely unclear to the owners of either use cases or solutions, since they are the very fabric of collaboration between partners. Potential relations are however almost always more subtle and difficult to detect. Usually this is done via manual analysis by domain experts such as the solution and use case developers. However, due to the MDE approach that we are following in AIDOaRt, other alternatives can be found.

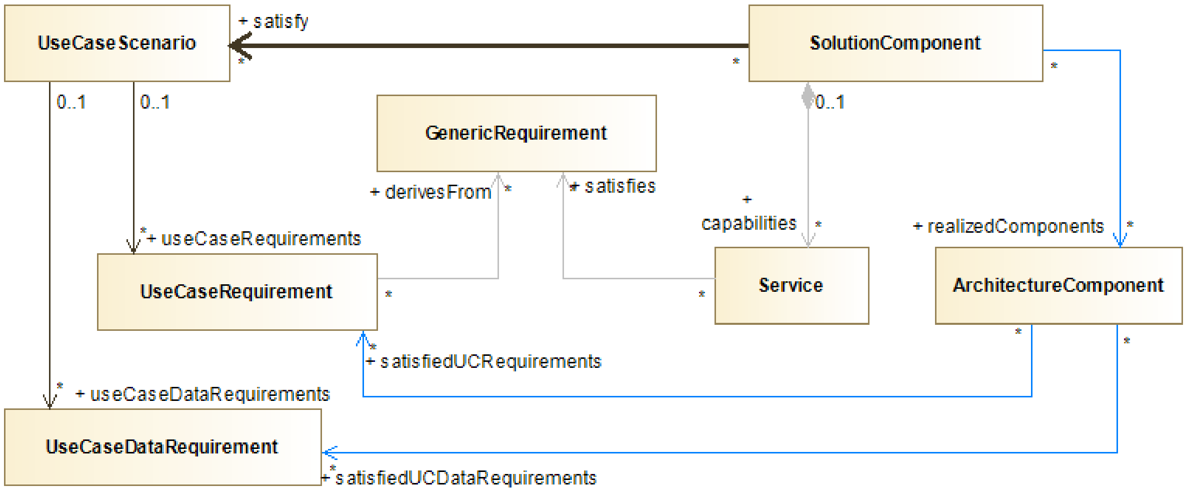


Figure 3 AIDOaRt metamodel and potential relationships

Figure 3 shows how, in the AIDOaRt metamodel, Solution Components *satisfy* the requirements of a Use Case Scenario. But this relation might not be one to one: if we analyze all the solutions in our model, we may find others that also fulfill the requirement by another means. This could for example be that both of these solutions fulfill the same Generic Requirement of AIDOaRt or maybe they address similar Architecture components (seen in the blue arrows that *indirectly* connect the solution component to the Use Case Scenario). This insight presents one of the main benefits of using a model-based approach and thus presents a wider array of options to complete this integration.

This is one of the core ideas that is used to analyze integration instances in this document. In the analyses of integration instances (see Chapter 3) it is also shown how in many of the uses cases there were solutions found to fulfil their needs that were apparent by analyzing the model (and then rating these potential connections to ensure that this is correctly identified). These ratings are one of the main outcomes in the deliverable as they signal a way forward for future integrations.

The other main element from D5.1 that is needed for an adequate understanding of this report is the discussion on Integration Aspects. Integration as a whole depends on identifying those aspects and providing analyses from both the provider of requirements (in our case the Use Case Provider) and the provider of solutions to these requirements (in our case the Solution Provider). This process can be summarized in Figure 4 that is the core and starting point of the AIDOaRt Integration Mediator Pattern.

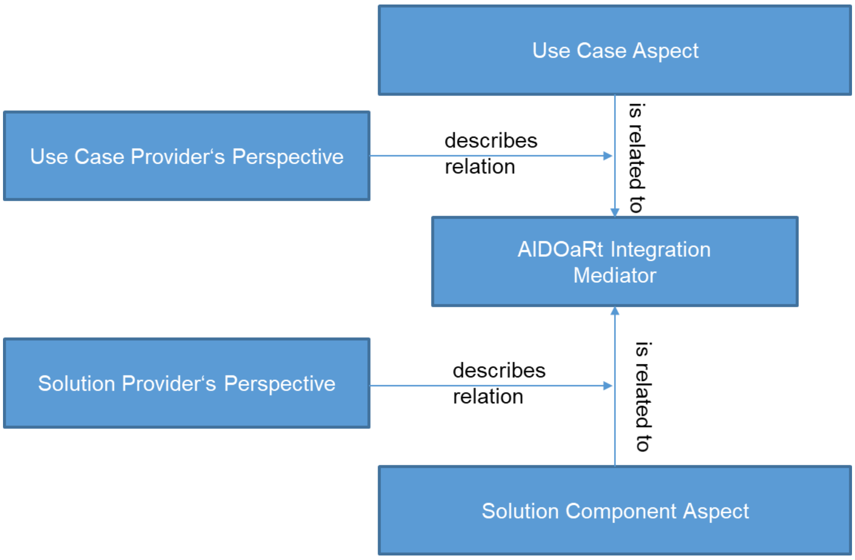

Use Case Aspect
Use Case Provider's Perspective
describes relation
is related to
AIDOaRt Integration Mediator
Solution Provider's Perspective
describes relation
is related to
Solution Component Aspect

Figure 4 The AIDOaRt integration mediator pattern

By analyzing the integration problem from both sides one can reach an optimum solution. For a detailed view on this, it is important to understand the integration aspects which are depicted in Figure 5.

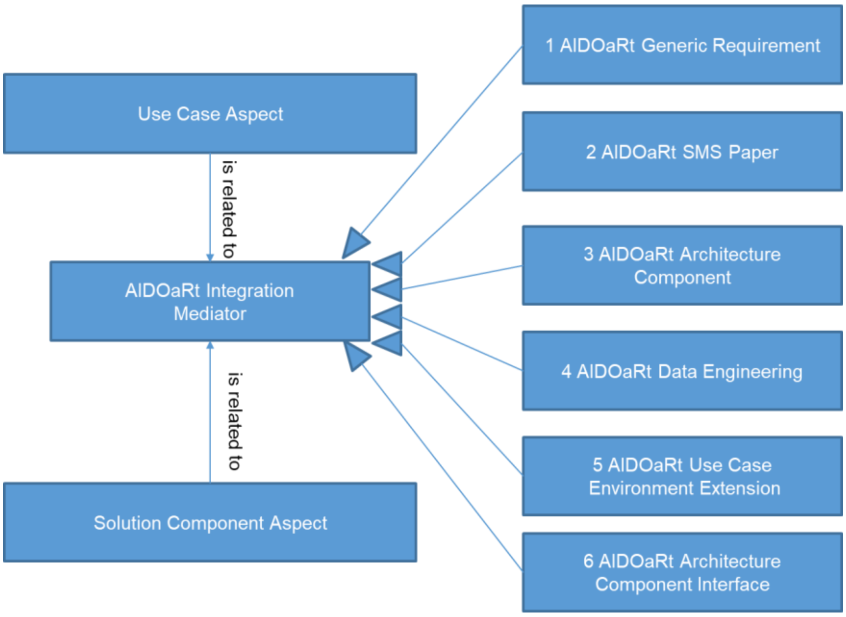

Use Case Aspect
is related to
AIDOaRt Integration Mediator
is related to
Solution Component Aspect
1 AIDOaRt Generic Requirement
2 AIDOaRt SMS Paper
3 AIDOaRt Architecture Component
4 AIDOaRt Data Engineering
5 AIDOaRt Use Case Environment Extension
6 AIDOaRt Architecture Component Interface

Figure 5 Concrete Integration Mediators

At their core, these aspects are different facets, fragments or sub-elements of the integration that are required for the overall integration. However, rather than leaving them to the particulars of each integration, in D5.1 a first taxonomy of such aspects was proposed, as depicted in Figure 5. This is not a closed list (for example, during the work for D5.2, the analysis of Cyber Physical Systems dimensions was added as aspects) but rather a useful summary of elements to consider.

Once we have identified the different components of this methodology, we only need to define a process that operates on them in order to generate the desired integration. This is depicted in Figure 6.

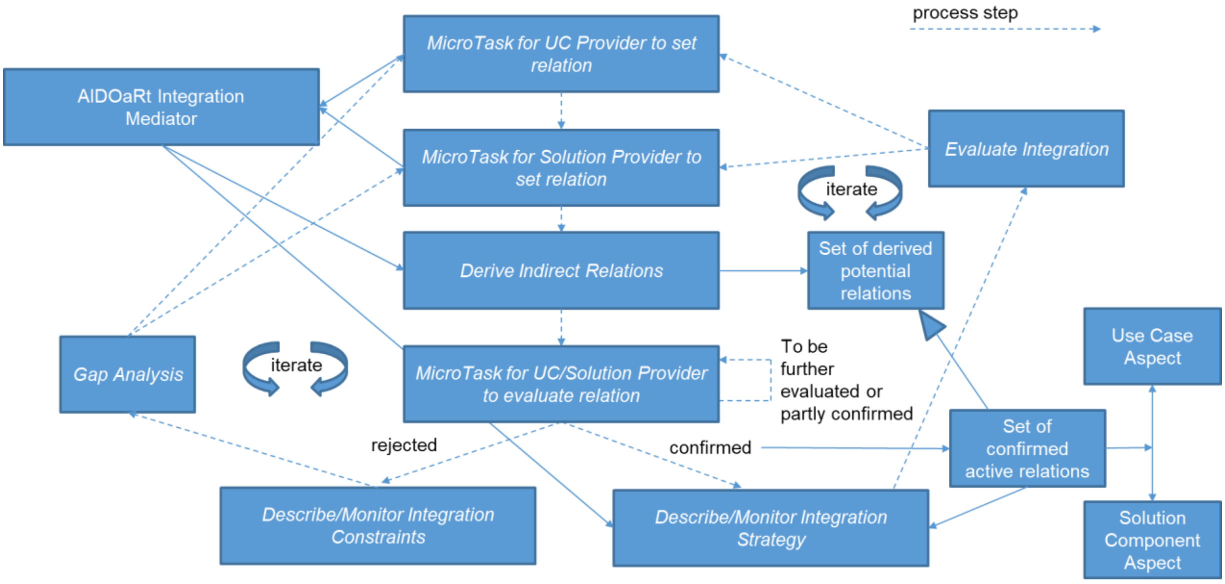
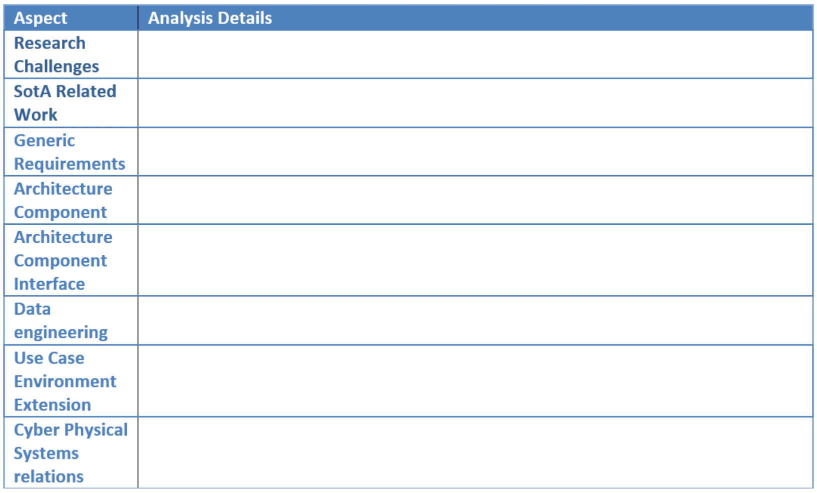

process step
MicroTask for UC Provider to set relation
AIDOaRt Integration Mediator
MicroTask for Solution Provider to set relation
Evaluate Integration
iterate
Derive Indirect Relations
Set of derived potential relations
Use Case Aspect
Gap Analysis
iterate
MicroTask for UC/Solution Provider to evaluate relation
To be further evaluated or partly confirmed
Set of confirmed active relations
rejected
confirmed
Describe/Monitor Integration Constraints
Describe/Monitor Integration Strategy
Solution Component Aspect

Figure 6 Mediator integration pattern process

In Figure 6, the process to extract integration data and document its progress is broken down as a succession of *micro tasks* (see D1.1 for more details) which are the essential units of work that we have been following in AIDOaRt. By following the stages and decision points depicted in the diagram (see D5.1 for a step-by-step description), we can generate the results of the integration and document them with a well-defined process.

This was the main process by which this deliverable was achieved. The collaborations in the project were subject of this analysis and what is delivered in Chapter 3 closely follows this:



Aspect	Analysis Details
Research Challenges	
SotA Related Work	
Generic Requirements	
Architecture Component	
Architecture Component Interface	
Data engineering	
Use Case Environment Extension	
Cyber Physical Systems relations	

Figure 7 Basic integration description pattern for Chapter 3

The headings of the rows in the table depicted in Figure 7 follow the list of aspects proposed in Figure 5. Some extra aspects (e.g., the connection to CPS and the research challenges) have been added to better align the integrations with other high-level objectives of the project, also providing some indication that the Integration-Mediator Pattern is flexible enough to adapt to the needs of an evolving process.

In the following chapters as well as in the following deliverables that will follow this work (e.g., D5.3 and D5.4 but also indirectly D5.6) this methodology will be leveraged to extract more information about the extent of the integrations in the AIDOaRt framework and how they work together in a real-world scenario such as solving the required integrations for the project's Use Cases. It can be concluded that the results of applying the Integration-Mediator Pattern have been very useful and together with the wealth of knowledge collected in our AIDOaRt Modelio model, enable us to go deeper into integration that what would have been possible otherwise.

3 Integration framework: Vertical Integrations

This chapter focuses on the implementation of the proposed integration methodology on the AIDOaRt WP5 so far as described in Chapter 2.

The objective is to stay as close as possible to the Integration-Mediator pattern, but for the sake of simplicity and due to the learning curve on the partners, a few simplifications have been made.

- In this deliverable we only document the results of vertical integrations --i.e., integrations that correspond to only one use case, and overlooking any horizontal, and cross-use case integrations.
- The focus is given on Direct Relations, i.e., relations between elements that have been explicitly identified by the use case/solution providers In a few instances, analysis of the Modelio model has been conducted to find indirect relationships, but these analyses were not the fundamental part of the analysis. These relationships are rated on a Likert derived scale, from -2 (strong reject) and -1 (weak reject) to 0 (neutral), +1 (weak confirm) and +2 (strong confirm) to rate the perceived quality of these found relationships.
- As a starting point of these Direct Relations, the results of the hackathons #1 and #2 conducted during the last two AIDOaRt meetings (the first virtually and the second face-to-face at a meeting in l’Aquila in October 2022) are described. These provide an excellent study subject for collaborations that already can be analyzed with the Integration-Mediator Pattern.
- As for the completeness of the application of the pattern, we focused on many of the defined aspects but left out for the time being some others (e.g., the analysis of Integration based on the application of results coming from the SMS). These will be documented in further stages of work of T5.1, such as in D5.3.

For the compilation of results for this chapter, different parts of the methodology are segregated into *micro tasks* which as mentioned in Chapter 2were already discussed in WP1 deliverables. This enabled the process to be agile and gave the possibility of tracking down the evolution with relative ease.

3.1 Big Data Monitoring Solution (PRO)

Collaboration on PRO_UCS3 and PRO_UCS4

Research Challenge – Big Data Monitoring Platform - (Hackathon 1)

Participant solutions: ACO (Position Monitoring for Industrial Environment), ITI (ak2-modev, ak2-runman) and UOC (AsyncAPI Toolkit).

Goal: The goal of the Hackathon 1 was to define the architecture of the Monitoring platform and select the monitor parameters.

In order to be able to carry out the different analyses proposed for the use case, it is essential to have the appropriate data. To this end, it is necessary to implement a new monitoring system that collects the necessary information to apply the analytics.

- Improve the current monitoring system of the platform.
- Monitor more parameters (infrastructure and application).
- All the collected Information will be needed for future analysis (AI).

Challenge:

Monitoring the flow of data and ensuring that the information is processed correctly without data loss:

1. Monitoring the infrastructure (+ some port operations).
2. Analyze data to find anomalies and patterns.
3. Predictions.
4. Dynamic resource allocation.
5. Automatic deployment and testing.

Approach:

The first step is the “Data gathering” related to:

- Infrastructure resources used.
- IoT data received (frequency, accuracy, errors, alerts).
- Specific functionalities of the application (bottleneck detection).

Once we have the data, we could apply “AI Techniques” in order to detect patterns and anomalies.

Remaining challenges and next steps:

In the next iterations of the use case, all collected information will be used to detect anomalies and try to predict them, for that reason it is very important that all the required data needed to apply anomaly detection will be monitored:

1. Provide the infrastructure of the use case + the initial monitoring platform.
2. Review the monitoring platform and the data collected.
3. Start applying different analytics:
    - IoT data received.
    - Pattern detection.
    - Forecast.

Aspect	Analysis Details
Research Challenges	The first challenge of this scenario is to define a monitoring solution able to control and gather data from a Big data Smart port platform. The desired solution will be monitoring the big data infrastructure (computation nodes) and the software components, both could have temporary or permanent failures and the system should be able to automatically detect them. Monitoring the flow of data and ensuring that the information is processed correctly without loss of data is another desired improvement. This 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. The research challenges are: (a) The solution should be able to handle a large quantity of data to be analyzed. (b) The solution should be able to monitor the infrastructure and the software that is running. The challenges (a) and (b), are technological challenges. Thanks to the expertise of the partners and after reviewing the best open-source solutions, it was decided to use Grafana and Prometheus as a monitoring platform. Specific Research Challenges (Hackathon 1) ● Investigation of suitable ML based algorithms for anomaly detection from system performance metrics and monitored sensor data streams. (a2k/detection service). ● Investigation of how to simulate synthetic anomalies for training the detection algorithms. (a2k/scheduling service).
SotA Related Work	To detect anomalies, ITI has evaluated the following methods: ● Machine learning: K-Nearest Neighbors (KNN), One-Class Support Vector Machine (OC-SVM), Isolation Forest (IF), Local Outlier Factor (LOF), Density-Based Spatial Clustering of Application with Noise (DBSCAN), Principal Component Analysis (PCA) and Gaussian Mixture Model (GMM). ● Deep Learning: Autoencoders (AE)

ITI has tested these methods with public data-based and data from simulations. These data are similar to those that a smart port produces. In addition, other deep anomaly detection algorithms, such as recurrent neural networks, generative adversarial networks, and hybrid methods, are being studied to address challenging detection problems usually found in real-world applications and that are commonly studied in the literature.

There is a lot of literature related to anomaly detection methods, in our case we need to focus on methods able to detect problems in IT infrastructures (availability and performance of the infrastructure) and in data gathering from IoT devices. The most cited research articles on this topic are:

(a) S. Yin and O. Kaynak, "Big Data for Modern Industry: Challenges and Trends [Point of View]," in Proceedings of the IEEE, vol. 103, no. 2, pp. 143-146, Feb. 2015, doi: 10.1109/JPROC.2015.2388958.

(b) S. Sagiroglu and D. Sinanc, "Big data: A review," 2013 International Conference on Collaboration Technologies and Systems (CTS), 2013, pp. 42-47, doi: 10.1109/CTS.2013.6567202.

(c) Elgendy, N., Elragal, A. (2014). Big Data Analytics: A Literature Review Paper. In: Perner, P. (eds) Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2014. Lecture Notes in Computer Science(), vol 8557. Springer, Cham. https://doi.org/10.1007/978-3-319-08976-8_16

(d) Y. Nait Malek, A. Kharbouch, H. El Khoukhi, M. Bakhouya, V. De Florio, D. El Ouadghiri, S. Latre, C. Blondia,

(e) On the use of IoT and Big Data Technologies for Real-time Monitoring and Data Processing, Procedia Computer Science, Volume 113, 2017, Pages 429-434, ISSN 1877-0509,

(f) REN, Hansheng, et al. Time-series anomaly detection service at Microsoft. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019. p. 3009-3017).

(g) LAPTEV, Nikolay; AMIZADEH, Saeed; FLINT, Ian. Generic and scalable framework for automated time-series anomaly detection. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. 2015. p. 1939-1947.

(h) SALVADOR, Stan; CHAN, Philip; BRODIE, John. Learning States and Rules for Time Series Anomaly Detection. In FLAIRS conference. 2004. p. 306-311.

(i) WEN, Tailai; KEYES, Roy. Time series anomaly detection using convolutional neural networks and transfer learning. arXiv preprint arXiv:1905.13628, 2019.

(j) SHAUKAT, Kamran, et al. A review of time-series anomaly detection techniques: A step to future perspectives. In Future of Information and Communication Conference. Springer, Cham, 2021. p. 865-877.

(k) Pang, Guansong, et al. Deep Learning for Anomaly Detection: A Review. ACM computing Surveys. 2021. 54 (2). p. 1-38.

Generic Requirements	The collaboration, directly addressing the requirement PRO_R07, which is related to the Generic Requirements GR MON 1.3, GR MON 1.4, GR MON 2.7 and GR MON 2.9. The solution proposed and defined during the hackathon addresses the requirements in the following way: ● GR MON 1.3. The monitoring platform proposed will be able to collect data during operation of the system. ● GR MON 1.4. The monitoring platform proposed will be able to collect data related to the consumption of infrastructure resources. **Confirmed collaboration** ● Position Monitoring for Industrial Environment (ACO), tools for monitoring and control provide for acquisitions of metrics. ● a2k-modev (ITI), for data collection and data management, ingestion, handling. Also helps in automation for anomaly detection and modeling of the port infrastructure in terms of edge devices, gateways, and cloud services. ● a2k-runman (ITI), for data collection and data management, ingestion, handling. Also helps in automation for anomaly detection in real-time. ● AsyncAPI Toolkit (UOC), it allows users to define message-driven APIs in a machine-readable format. **Potential collaborations** ● ESDE (ACO) (Indirect- both). Software Development Environment to improve embedded software design productivity, not related to UCS. Relationship -2 (strong reject). ● HIB_logAnalyzer (HIB) (Indirect- both). UCS is not an NLP problem so adaptation to UC would be challenging. Relationship -2 (Strong Reject). ● Cloud expertise (AND), (Indirect - generic requirements mapping). Cloud architecture and infrastructure development for public cloud providers are not needed in the UCS. Relationship 0 (Neutral). ● Infrastructure as Code (IaC) expertise (AND) (Indirect - generic req. mapping). Provides IaC solutions for cloud infrastructure, etc., but there is no need in the UCS, because the IaC is already created. Relationship 0 (Neutral). ● Keptn (DT) (Indirect - generic req. mapping) Not a DevOps automation problem. Relationship -2 (Strong Reject).

Architecture Component	***Potential collaborations*** ● Kolga (AND) (Indirect), no deployment pipelines used in UCS. Relationship -2 (Strong reject). ● devmate (AST)(Indirect), no software for automation test code generation needed. Relationship -1 (Weak reject). ● TemporalEMF (UOC) (Indirect), no temporal metamodeling is needed. Relationship -2 (Strong reject)**.**
Architecture Component Interface	Based on UCS requirements, the following interfaces are related: ● IF-MODEL-LOADING: relationship +2 (Strong accept). ● IF-MODEL-NAVIGATION: relationship +2 (Strong accept). ● IF-MODEL-TRANSFORMATION: relationship +2 (Strong Accept). ● IF-MODEL-SAVING: relationship +2 (Strong accept).
Data engineering	Data collection: this collaboration with ITI will help in the improvement of the Port Monitoring Platform by assisting to ensure that the necessary resources are being used for a proper functioning. This data will be used to perform an anomaly detection or prediction analysis to identify critical conditions, which will help determine the needs of the platform in different scenarios. **Relationship +2 (Strong Accept)**.
Use Case Environment Extension	The objective of the UC is to monitor infrastructure resources consumption and data gathering, a generic problem with potential to be applicable to other UCs. As mentioned in the previous section, the capacity of "data collection" has been improved and this capability can be extended to other UCs.
Cyber Physical Systems relations	The use case presents a smart port platform, which is a network of interacting processing elements with physical input and output to/from sensors and actuators, and with multiple computer systems monitoring the nodes. Such an architecture is a classic cyber-physical system. Therefore, the solution to handle a large quantity of data and monitor the infrastructures can be helpful for other cyber-physical systems.

3.2 Anomaly Detection (PRO)

Collaboration on PRO_UCS

Research Challenge – Anomaly detection for IoT Data Platforms - (Hackathon 2)

Participant solutions: ACO (Position Monitoring for Industrial Environment), ITI (ak2-modev, ak2-runman) and UOC (AsyncAPI Toolkit)

Goal:

This challenge is about finding problems as soon as possible in order to try to prevent them from occurring, by using anomaly detection techniques.

The ultimate goal is to enable the possibility to actuate, to ensure at least a smooth monitoring capability, and as a consequence, smooth port operation. In this sense, the use case reflects a distributed CPS example, relying on a heterogeneous and dispersed set of sensors and with a specific type of actuation to ensure proper monitoring.

Challenge:

The lack of accuracy in the data affects the analysis performed on the platform, and consequently, the decision making based on this analysis. The challenge consists of finding problems in the data and ensuring that the devices linked to the platform behave as agreed. Moreover, ensuring that the platform has the necessary resources to ensure its proper functioning is another of the subjects to be solved.

This challenge was divided into several sub-challenges:

- Problems with the platform itself, for that purpose a monitoring system has been installed. Thanks to this system, information about the platform is being collected.
- Problems with the Service Level Agreement (SLA) of the sensors (frequency, % of correct data received)
- Problems with the accuracy of the data (some sensors have a limited time life and should be changed periodically)
- Detecting anomaly patterns.

Approach:

The scope of the challenge is very broad. For this reason, it was divided into 3 sub-challenges that are shown in the following Figure 8:

The first one “Data Quality IoT” focuses on ensuring that the information received is correct. The second, “Infrastructure Performance & Availability” for guaranteeing that the infrastructure and elements are available and sufficient to process the data. Finally, in the "Location Optimization", for improving the position monitoring of the different elements, their accuracy and resilience, exploiting the distributed architecture, and better integrating the data in an advanced monitoring platform.

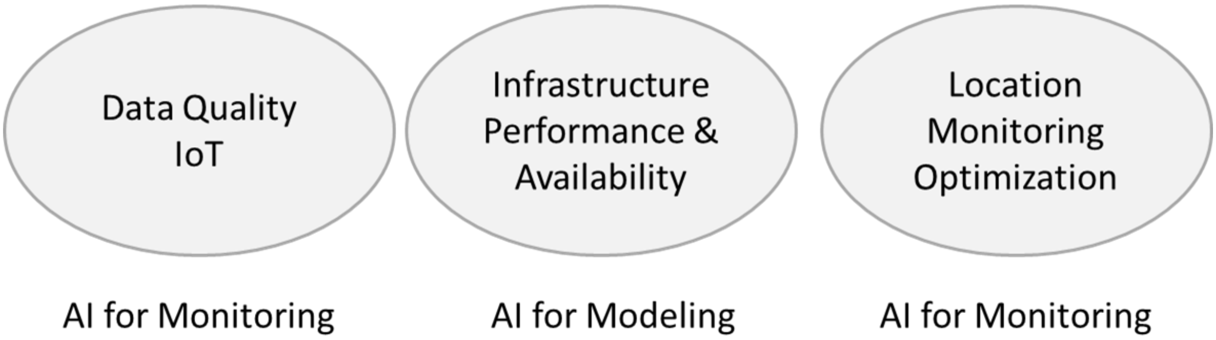

Data Quality IoT
Infrastructure Performance & Availability
Location Monitoring Optimization
AI for Monitoring
AI for Modeling
AI for Monitoring

Figure 8 Three sub-challenges in the anomaly detection challenge.

After working in these three fields, the final result will be a more complete platform capable of detecting anomalies in the data efficiently.

Remaining challenges and next steps:
In the Hackathon 2 (October 2022), significant progress was made for 3 sub-challenges proposed by the use case:

- The methodology for sending data has been defined. System monitoring can start from this point.
- Initial parameters for simulating self-healing mechanisms have been identified.
- Finally, the needs to improve positioning monitoring, relying on GNSS, UWB, and the Cloud-Continuum architecture.

In the future, Apache Kafka native support will be added as the messaging infrastructure. When this new milestone is achieved, the focus will be on the generation of monitoring code for anomaly detection in the QoS of the infrastructure.

Aspect	Analysis Details
Research Challenges	The challenge is the continuation of the "Big Data Monitoring Platform". Because before being able to carry out information analysis, it is necessary to collect quality data so that the work done on them produces good results. The research challenge is: the system should be able to detect a problem based on anomaly detection and other AI techniques. This challenge, unlike those proposed in Hackathon 1 (which had a more technological nature), has a more important role from the research point of view. Specific Research Challenges (hackathon 2).

	● Anomaly detection techniques. A specific challenge is in the training of machine learning algorithms for anomaly detection. We are currently investigating and developing simulation tools for this purpose. ● Simulation techniques to determine the needs of the platform in different scenarios and for optimization thereof under multiple objectives and constraints. ● Monitor data quality of IoT sensors and platforms.
SotA Related Work	There is a lot of literature related to self-healing and self-learning, and anomaly detection methods. In our case the scope is focused on methods able to detect problems in IT infrastructures (Availability and performance of the infrastructure) and in data gathering from IoT devices. The most cited research articles on this topic are: (l) Alonso, Juncal, et al., Optimization and Prediction Techniques for Self-Healing and Self-Learning Applications in a Trustworthy Cloud Continuum, Information, vol. 12, nº 308, 2021. (m) Ruban, I., Martovytskyy, V., Barkovska, O. (2022). Self-healing Systems Monitoring. In: Ruban, I., Kovalenko, A., Levashenko, V. (eds) Advances in Self-healing Systems Monitoring and Data Processing. Studies in Systems, Decision and Control, vol 425. Springer, Cham. https://doi.org/10.1007/978-3-030-96546-4_1 (n) Gheibi, Omid et al., Ian. Applying Machine Learning in Self-Adaptive Systems: A Systematic Literature Review, vol. 15, nº 9, 2021. (o) Sterritt, Roy, Autonomic networks: engineering the self-healing property, Engineering Applications of Artificial Intelligence, Volume 17, Issue 7, 2004, Pages 727-739, ISSN 0952-1976. (p) SA Malik, TM Gondal, S. Ahmad, M. Adil y R. Qureshi, "Towards Optimization Approaches in Smart Grid A Review", 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019, pp. 1- 5, doi: 10.1109/ICOMET.2019.8673392. (q) V. Degeler, R. French and K. Jones, "Self-Healing Intrusion Detection System Concept," 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016, pp. 351-356, doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.27.
Generic Requirements	The collaboration, directly addressing the requirements PRO_06 and PRO_R07, which are related with the Generic Requirements GR MON 1.3, GR MON 1.4, GR MON 2.7 GR MON 2.8, and GR MON 2.9. The solution proposed and defined during the hackathon addresses the requirements in the following way:

	● GR MON 2.7. The monitoring will improve the availability of the system guaranteeing that the infrastructure and elements are available and sufficient to process the data. ● GR MON 2.8. To predict the demand for the resource based we will also use data from the historical demand. ● GR MON 2.9. Through monitoring the availability of the system will be improved and track the resources demand of the system. **Confirmed collaboration** ● Position Monitoring for Industrial Environment (ACO) (Direct), tools for monitoring and control provide for acquisitions of metrics. ● a2k-modev (ITI) (Direct), provides analysis for computation and display of various system performance metrics for helping in automation for anomaly detection. ● a2k-runman (ITI) (Direct), for monitoring the operation of a cyber-physical system in real-time and to provide warnings and advice when critical situations are observed or predicted for anomaly detection. ● AsyncAPI Toolkit (UOC) (Direct) for communications in a machine-readable format. Potential collaborations ● ESDE (ACO) (Indirect- both). Software Development Environment to improve embedded software design productivity, to be demonstrated on the positioning IOT, and enabling anomaly analysis techniques on embedded level related to UCS3. Relationship 0 (Neutral). ● HIB_logAnalyzer (HIB) (Indirect- both). UCS is not an NLP problem so adaptation to UC would be challenging. Relationship -2 (Strong reject). ● Cloud expertise (AND), (generic requirements mapping). Cloud architecture and infrastructure development for public cloud providers are not needed in the UCS. Relationship 0 (Neutral). ● Infrastructure as Code (IaC) expertise (AND) (Indirect - generic req. mapping). Provides IaC solutions for cloud infrastructure, database setups, etc., but there is no need in the UCS, because the IaC is already created. Relationship 0 (Neutral). ● Keptn (DT) (generic req. mapping) Not a DevOps automation problem. Relationship -2 (Strong reject). ● Modeling Process Mining Tool (JKU) (Indirect - generic req. mapping), no graphical editor is needed to catch events and trace them. Relationship -2 (Strong reject). ● ConvHandler (ROTECH) (Indirect - generic req. mapping), data management was managed in Hackathon 1. Relationship 0 (Neutral).

	● Bridger (ROTECH) (Indirect - generic req. mapping), communication task was managed in Hackathon 1. Relationship -2 (Strong reject). ● DataAggregator (ROTECH), data management was managed in Hackathon 1. Relationship 0 (Neutral).
Architecture Component	● a2k-runman (ITI) (Direct). Relationship +2 (Strong accept). ● AsyncAPI Toolkit (UOC) (Direct). Relationship +2 (Strong accept). ● Kolga (AND) (Indirect), no deployment pipelines used in UCS. Relationship -2 (Strong accept). ● devmate (AST)(Indirect), no software for automation test code generation needed. Relationship -1 (Weak reject). ● TemporalEMF (UOC) (Indirect), no temporal metamodeling is needed. Relationship -2 (Strong reject). The following AIDOaRt architecture components will be impacted: Core Tool Set: ● *Computation Capabilities*: For implementation of compositional performance analysis algorithms, anomaly detection algorithms (both in training and detection modes), and for implementation of the optimization algorithms. ● *Model Based Capabilities*: A continuum computing model is under development for the smart port monitoring architecture.
Architecture Component Interface	Based on UCS requirements, the following interfaces are related: ● IF-INSIGHT-ANALYSIS: relationship +2 (Strong accept). ● AI/ML for Anomaly Detection: relationship +2 (Strong accept) ● IF-RUNTIME-DATA-COLLECTION: relationship +2 (Strong accept). ● IF-DATA-FILTERING-AGGREGATION: relationship +2 (Strong accept). ● IF-DATA-LOADING: relationship +2 (Strong accept). ● IF-CONTINUOUS-MONITORING: relationship +1 (Weak accept). ● IF-PREDICTIVE-ALALYSIS: relationship +2 (Strong accept). ● ML-based Prediction For Performance and Resource Utilization: relationship +1 (Weak accept). ● IF-RESPONSE-AUTOMATION: relationship 0 (Neutral). ● IF-AI-FOR-MONITORING: relationship 0 (Neutral).
Data engineering	The hackathon 2 collaboration relates to the following Data Engineering components: ● **Data collection:** this collaboration will help in the improvement of the Port Monitoring Platform by assisting to ensure that the necessary resources are being used for a proper functioning. For this purpose, a simulation of the port infrastructure performance is being carried out, acquiring the necessary data, both runtime and design time. This data will be used to perform an anomaly detection or prediction analysis to

	identify critical conditions, which will help determine the needs of the platform in different scenarios. **Relationship: +2**. ● **Data Management:** there is not an important improvement in this component.
Use Case Environment Extension	The objective of the UC in this second hackathon is to ensure that the data collected is correct and that the infrastructure resources are sufficient to process them so it would potentially be applicable to other use cases.
Cyber Physical Systems relations	The use case presents a smart port platform, which is a network of interacting processing elements with physical input and output to/from sensors and actuators, and with multiple computer systems monitoring the nodes. Such an architecture is a classic cyber-physical system. Therefore, the solution to handle a large quantity of data and monitor the infrastructures can be helpful for other cyber-physical systems.

3.3 500 Nights of Testing (Westermo)

Collaboration on W_UCS_1, W_UCS_2, W_UCS_3

Research challenge: Visualize, explore or analyze test results data from 500 nights of testing.

Participant solutions; INTECS, and ABO, no concrete solutions but exploratory collaboration.

Goal:

This was an open-ended challenge, with a data set in the center. From a high-level perspective, Westermo wished to tighten collaborations with the partners in the project, bridge gaps between industry and academia, explore applied AI, and learn more about techniques and practices suitable for our type of data.

Furthermore, Westermo has approved to publish this data set for the general public. This was released in Westermo's GitHub page (https://github.com/westermo) during the development period of this deliverable.

Challenge:

From the perspective of Westermo, we have progressed well with respect to test automation and data collection. We are improving our DevOps, and we know we wish to further improve it. However, knowing exactly how to do this may be hard, and we hoped that this exploratory collaboration would find suitable ways to progress beyond the initial scope of AIDOaRt.

At the center of this challenge is a data set. From a company perspective, understanding what this data is about is almost trivial since we collected it and are working with this type of data every day. However, to explain what this data is about, and the context it was collected in was not trivial, at least not initially.

Approach:

This challenge was explored with the goal of identifying test case dependencies in terms of tests that pass and fail together (as described below). This was done from four perspectives: (i) using a home-made similarity score, (ii) using correlation, (iii) using association rules, and (iv) visualizing any dependencies. All four approaches showed initial promising results. The Figure below illustrates test cases (by their ID) with correlating time series of verdicts.

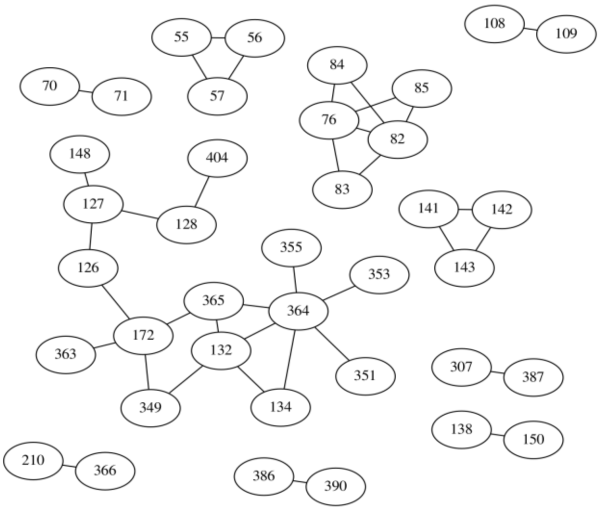


Figure 9 Initial illustration of test cases (by their ID) with correlating time series of verdicts.

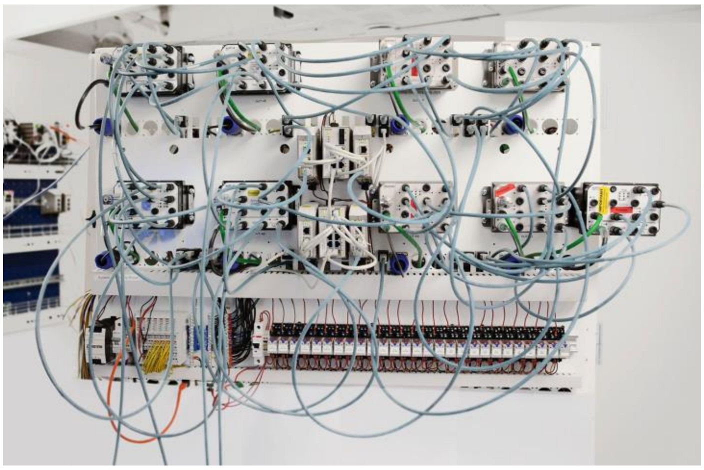
Remaining challenges and next steps:

This hackathon challenge answered several questions: (i) Are there test cases that pass and fail together? Answer: yes. (ii) can we make a meaningful visualization of these? Answer: yes. However, some unanswered questions are: (iii) How can this information be useful in everyday work at Westermo? Also: (iv) How do these correlations change over time? And (v) What is the root cause of these correlations? Perhaps future work in AIDOaRt, or after, will explore this more.

Aspect	Analysis Details
Research Challenges	In the industrial context at Westermo, we do automated nightly regression testing each night. These test results are multi-dimensional in that we test, for each software project (branch), many test cases, on many test systems, with a set of possible verdicts (pass, fail, invalid, …). The purpose of the hackathon collaboration was to use a data set from Westermo, in order to explore: (a) Test case dependencies, i.e., are there test cases A and B such that, if A fails, then B also always (or almost always) fails? There are several potential reasons for why this could be the case. (i) A and B could test the same thing C. If so, do we really need both A and B? (ii) It could also be the case that A pollutes the system state for B, e.g., by corrupting some resources in the test environment. (b) Test results visualization. Presenting the multidimensional test results in a way such that colleagues at Westermo can make them actionable is non-trivial -- so what are good ways of doing this? (c) When test automation continues over time, there is an increased production of test cases. Each test case requires resources in terms of devices needed for testing, and a non-zero time. When we are out of

	resources, in particular time, we have to start prioritizing test cases. How could that be done in Westermo's context? Due to the limited time scope of the Hackathon, we only explored the first topic: test case dependencies. It is possible that the data set used could be released to the general public such that future collaborations or other research groups could explore other topics.
SotA Related Work	There is a very large body of knowledge on regression test selection (the third topic). In the AIDOaRt literature (mapping) study, two papers are of particular importance since they both cover test selection (a topic not covered in the Hackathon): [1] Medhat et al. (2020). A framework for continuous regression and integration testing in IoT systems… *IEEE Access*. [2] Azizi. (2021). A tag-based recommender system for regression test case prioritization. *IEEE ICST Workshops*. In our own previous work, we have explored decision-making and test results visualization, see below. Again, this topic was not covered in the Hackathon. By releasing a public data set, broader research could be encouraged. [3] Strandberg et al. (2019). Information flow in software testing–an interview study with embedded software engineering practitioners. *IEEE Access*. [4] Strandberg et al. (2022). Software test results exploration and visualization with continuous integration and nightly testing. *Springer STTT*. There is quite a bit of work previously conducted on test case dependencies, e.g., in the automotive domain and wrt. code smells, and on detecting false test and test dependencies using association rules method, with special focus to a priori algorithm. [5] Arlt et al. (2015). If A fails, can B still succeed? Inferring dependencies between test results in automotive system testing. *IEEE ICST*. [6] Garousi and Küçük. (2018). Smells in software test code: A survey of knowledge in industry and academia. *Elsevier JSS*. [7] Tahvili et al. (2016). Dynamic Integration Test Selection Based on Test Case Dependencies. *IEEE ICSTW*. [8] Chawla (2010). Feature Selection, Association Rules Network and Theory Building. *PMLR*. [9] Herzig and Nagappan (2015). Empirically Detecting False Test Alarms Using Association Rules. *IEEE/ACM*.
Generic Requirements	**Test.6 integrate and analyze the testing phase into the DevOps pipeline**: identifying test case dependencies is a type of automated analysis (done in the DevOps phase).

	Mon1.1 access off-line data, and **Mon1.3 access on-line data**: for the hackathon, we accessed off-line data, but a solution coming out of it could work on on-line data. **Mon2.3 monitor and identify clusters of anomalies**: if there are (unknown) test case dependencies, the clusters of related tests coming out of the analysis would be first identified, and then they could be monitored.
Architecture Component	The collaboration impacts the following architecture components: **Engagement & Analysis**: automated analysis of test results in order to enhance test results exploration is a strong link to engagement and analysis since it helps test results consumers. Relationship: +2 **AI for Monitoring**: using AI to monitor e.g., test case dependencies represents a strong link with this collaboration. Relationship: +2 **Data Collection**: this architecture component is relevant for this collaboration, but is not at the core of the challenge. Relationship: +1 **Data Management**: this architecture component is relevant for this collaboration, but is not at the core of the challenge. Relationship: +1 **Data Representation**: this architecture component is relevant for this collaboration, but is not at the core of the challenge. Relationship: +1 **Storage Capabilities**: this architecture component is relevant for this collaboration, but is not at the core of the challenge. Relationship: +1
Architecture Component Interface	The collaboration impacts the following generic architecture component interfaces: **IF-INSIGHT-ANALYSIS**: root cause investigation of test case dependencies based on ML techniques. Relationship: +2 **IF-PREDICTIVE-ANALYSIS**: usage of retrieved dependencies in order to prevent unnecessary testing based on ML techniques. Relationship: +2 **IF-AI-FOR-MONITORING**: ML based capabilities to support the monitoring phase of automated nightly regression testing. Relationship: +1 **IF-DATA-TRANSFORMATION**: capabilities to transform collected test log data in anonymized and simplified way (path semantic, visualization strategies). Relationship: +2 **IF-DATA-FILTERING-AGGREGATION**: capabilities to filter and aggregate data related to different test logs. Relationship: +2

	In our collaboration, we further break these links down as follows: **IF-AI-FOR-MONITORING**: *AI for Text Analytics*, usage of ML techniques to analyze large quantities of text representing test logs. Relationship: +2 **IF-AI-FOR-TESTING:** *AI for Test Case Reduction*, usage ML techniques to identify a subset of tests to be performed among all possible ones. Relationship: +2 **IF-INSIGHT-ANALYSIS:** *AI/ML for Anomaly Detection*, usage of ML algorithms for the detection of anomalies in time series monitoring data (related test failures). Relationship: +2
Data engineering	At Westermo, a test results database is in place. This database has a structure described in Strandberg, P. E., Afzal, W., & Sundmark, D. (2022). Software test results exploration and visualization with continuous integration and nightly testing. *International Journal on Software Tools for Technology Transfer*, *24*(2), 261-285. Knowledge of this structure, experience and enhancements of it are valuable to AIDOaRt, and vice versa. In the hackathon, a simplified structure was used (it was exported to a CSV file).
Use Case Environment Extension	Yes, a tool coming out of this hackathon could potentially be built into Westermo's DevOps tool chain, or enhance the system already in place for exploring test results.
Cyber Physical Systems relations	Westermo develops switches and routers for industrial communication networks, such as on-board rail, power distribution or industry automation, etc. In the nightly testing at Westermo, this testing is conducted on a number of heterogeneous test systems. One of more than 20 physical test systems is illustrated below (in addition to physical ones, there are also half a dozen test systems with virtualized devices.  When testing, one or several devices are used in each test case, a network topology is configured for each test case and ports are enabled or disabled, etc., based on the needs of each test case. This way, the same test case can run on different test systems with different types of product families.

	This means that, each night, for each software version being tested, the test results stem from several test cases that ran on several test systems. This gives the testing realism in the sense that **timing** in the test system would be the same as for a customer using Westermo products in their networks. It may also mean that, on a certain hardware product, a software feature may not work, because of issues **in the software-hardware-integration** on this particular product. This also makes the testing **slow**, as these products are physical and resource constrained when compared to testing without dedicated hardware (such as on some server in the cloud). There may also be some test case dependencies, e.g., a test case that **pollutes the system state** on a device so that another test case fails. Finally, the test results are **multi-dimensional** and thousands of **log files** are produced each night, so exploring test results may be difficult. Because of these distinct CPS-related topics (differences in timing over products, software-hardware-integration, slow testing, system state pollution multi-dimensional test results, vast amounts of log files), the test results consumption may be difficult, and we aimed at the challenges of exploring (a) test case dependencies, (b) test results visualization, and (c) test case prioritization in this hackathon.

3.4 Exploring Test Results Data (Westermo)

Collaboration on W_UCS_1, W_UCS_2, W_UCS_3

Research challenge: Visualize, explore, or analyze test results data from nightly testing.

Participant solutions: CRT (Copado), Flaky Test Detector (RISE)

Goal:

This hackathon challenge is an extension of Westermo's first hackathon challenge (see section 3.3). However, this year we focused on more qualitative aspects, i.e., on flaky tests and performance of the testing (in terms of duration of test executions).

Challenge:

As for the first year's hackathon, this challenge had a focus on data, but this time on quality attributes. Perhaps one could think of test result data as having had three stages at Westermo:

- showing a list of failing tests -- done before AIDOaRt
- exploring test case correlations -- first hackathon
- exploring quality attributes of testing -- done in the second hackathon

Approach:

Similar to the first year, this was an exploratory challenge. Copado used their CRT tool to analyze possible changepoints in test case durations. One such change point, for the execution of one test case over time is illustrated below in Figure 10 -- the change point occurs when the plot changes color. RISE explored how and if a flaky test identifier could be implemented.

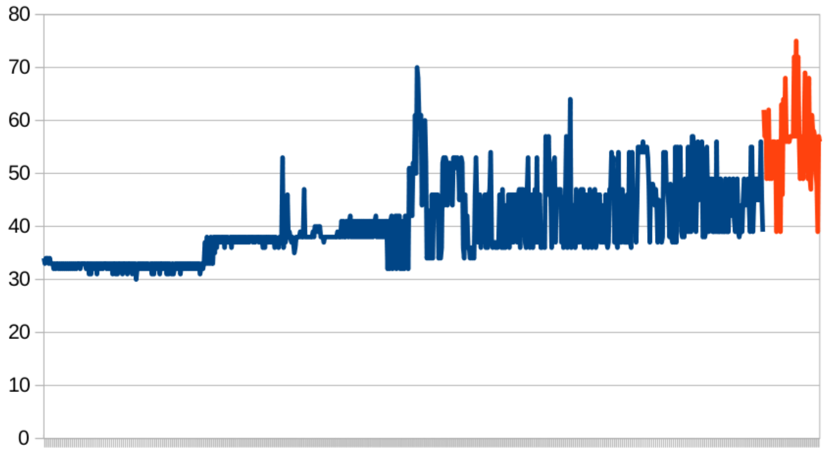


Figure 10 Illustration of test case execution durations (y-axis), for one Westermo test case on one Westermo test system, over time (x-axis).

Remaining challenges and next steps:

Like for the first year, this challenge answered some questions, e.g., are there change points in the durations in test cases, and can they be identified -- answer: Yes. Again, unanswered questions remain: how could this information be made available to Westermo staff, and what are the root causes (e.g., degradation in the software under test, or degradation in the test framework)?

Aspect	Analysis Details
Research Challenges	This collaboration was related to the hackathon 2, and used many of the same ingredients as the first year, see section 3.3, in particular the same data set. We asked if 1. some tests pass and fail together (as in hackathon 1), 2. there are some patterns when/how tests become flaky and for how long, 3. if there are correlations between code changes and changed verdicts, and 4. how one could present test results.
SotA Related Work	In addition to the papers mentioned in section 3.3, for the second year's challenge, the below papers are also of relevance: [1] Ahmad, A. (2022). *Contributions to Improving Feedback and Trust in Automated Testing and Continuous Integration and Delivery.* Doctoral dissertation, Linköping University. [2] Strandberg, P. E. (2021) *Automated System-Level Software Testing of Industrial Networked Embedded Systems.* Doctoral dissertation, Mälardalen University. [3] Barboni, M., Bertolino, A., & Angelis, G. D. (2021). What We Talk About When We Talk About Software Test Flakiness. In *QUATIC'21*. Springer. [4] Fatima, S., Ghaleb, T. A., & Briand, L. (2022). Flakify: A black-box, language model-based predictor for flaky tests. *IEEE Transactions on Software Engineering*. [5] Bell, J., Legunsen, O., Hilton, M., Eloussi, L., Yung, T., & Marinov, D. (2018, May). DeFlaker: Automatically detecting flaky tests. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (pp. 433-444). IEEE.

	[6] Killick, Rebecca & Fearnhead, Paul & Eckley, I.A. (2012). Optimal Detection of Changepoints With a Linear Computational Cost. Journal of the American Statistical Association. 107. 1590-1598. 10.1080/01621459.2012.737745. [7] Gachomo, Dorcas. (2015). The Power of the Pruned Exact Linear Time (PELT) Test in Multiple Changepoint Detection. American Journal of Theoretical and Applied Statistics. 4. 581. 10.11648/j.ajtas.20150406.30.
Generic Requirements	In addition to the generic requirements of relevance in the first hackathon (Test.6, Mon1.1, Mon1.3 and Mon2.3), see section 3.3, the second hackathon also covers: GR **Mon 2.4 monitor and identify unwanted trends/drifts:** Test cases could become flaky and exhibit degraded execution times. As a consequence, monitoring is of central importance in the hackathon. All of these have a strong accept (+2).
Architecture Component	+2 (strong accept) for engagement & analysis, as this challenge was on the topic of exploring test results data. +2 (strong accept) for components related to collecting and storing data, since this is a prerequisite for showing data: ● Data Collection ● Data Management ● Data Representation ● Storage Capabilities +2 (strong accept), for AI for Testing, since this is an AI tool used to aid understanding test results.
Architecture Component Interface	(+1 weak accept) - Based on the requirements for Flaky Test detection, the use case has an impact on the ***IF-AI-FOR-TESTING*** interface to support the testing phase of software development. (+2 strong accept) - Copado's solution for monitoring durations of test cases strongly links to ***IF-AI-FOR-MONITORING***
Data engineering	+2 strong accept - with the same motivation as section 3.3.
Use Case Environment Extension	A tool coming out of the second hackathon could potentially be built into Westermo's DevOps tool chain, or enhance the system already in place for exploring test results.
Cyber Physical Systems relations	This collaboration has the same type of relation to CPS as section 3.3.

3.5 LogGrouper (Westermo)

Collaboration on W_UCS_2

This collaboration targets W_UCS_2 on Prediction/Monitoring of reliability or root causes of failures based on development artifacts, in this case test framework log files.

Participant solutions: LogGrouper (RISE)

Goal:

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

Approach:

LogGrouper communicates with a Westermo system for test-result exploration to get a list of failing tests, and then collects the logs from these tests. The logs are pre-processed (e.g., for removal of stop words), vectorized and clustered. In the end, a JSON-structure is presented through a web-interface. This approach is illustrated in the Figure 11 below.

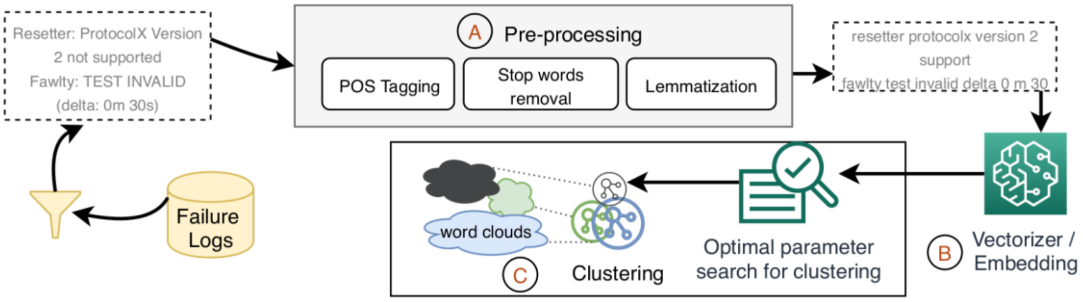

Resetter: ProtocolX Version 2 not supported Fawlty: TEST INVALID (delta: 0m 30s)
A
Pre-processing
POS Tagging
Stop words removal
Lemmatization
resetter protocolx version 2 support fawlty test invalid delta 0 m 30
Failure Logs
word clouds
C
Clustering
Optimal parameter search for clustering
B
Vectorizer / Embedding

Figure 11 Overall process of LogGrouper. Illustration from Abbas et al., Making Sense of Failure Logs..., (in submission).

Remaining challenges and next steps:

A prototype implementation of LogGrouper by RISE has been delivered to Westermo. Westermo is currently exploring how to incorporate it into their pre-existing systems. A simple front-end has been implemented that interfaces with LogGrouper and just shows whatever it produces. The result is a table that (for each cluster) contains: a cluster id, the number of verdicts in the cluster, a link each per verdict in the cluster, and a list of plausible log lines that are meant to explain what this cluster is about. Future steps are to improve the explanations of the clusters. Currently, there are many irrelevant words that have to be added to the stop list, and Westermo should probably also improve the contents of the logging to improve the tools performance. Finally, the clustering ought to be integrated into systems already in use, instead of being an external tool.

Aspect	Analysis Details
Research Challenges	The rapid development, updating, and maintenance of industrial software systems have increased the necessity for software test automation. Due to the potentially large amounts of test data such as log files, there is a desire to explore automated test analysis. E.g., one could imagine that the examination of test results could be automated by grouping them into subsets of comparable test outcomes for a batch analysis. Some of the goals of this collaboration is to explore: ● What impact does text preprocessing of logs have on the execution time and clustering quality of industrial test results? ● Which clustering approach provides the best results in computation time and evaluation metrics for clustering failure logs in industrial settings? ● To which degree do the selected clustering approaches agree with each other in the context of our dataset? ● How understandable and usable are the results of clustering from a human point of view?
SotA Related Work	Of particular importance for this collaboration is research on natural language processing (NLP), log analysis, and text clustering, e.g. [1] Eljasik-Swoboda and Demuth. (2020). Leveraging Clustering and Natural Language Processing to Overcome Variety Issues in Log Management. In ICAART. [2] Bertero et al. (2017) Experience report: Log mining using natural language processing and application to anomaly detection. In IEEE ISSRE. [3] Xiao et al. (2020). LPV: A log parser based on vectorization for offline and online log parsing. In IEEE ICDM. [4] Sharp et al. (2016). Semi-Autonomous Labeling of Unstructured Maintenance Log Data for Diagnostic Root Cause Analysis. APMS. [5] Fu et al. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. IEEE ICDM. [6] Lin et al. (2016). Log clustering based problem identification for online service systems. In IEEE/ACM ICSE-C. [7] Aussel et al. (2018). Improving performances of log mining for anomaly prediction through nlp-based log parsing. In IEEE MASCOTS. [8] He et al. (2016). An evaluation study on log parsing and its use in log mining. In IEEE/IFIP DSN. [9] Itkin et al. (2019). User-assisted log analysis for quality control of distributed fintech applications. In IEEE AITest. [10] Korzeniowski and Goczyła. (2022). Landscape of Automated Log Analysis: a Systematic Literature Review and Mapping Study. IEEE Access.

Generic Requirements	In addition to the generic requirements of relevance in the first hackathon (Test.6, Mon1.1, Mon1.3 and Mon2.3), see section 3.3, this collaboration also covers: GR **Mon 2.6 monitor and identify root causes to anomalies** and GR **Test 4 automated evaluation of test results**, since the logs from testing are analyzed, and relevant phrases are suggested as possible causes.
Architecture Component	+2 strong accept, for components related to collecting and storing logs: ● Data Collection ● Data Management ● Data Representation ● Storage Capabilities +2, strong accept, for Explainability -- certain keywords and key phrases are highlighted to illustrate why logs are grouped +2, strong accept, for AI for Testing, since this is an AI tool used to aid understanding test results
Architecture Component Interface	(+2 strong accept) - LogGrouper impacts the interface components: ***IF-AI-FOR-MONITORING*** and ***IF-AI-FOR-TESTING*** to support the monitoring and testing phase of the system development.
Data engineering	+2 strong accept - Westermo has more than 2 TB of log files, improving data engineering on this topic is very central.
Use Case Environment Extension	Westermo is currently working on integrating LogGrouper into their environment.
Cyber Physical Systems relations	This collaboration has the same type of relation to CPS as the one detailed in section 3.3.

3.6 Recommendation System for RE (Alstom)

Collaboration on: UCS_BT_1

Participant solutions:

1. Requirements Ambiguity Checker (MDU)

   Identifies ambiguous requirements from textual documents using a set of ambiguous keywords and patterns, and NLP & AI/ML techniques.

2. VARA (RISE)

   Automated similarity analysis and feature reuse recommendation using Natural Language Processing (NLP): VARA enables to perform automatic analysis of textual requirements for a new project and identify components and artifacts that can be reused from a previous project for the implementation of the new requirements based on similarity analysis.

3. Modelio (SOFT)

   Automated Requirements Identification, Extraction & Classification: Identify, extract, and classify requirements from textual documents with NLP & AI/ML techniques

Goal:

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

Today, as a consequence, vast numbers of customer requirements are manually analyzed, allocated and further broken down. To be carried out effectively, this usually requires highly experienced bid and customer project engineers.

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

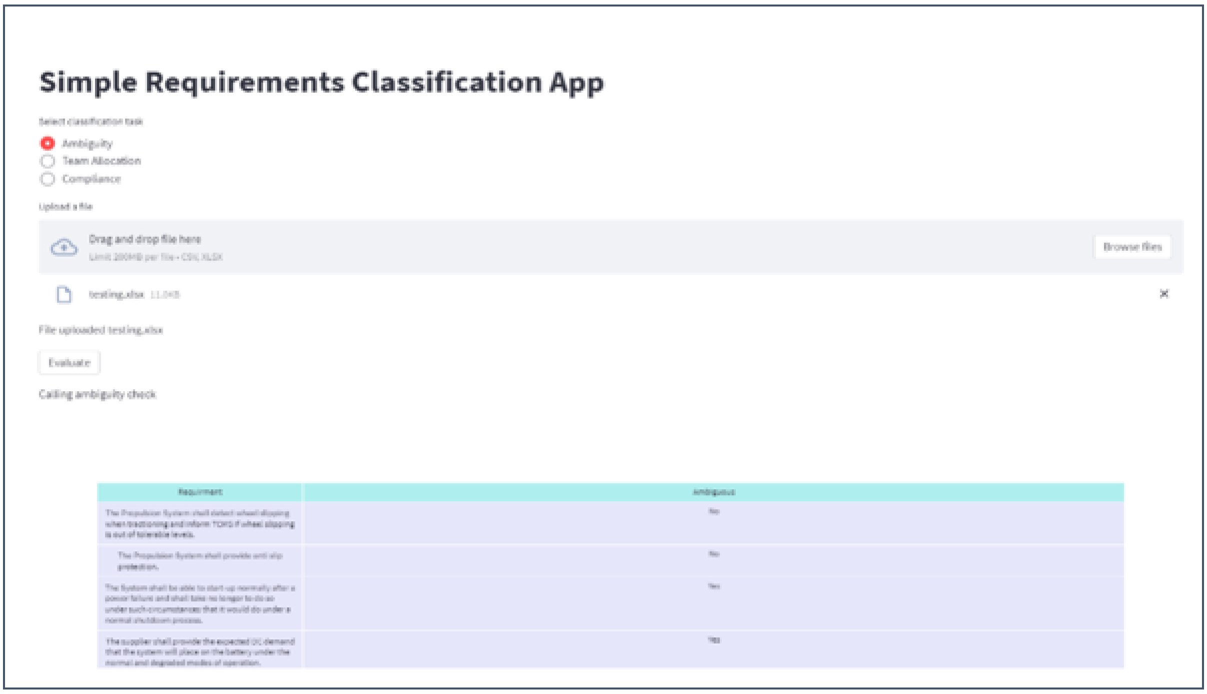
Challenges:

The common challenges in creating recommendation-system solutions stem from the nature of the data. There is a need to have appropriate volumes in the training data that are representative of the actual formats and variety encountered in reality. Both training and application data needs to be pre-processed wherein it is cleaned, requirements clearly identified and extracted. Last, but not least, there need to be means to evaluate data from an often-subjective labeling and make the output explainable.

Approach:

The approach has initially centered on certain elements of the requirements analysis process, namely ambiguity checking, similarity checking and team allocation prediction. Within AIDOaRt collaboration, tools that address these elements have been developed by applying NLP methods for pattern detection and transformer models. Using initial datasets, the preliminary feasibility of these tools has been successfully demonstrated as shown in Figure 12 and Figure 13:

Ambiguity checking:



Simple Requirements Classification App

Select classification task

● Ambiguity
○ Team Allocation
○ Compliance

Upload a file

Drag and drop file here
Limit 200MB per file • CSV, XLSX

testing.xlsx 11.0KB

File uploaded testing.xlsx

Calling ambiguity check

Requirment	Ambiguous
The Propulsion System shall detect wheel slipping when tractioning and inform TCMS if wheel slipping is out of tolerable levels.	No
The Propulsion System shall provide anti slip protection.	No
The System shall be able to start up normally after a power failure and shall take no longer to do so under such circumstances that it would do under a normal shutdown process.	Yes
The supplier shall provide the expected DC demand that the system will place on the battery under the normal and degraded modes of operation.	Yes

Figure 12 Illustration of Ambiguity Checking

Similarity checking:

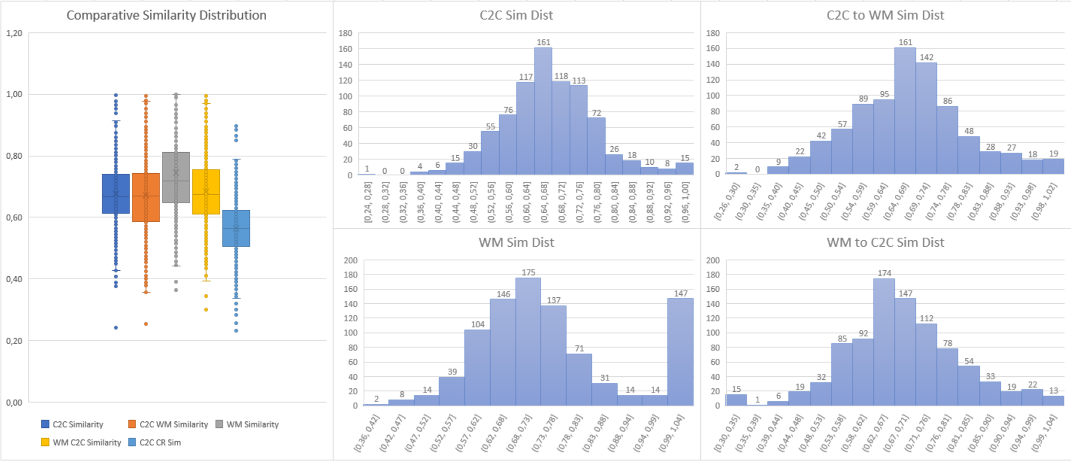


Figure 13 Illustration of Similarity Checking

Remaining challenges and next steps:

The initial labeled datasets have been relatively small. Steps to further validate the tools will be taken by providing relatively larger labeled datasets. Next actual response recommendations will be explored and developed.

Aspect	Analysis Details
Research Challenges	Common challenges ● Data volume, format & variety ● Data pre-processing (cleaning, requirements identification & extraction…) ● Evaluation (subjective data labeling…) & Explainability Requirements Assignment: Requirement assignment module is the smart allocation of customer requirements to their respective teams within the railway domain. The research challenges identified are: 1. Potential manual labeling biases of customer requirements, which can be resolved by validating requirements from multiple team members through surveys. 2. Less data volume of different project requirements to train SotA deep learning models. 3. In requirements engineering, most of the related work is classification of functional/non-functional requirements of the software systems. There is a

	research gap related to railway domain requirements and a baseline needs to be developed for evaluation of developed AI/ML models. Requirements Ambiguity Check: Requirement ambiguity checker is the classification of customer requirements into ambiguous/unambiguous within the railway domain. The research challenges identified are: 1. Less data volume. Data augmentation required to increase the size of data for training ML/AI models. 2. Unbalanced target classes in dataset. More project requirements required to balance the dataset for better training the AI/ML models. 3. Integration of the ambiguity checker module with the requirement team allocator module with the prospect of creating a tool, which first identifies if a requirement is unambiguous then allocates it to the related team, therefore facilitating in the automation of the requirements engineering process.
SotA Related Work	The Systematic Mapping Study (SMS) of AIDOaRt did not produce any relevant papers for the following reason: The SMS aim, research questions, and search strings were not designed to capture work related to BT_UCS1. Instead, it is proposed to search according to the following (a) "[{ NLP | Survey | Overview }] { requirements | sentence | text } ambiguity [check]" (b) "[{ NLP | Survey | Overview }] { requirements | sentence | text } similarity [check]" (c) "{Automated | AI/ML | NLP} requirements assignment" (d) Datasets, NN models, tools… & much more details The key selection of the research articles based on the direct searches regarding ambiguity checks are: 1. B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, S. Bacherini, Using nlp to detect requirements defects: An industrial experience in the railway domain, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2017, pp. 344–360. 2. M. Kassab, C. Neill, P. Laplante, State of practice in requirements engineering: contemporary data, Innovations in Systems and Software Engineering 10 (2014) 235–241. 3. C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, Automated checking of conformance to requirements templates using natural language processing, IEEE transactions on Software Engineering 41 (2015) 944–968.

	4. N. Carlson, P. Laplante, The NASA automated requirements measurement tool: a reconstruction, Innovations in Systems and Software Engineering 10 (2014) 77–91. 5. S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, Maana: An automated tool for domain-specific handling of ambiguity, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, 2021, pp. 188–189. 6. L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. Ajagbe, E.-V. Chioasca, R. T. Batista-Navarro, Natural language processing (nlp) for requirements engineering (re): A systematic mapping study, ACM Computing Surveys (2020). 7. M. Arrabito, A. Fantechi, S. Gnesi, L. Semini, An experience with the application of three nlp tools for the analysis of natural language requirements, in: International Conference on the Quality of Information and Communications Technology, Springer, 2020, pp. 488–498. 8. M. Q. Riaz, W. H. Butt, S. Rehman, Automatic detection of ambiguous software requirements: An insight, in: 2019 5th International Conference on Information Management (ICIM), IEEE, 2019, pp. 1–6. 9. K. H. Oo, A. Nordin, A. R. Ismail, S. Sulaiman, An analysis of ambiguity detection techniques for software requirements specification (srs), International Journal of Engineering & Technology 7 (2018) 501–505. 10. A. Yadav, A. Patel, M. Shah, A comprehensive review on resolving ambiguities in natural language processing, AI Open 2 (2021) 85–92
Generic Requirements	The collaboration links to the following generic requirements: ● **GR RE 01:** The Requirement Management Tool of the AIDOaRt Framework translates requirements from semi-structured language to formal language. ○ Relationship: +2 ● **GR RE 03:** The Requirement Management Tool of the AIDOaRt Framework analyzes the requirements expressed in formal language and produces suggestions or prescriptions for the Requirements Engineer and the System Engineer. ○ Relationship: +2
Architecture Component	The collaboration impacts the following architecture components: ● **AI for Requirements Engineering:** This component enables the integration of AI/ML techniques to support the requirements engineering phase of the system development ○ Relationship: +2 ● **Data Collection:** This component supports the data collection process from different data sources (both runtime and design time) in the AIDOaRt framework. ○ Relationship: +1 ● **Storage Capabilities:** The Storage Capabilities component supports the provisioning of physical and logical resources allowing to efficiently store and retrieve the possibly numerous and large data artifacts and models.

	○ Relationship: +1 ● **Data Handling Capabilities:** The Data Handling Capabilities component supports the loading, navigation, querying and then the saving of the required data. ○ Relationship: +1
Architecture Component Interface	The collaboration impacts the following architecture component interfaces: ● **AI for Requirements Ambiguity Check:** AI and NLP techniques used to identify ambiguity in textual requirements. ○ Relationship: +2 ● **AI for Requirements Similarity Check**: AI and NLP techniques used to evaluate similarity rate between textual requirements. ○ Relationship: +2 ● **AI for Requirements Allocation**: AI and NLP techniques used to auto assign textual requirements to teams based on allocations done in previous projects. ○ Relationship: +2 ● **AI for Specifications Consistency Verification:** AI/ML based capabilities (e.g., formal methods, automated reasoning, NLP methods...) used to enable automated consistency verification of technical specifications (w.r.t. standard guidelines, e.g., ISO, or specific criteria). ○ Relationship: +1
Data engineering	No particular contributions to data engineering are necessary for the use case. Existing tools will suffice.
Use Case Environment Extension	The solutions could be extended beyond the current use case environment in other areas of requirements engineering within Alstom (BT), e.g., for the verification/validation phases and in specific domains such as software development and off-cycle R&D programs
Cyber Physical Systems relations	Railway traction equipment consists of complex hardware and software elements that in their aggregation constitute cyber-physical systems. While customer requirements frequently impact the combination of physical hardware and control systems, it is often not easy to trace these impacts in the early requirements engineering stages. This is largely due to the vast number of customer requirements that need to be analyzed, allocated and further broken down, a task that is essentially done manually today. The requirements often exhibit different levels of detail which means that the specific implications may first be discovered during subsequent design, build/manufacturing and verification stages, the latter including integration testing. With the tools developed in the collaboration it is envisioned that ambiguities, inconsistencies and in-depth implications can be discovered much sooner and remedies before either software or hardware is designed.

3.7 Automated Model Parametrization (Alstom)

Collaboration on: UCS_BT_2

Participant solutions:

1. Active DoE (AVL):AI Solution for creating optimal Design of Experiments (DoE).
2. STGEM (ABO): System Testing using Generative Models. Test generation and prioritization.

Context and goal:

A railway electric traction system transforms electrical energy from the wayside power supply to a resulting mechanical torque at the wheels of the vehicle. Power electronic converters, electric motors and the traction control system are all key elements to accomplish this transformation, generating desired torque and speed at different operating points of the train. Conventional control systems use sensors to continuously adapt the converter's output voltage, current and frequency to control motor operation. However, more advanced control methods allow a virtualization of these sensors by modelling the motor behavior in the controller. The controller itself operates in real time and cannot use the same degree of detailed physics-based models used for motor design, but rather rely on reduced order models. In doing this, an initial set of motor model parameters typically need to be tuned during system integration tests. This high effort manual tuning task would largely benefit from automation. One approach is to aggregate test data in a data driven tuning model for the parameters. This is the main objective for this use case.

Challenges:

The main challenge includes identifying representative measured datasets that can accurately correlate the parameters of the reduced order model in the controller to the actual behavior of the motor. This identification process implies a design of experiment that keeps the number of actual time-consuming duty cycle runs needed to create a representative measured dataset. The criterium is that an ML-aided approach can demonstrate a significant advantage when compared to manual tuning in terms of time and accuracy.

Approach:

The approach has begun by providing a set of training data generated from the detailed physics-based motor design models for a given duty cycle. Different alternatives for ML based solutions from the respective solution providers are trained with this data. Next a new set of data for another drive cycle is generated against which the trained algorithms are to be validated.

Remaining challenges and next steps:

The collaborations in this use case have been initiated. The next steps include the initial validation of the different ML-based solutions, design of experiment to minimize the number of duty cycle runs along with further validation.

Aspect	Analysis Details
Research Challenges	On the test bench, several of the control parameters are manually tuned to get the accurate behavior of the propulsion system components. Examples of such information are rotor flux and speed information. The exact values of these parameters are difficult to measure and changeable with respect to the operating conditions (speed and torque). One of the dependable factors which causes change in the motor parameters is the temperature level. Real time temperature monitoring capabilities are necessary for speeding up the procedure. A sensor-based temperature measurement would yield precise knowledge regarding the machine's thermal state. However, it is technically or economically infeasible to place sensors in the rotor part due to an electric motor's sophisticated internal structure. In similar lines, sensor-based measurement of the stator winding temperature is impaired in case of faults. In addition, sensor functionality may deteriorate during the motor's life cycle. Numerically, internal component temperature estimation can be made through lumped parameter thermal networks (LPTNs), finite element methods (FEMs), but require expertise in choosing model parameters. These models lack physical interpretability as soon as their boundary condition deviates to meet the real-time requirement. As an alternative, machine learning model can achieve high estimation performance with time invariant properties and low computational complexity. Being fitted on measured test bench data and with pre-processing directly it has the potential to solve the challenges. In the above backdrop following research challenges are articulated for the use case 2: ● Identification of reliable dataset to perform exploratory data analysis. ● Identifying the control parameters that can accurately be estimated from external temperature factors like ambient temperature/coolant temperature and electrical characteristics like voltage and current. ● Apply several State-of-the-Art machines learning /deep-learning algorithms to estimate the fitting parameters. ● Identifying parameters for generalization capability in different drive cycles. ● Evaluate and validate the performance of the ML algorithm while utilizing different driving cycles

SotA Related Work	The research challenges of predicting the temperature of electrical machines using MI algorithm has been captured by researchers in recent times. Some notable research work as found: 1. Kirchgässner, W., Wallscheid, O., & Böcker, J. (2020). Estimating electric motor temperatures with deep residual machine learning. IEEE Transactions on Power Electronics, 36(7), 7480-7488. In this article, deep recurrent and convolutional neural networks (NNs) with residual connections are evaluated for predicting temperatures inside permanent magnet synchronous motors. Here, the temperature profile in the stator teeth, winding, and yoke as well as the rotor's permanent magnets are estimated with the available test bench data. Furthermore, with an automated hyperparameter search through Bayesian optimization, lean models with minimal model sizes are presented that exhibited a strong performance estimation. The results were validated through the mean squared error and maximum absolute deviation performance indicators. Finally, learning curves for varying training set sizes and interpretations of model estimates through expected gradients are presented. 2. Kirchgässner, W., Wallscheid, O., & Böcker, J. (2019, June). Empirical evaluation of exponentially weighted moving averages for simple linear thermal modeling of permanent magnet synchronous machines. In 2019 IEEE 28th International Symposium on industrial electronics (ISIE) (pp. 318-323). IEEE. In this work, linear regression is used as an alternative to LPTNs for predicting the temperatures. The test bench measurement collected data are preprocessed with exponentially weighted moving averages. 3. Czerwinski, D., Gęca, J., & Kolano, K. (2021). Machine learning for sensorless temperature estimation of a BLDC motor. Sensors, 21(14), 4655. In this article, the authors propose two models for motor winding temperature estimation using machine learning methods. For the purposes of prediction, measurement data were used. The algorithms such as ElasticNet, stochastic gradient descent regressor, support vector machines, decision trees, and AdaBoost were used for predictive modeling. The ability of the models was to generalize by hyperparameter tuning with the use of cross-validation.

	4. Zhu, Y., Xiao, M., Lu, K., Wu, Z., & Tao, B. (2019). A simplified thermal model and online temperature estimation method of permanent magnet synchronous motors. Applied Sciences, 9(15), 3158. In this article, a five-node LPTN is proposed to predict the motor temperature online. The parameter identification method is based on multiple linear regression on the state equation and Kalman filter algorithm. 5. Zahid, T., Xu, K., Li, W., Li, C., & Li, H. (2018). State of charge estimation for electric vehicle power battery using advanced machine learning algorithm under diversified drive cycles. Energy, 162, 871-882. In this paper, an approach for using subtractive clustering-based neuro-fuzzy systems is presented to define the input parameters to model the battery state of charge. Different drive cycles data are utilized for the training and testing stages of the state of charge estimation model. This algorithm could be useful for use in the traction system, to include performance changes due to various drive-cycle operations.
Generic Requirements	Since the developed AI models for the controller would be utilized on the real time test bench, this generic requirement will be to generate test cases in the real test bench with use of efficient AI algorithms. The collaboration, directly addressing the requirement BT_R02, which is related with the Generic Requirements **GR Mod 02, GT Test 01, GT Test 02, GR Mon 1.2 and GR Mon 1.2.** The solution proposed and defined during the hackathon addresses the requirements in the following way: **GR Mod 02 As** the AI algorithm based tuned parameters are meant to be implemented in the controller for generating the test case scenarios in a real test rig, this generic requirement is considered to have a relationship with the UC requirement. ○ Relationship: +2 **GR Test 01** As setting up pf controller parameters require personally supervised simulation inputs from the physics-based models, that would be benefited from the ML techniques ○ Relationship: +2

GR Test 02 As setting up pf controller parameters require expert supervision to extract the inputs from the physics-based model/ test data, it is time consuming and thus, requires automation of some these procedure

- Relationship: +2

GR Mon 1.1 The AI enabled test platform should have the capability of self-learning from both online and offline test rig data.

- Relationship: +1

GR Mon 1.2: The AI enabled test platform should have the capability of self-learning from both online and offline test rig data

- Relationship: +1

Confirmed collaborations

- **Active DoE (AVL)**, AI Solution for creating optimal Design of Experiments (DoE).
- **STGEM (ABO)**, System Testing using Generative Models. Test generation and prioritization.

Potential Collaborations

- **ESDE (ACO)** – Potential for use of AI for Testing capabilities in the UC
  - Relationship 0
- **Position Monitoring for Industrial Environment (ACO)** – Data Collection Data Management Engagement & Analysis is not perceived as a challenge in the test bench setup
  - Relationship 0
- **ESDE (ACO)** – Potential for use of AI for Testing capabilities
  - Relationship 0
- **Position Monitoring for Industrial Environment (ACO) –** Data Collection Data Management Engagement & Analysis is not perceived as a challenge in the test bench setup
  - Relationship -1
- **DTsynth (AIT)** – AI for Testing and AI for Modeling capabilities potentially can be utilized
  - Relationship 0
- **Kolga (AND) –** Data Collection Computation Capabilities Automation is not perceived the challenge in the test bench setup
  - Relationship -1

- **Cloud expertise (AND) -** Computation Capabilities Data Handling Capabilities Automation is not perceived as a challenge in the test bench setup
    - Relationship -1
- **Infrastructure as Code (IaC) - expertise (AND) –** Storage Capabilities Automation is not perceived as a challenge in the test bench setup
    - Relationship -1
- **devmate (AST)** – Potential for use of AI for Testing capabilities
    - Relationship 0
- **Keptn (DT) - Data** Management Automation is not perceived the challenge in the test bench setup
    - Relationship -2
- **HIB_logAnalyzer (HIB) -** Data Collection Ingestion & Handling AI for Monitoring is not foreseen as the requirement
    - Relationship -2
- **EMF Views (IMTA) -** AI for Modeling, capabilities potentially be utilized
    - Relationship 0
- **ATL (IMTA) – Data** Handling Capabilities Model-Based Capabilities would not be utilized in the test platform
    - Relationship -2
- **a2k-modev (ITI) –** We don’t see the usage of Computation Capabilities Model-Based Capabilities in this implementation
    - Relationship -2
- **a2k-depman (ITI) –** We don’t see the usage of Model-Based Capabilities Engagement & Analysis AI for Modeling
    - Relationship -2
- **a2k-runman (ITI) -** We don’t see the usage of Computation Data Collection Data Management Data Handling Capabilities Ingestion & Handling Engagement & Analysis Automation AI for Monitoring
    - Relationship -2
- **DevOpsML (JKU)** – We don’t see the usage of Model-Based Capabilities for our requirement
    - Relationship -2
- **JSON Schema DSL (or MDE4JSON) (JKU)** - We don’t see the usage of Model-Based Capabilities for our requirement
    - Relationship -2
- **MOMOT (JKU) –** We don’t see the usage of Computation Capabilities Model-Based Capabilities for our UC requirement
    - Relationship -2

	● **AutomationML Modeling (JKU)**– We don’t see the usage of Model-Based Capabilities for our UC requirement ○ Relationship -2 ● **Modeling Process Mining Tool (JKU)** - We don’t see the usage of Model-Based Capabilities for our requirement ○ Relationship -2 ● **GAN-Based Instance Model Generator (JKU) - AI** for Modeling capabilities can potentially be utilized but does not satisfy our UC requirement ○ Relationship 0 ● **pio (PIO) –** Not perceived as potential solution to our requirement ○ Relationship -2 ● **TATAT (PRO)** - Automation is intended to be applied ○ Relationship 0 ● **CRT (QEN) -** I for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **CRTQI (QEN)** - I for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **QEDITOR (QEN -** I for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **QEDITOR (QEN -** AI for Testing is intended to be applied ● **ConvHandler (ROTECH)** - These capabilities are not required to be built-up for this UC requirements ○ Relationship -2 ● **Bridger (ROTECH)**: These capabilities are not required to be built-up for this UC requirements ○ Relationship -2 ● **DataAggregator (ROTECH)**: AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **RELOAD (RISE)** -AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **Deeper (RISE) -** AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0

	● **Modelio (SOFT)**: AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **Constellation (SOFT -Storage Capabilities Automation):** These capabilities are not required to be built-up for this UC requirements ○ Relationship -2 ● **AALpy (TUG)**- AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **S3D (UCAN)** - These capabilities are not required to be built-up for this UC requirements ○ Relationship -2 ● **SoSIM (UCAN)** - These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **UNISS_SOL_03 (UNISS)** - AI for Testing capabilities can potentially be explored for this UC requirement. ○ Relationship 0 ● **HEPSYCODE (UNIVAQ)**: These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **FOCUS (UNIVAQ):** These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **MORGAN (UNIVAQ):** These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **TWIMO (UNIVAQ)**: AI for Testing capabilities potentially be explored. ○ Relationship 0 ● **TemporalEMF (UOC):** These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **AsyncAPI Toolkit (UOC)**: These capabilities are not required to be built-up for this UC requirement ○ Relationship -2 ● **WAPIml (UOC)**- These capabilities are not required to be built-up for this UC requirement ○ Relationship -2

Architecture Component	**IF-AI-FOR-MODELING**: AI/ML based capabilities to support the modelling phase of the system development ○ Relationship: +2 **IF-AI-FOR-TESTING**: AI/ML based capabilities to support the testing phase of the system development ○ Relationship: +2
Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces: ● **Design Space Explorer**: New AI driven optimization algorithms for design space exploration. ○ Relationship +2 ● **AI for Test Model Generation**: AI techniques for analysis and transformation of test models. ○ Relationship +2 ● **Learning Based Testing**: Testing based on model learning and requirements ○ Relationship +2
Data engineering	**The collaboration improves the following data engineering components:** **Automation –** The automation of the test case generation/ tuning of the parameter. ○ Relationship: +2 **AI for Testing –** Since we need to not only create models automatically but test them, preferably with AI methods ○ Relationship: +2
Use Case Environment Extension	The solutions could be extended beyond the current use case environment in other areas of requirements engineering within Alstom (BT), e.g., reducing the test bench hours for determining thermal performance in different components such as for power converters, filters and on the controller for the powertrain drive etc.
Cyber Physical Systems relations	Railway traction test bench deals with several types of drive cycle operation. With manual retuning of the parameters, the control algorithm deals with such test requirements. Since it involves several hardware and software elements that in their aggregation constitute cyber-physical systems. With the tools developed in the collaboration it is envisioned that MI algorithms-based test case generation, automated tuning of the parameter could potentially reduce the associated test time and repetition.

3.8 Using AI and ML for Safety-Critical Systems in the Automotive Domain (ABI)

Collaboration related to all Abinsula UCSs

- UCS1: Functionality verification
- UCS2: Test definition
- UCS3: Compliance verification

Participant solutions:

- From UNISS: UNISS_SOL_01, UNISS_SOL_02, UNISS_SOL_03, UNISS_SOL_04, UNISS_SOL_05
- From INTECS: INT-DET, INT-DEPTH, INT-XAI

Goal:

The Abinsula Case Study presents a virtual rear-view mirror scenario in which multiple cooperative cameras are used to capture the context outside the vehicle, by means of AI-based technology. AI is a recognized innovative technology, but it is still far from being applied in real safety-critical applications, as well as cameras are far from completely replacing the mirrors in a vehicle. This is something allowed only in concept cars and small productions that do not apply the same regulations of large productions. The main goal of the Abinsula Case Study is related to the introduction of AI/ML techniques in the modeling and testing phase of the system development life cycle.

Challenge:

To implement AI based systems that overcome current technological limitations and enable the possibility of freely playing with all the available technology in the development of future cars, we consider two main macro challenges:

1. **ABI-CH1.** To guarantee the predictability of AI-based systems, formal verification with respect to given specifications and guidelines is necessary. This includes also the need for formally verifying any neural network adopted in the system. This challenge is strictly related to ABI UC requirements: ABI_R01, ABI_R02, ABI_R03, ABI_R05.
2. **ABI-CH2.** There is a need for measuring the reliability of the AI and ML algorithms in a humanly interpretable way, in the aforementioned safety-critical context. This challenge is strictly related to ABI UC requirements: ABI_R04, ABI_R11, ABI_R12.

Approach:

In AIDOaRt, Abinsula is collaborating with the University of Sassari and Intecs Solutions to study the adoption of formal methods in the automotive domain, to support the predictability of AI based systems and loosen the current technological limitations and enable the possibility of freely playing with all the available technology in the development of future cars. Details on the adopted approach are reported in the table below.

Aspect	Analysis Details
Research Challenges	To address **ABI-CH1**, Abinsula collaborates with UNISS to investigate the introduction of formal verification, in the automotive context, for reducing inconsistencies during the system development. We intend to leverage on models at higher levels of abstraction which are able to capture systems' properties. These models can then be used to check system-level properties or to explore alternative architectural solutions for the same set of requirements. For this reason, Abinsula has been deeply collaborating since the beginning of the project with the University of Sassari, giving feedback based on Abinsula internal processes to achieve solutions and methodologies that converge toward the need of the use case. Two specific activities, related to this collaboration are reported, in sections 3.9 and 3.10. *Activities in ABI-CH1 are mainly related to Abinsula requirements: ABI_R01, ABI_R02, ABI_R03, ABI_R05.* To address **ABI-CH2**, Abinsula collaborates with Intecs that offers solutions based on deep learning architectures. The activities related to this collaboration involved an analysis of the use case and of its needs to determine the Neural Networks (NNs) to adopt. This analysis guided Intecs in the selection of the technological solutions to focus on. In particular, Intecs decided to invest in solutions for object detection and tracking based as well as on solutions for depth perception, to provide indications of potential hazardous situations. To provide an important hint on what aspects of the objects (vehicles, pedestrians, etc.) are important to be recognized, Intecs is working on a solution that brings the concept of explainability to the deep learning modeling world. This is meant to give an additional validation method, by giving a humanly interpretable reason for the model output. From the implementation point of view, we did a preliminary study of the workflow to be adopted. In particular, we have started considering: the adoption of a Pynq-Z1 FPGA, a preliminary implementation of an ONNX network, and the usage of a workflow that involves Brevitas and FINN tools. However, this required modifications in the ONNX, to provide a FINN-ONNX. Therefore, we moved also to the evaluation of the Xilinx Tool Vitis AI. This activity is currently ongoing. *Activities in ABI-CH2 are mainly related to Abinsula requirements: ABI_R04, ABI_R06, ABI_R07, ABI_R08, ABI_R09, ABI_R10, ABI_R11, ABI_R12.*
SotA Related Work	The emergence of the recent ISO 21434 [1] helps the automotive industry to focus on practice to address cybersecurity in a systematic and consistent way, and together with the ISO 26262 [2], defines the necessary requirements to provide safety and cyber-security in cars. [1] https://www.iso.org/standard/70918.html [2] https://www.iso.org/obp/ui/#iso:std:iso:26262:-2:ed-2:v1:en
Generic Requirements	The collaborations link to the following generic requirements: ● **GR Mod 01**: Use AI techniques for verification of specifications and high-level models. **Relationship: +2**

	● **GR Mod 02**: Use formal models, automated reasoning and/or ML techniques for test generation. **Relationship: +2** ● **GR RE 04**: The Requirement Management Tool of the AIDOaRt Framework verifies the consistency of the requirements. **Relationship: +2** ● **GR Test 01**: AI/ML techniques for test case generation from high level models. **Relationship: +1**
Architecture Component	The collaborations on ABI.CH1 build up the AI for Engagement Analysis, the AI for Requirements, and the AI for Testing architecture components. **Relationship: +2** The collaborations on ABI.CH2 build up the AI for Engagement Analysis and the Explainability architecture components. **Relationship: +2**
Architecture Component Interface	Based on the requirements related to ABI-CH1 there are the following links to the interfaces: ● IF-INSIGHT-ANALYSIS. **Relationship: +1** ● IF-AI-FOR-REQUIREMENTS. **Relationship: +2** ○ AI for Model Consistency Verification. **Relationship: +2** ○ AI for Specification Consistency Verification. **Relationship: +2** ● IF-AI-FOR-TESTING. **Relationship: +1** ○ AI for Test Suite Generation. **Relationship: +1** Based on the requirements related to ABI-CH2, we see the following interface links: ● IF-PREDICTIVE-ANALYSIS. **Relationship: +2** ○ ML-based Object Detection. **Relationship: +2** ● IF-GENSERV-GETING-ANALYSIS-RESULTS. **Relationship: +1** ● IF-GENSERV-ANALYZING-ARTIFACTS. **Relationship: +1**
Data engineering	These collaborations do not contribute to data engineering.
Use Case Environment Extension	These collaborations are perfectly placed in the context of the use case activities and are fundamental for its implementation.
Cyber Physical Systems relations	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 cameras that are meant to replace the rear-view mirror. The system is expected to autonomously react according to the external stimuli and internal needs. With such a context, 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. The combination of new and disruptive technology like AI and ML can enhance the entire development of safety-critical systems and support the prediction of new scenarios that might be considered as safety critical.

3.9 Modeling Property Constraints (ABI)

Collaboration related to all Abinsula UCSs

- UCS1: Functionality verification
- UCS2: Test definition
- UCS3: Compliance verification

Participant solutions: UNISS_SOL_01, UNISS_SOL_04, UNISS_SOL_05 (UNISS)

The context of this challenge is related to the Abinsula macro challenge ABI.CH1 described in Section 3.8, and is related to the introduction of formal verification, in the automotive context, for reducing inconsistencies during the system development. Please, refer to that section for the general context.

Aspect	Analysis Details
Research Challenges	In this challenge, Abinsula and the University of Sassari collaborated on a set of requirements to translate them using a formal semantics. Starting from a set of property constraints of the Abinsula UC expressed in natural language, the main goal of this challenge was to model them by using a specific language with formal semantics, necessary to make use of formal methods to check the consistency of properties. For the modeling language to be adopted, we considered: ● Adoption of a subset of Modelica language [Modelica. ● The CSL4P [CSL4P] language, which resembles the Modelica language. Modelica language allows checking properties in simulation or by formal methods, while CSL4P extends the notion of property to contracts, providing a declarative specification language, and uses only formal methods rather than simulation.
SotA Related Work	[Modelica] P. Fritzson, Principles of Object-Oriented Modeling and Simulation with Modelica, Wiley-IEEE Computer Society Pr, 2003 [CSL4P] Pinto, A., & Sangiovanni Vincentelli, A. L. (2017). CSL4P: A contract specification language for platforms. Systems Engineering, 20(3), 220-234.
Generic Requirements	One of Abinsula requirements (ABI_R01) is related to the verification of specifications and high-level models by using automated reasoning and machine learning techniques. And it refines the GR Mod 01 generic requirement (Use AI techniques for verification of specifications and high-level models). This challenge, being related to the translation of requirements in formal semantics, is the first step necessary to reach automated verification of specifications. **Relationship: +2**
Architecture Component	This collaboration builds up the AI for Engagement Analysis and AI for Requirements architecture components. **Relationship: +1**

Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces: ● IF-AI-FOR-REQUIREMENTS. **Relationship: +1** ● AI for Model Consistency Verification. **Relationship: +1**
Data engineering	This collaboration does not contribute to data engineering.
Use Case Environment Extension	This collaboration is perfectly placed in the context of the use case activities. Indeed, the long-term goal of this use case is to formally verify, at the design stage, the consistency of the system design with respect to some given property constraints, with the purpose to reduce inconsistencies during the system development process. Therefore, the translation of requirements in formal language is a key step of this goal.
Cyber Physical Systems relations	See the "Cyber Physical Systems relation" described in Section 3.8

3.10 Formal Verification of Neural Networks (ABI)

Collaboration on all Abinsula UCSs

- UCS1: Functionality verification
- UCS2: Test definition
- UCS3: Compliance verification

Participant solutions: UNISS_SOL_02 (UNISS), INT-DET, INT-DEPTH (INTECS)

The context of this challenge is related to the Abinsula macro challenge ABI.CH1 described in Section 3.8, and is related to the introduction of formal verification, in the automotive context, for reducing inconsistencies during the system development. Please, refer to that section for the general context.

Aspect	Analysis Details
Research Challenges	In particular, this challenge focuses on the formal verification of Neural Networks (NNs). NNs are one of the most investigated and widely used techniques in Machine Learning. However, despite their success, they still find limited application in safety- and security-critical contexts, wherein assurance about networks behavior must be provided. As an example, a specific concern about the reliability of neural networks is their vulnerability to adversarial attacks, which are small variations of the inputs which cause unforeseeable changes in the behavior of the neural network. During the second AIDOaRt Hackathon we made a preliminary investigation to identify a plausible network architecture that presents satisfactory accuracy and robustness when applied to the Abinsula case study and which is possible to verify leveraging the current state-of-the-art verification tools. The first activities related to this challenge are: ● Preliminary analysis of verification state of the art and identification of supported architectures and expected scalability. ● Preliminary analysis of state of the art of NNs for object detection and identification of best performing models for the task of interest. ● Estimation of necessary trade-offs between best performing models and verifiable ones. ● Investigation of some NNs properties of interest to be verified.
SotA Related Work	Formal verification aims to guarantee that NNs satisfy stated input-output relations and in the last decade several verification methodologies have been proposed for different specifications and architectures. However, the scalability of verification methodology is still far from what would be needed to support state-of-the-art neural networks, whose size and complexity are notoriously high. Indeed, by evaluating results of the 2nd International Verification of Neural Networks Competition [1] (VNN-COMP'21) it appears clear that, even the best of the current verification methodologies are still far from supporting neural networks whose size

	is comparable with, for example, the YOLO architectures [2] which is a family of popular architectures used for object detection tasks. [1] Bak, S., Liu, C., Johnson, T.T.: "The second international verification of neural networks competition (VNN-COMP 2021): Summary and results." – CoRRabs/2109.00498 (2021) [2] Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: "You only look once: Unified, real-time object detection." In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 779–788. IEEE Computer Society (2016)
Generic Requirements	The Abinsula use case includes the use of ML and NN for video elaboration (ABI_R04, ABI_R09, ABI_R10. ABI_R11, ABI_R12). Therefore, one of the Abinsula requirements (ABI_R02) is related to the verification of deep neural networks, and it is directly linked to the GR Mod 01 generic requirement (Use AI techniques for verification of specifications and high-level models). This challenge is the first step necessary to reach automated verification of NNs. **Relationship: +2**
Architecture Component	This collaboration builds up the Engagement & Analysis component. **Relationship: +1**
Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces: ● IF-INSIGHT-ANALYSIS. **Relationship: +1** ● IF-PREDICTIVE-ANALYSIS. **Relationship: +1**
Data engineering	The collaboration does not improve any of the basic data engineering components.
Use Case Environment Extension	This collaboration is perfectly placed in the context of the use case activities. Among the goals of this use case there is the development of models accurate and robust enough to be successfully utilized in the application of interest but, at the same time, limited in complexity to be amenable to formal analysis, which is notoriously expensive in terms of computational resources.
Cyber Physical Systems relations	See the "Cyber Physical Systems relation" described in Section 3.8.

3.11 Architecture modeling patterns (VCE)

Collaboration on VCE_UCS_01, VCE_UCS_02, VCE_UCS_03

The collaboration was founded in the first hackathon around the challenge named **Architecture modeling patterns,** since then it has continuously evolved within the same partnership.

Participant solutions: Modelio (SOFT), AutomationML Modeling (JKU), EMF Views (IMTA), ATL (IMTA), Modeling process mining tool (JKU), MORGAN (UNIVAQ), Keptn (DT), AIDOaRt framework (MDU)

Goal:

The goal of the collaboration can be broken down into several sub-goals. (1) The definition and creation of modeling patterns for system architectures representative for the VCE products. (2) Capabilities of analysis on these architecture models. (3) The efficient creation of these models. (4) The addition of more advanced capabilities into the current workflow of designing and managing the architecture descriptions of VCE systems. These goals are worked towards in parallel by the various partners with much overlap between the partners activities and solutions.

Challenge:

To achieve the defined goals there are a few challenges to overcome. These are challenges related to the capabilities of languages and tooling to enable the creation of standard patterns for the architecture models, as there is a need to capture domain-specific knowledge. Another important challenge is the integration of legacy solutions into the developed solutions in AIDOaRt, as it is not possible to completely change the workflows. It is also necessary to show the improved capabilities in an internal context to demonstrate the added capabilities of the developed solutions inside AIDOaRt.

Approach:

The approach so far has been to utilize standard languages for the definition of architecture models, namely SysML and AutomationML. Views can be created for the models based on the need of the modeler and the model activities. With models defined in these formats the next step is to introduce a recommender system to allow engineers to perform the modeling activities more efficiently in order to correctly create models. To enable the recommender system process mining is employed.

Remaining challenges and next steps:

The current activities are only tackling early challenges. Mainly the next steps regard the more realistic evaluation of these methods in the VCE context, along with clear definitions of the patterns to be developed. So far, the solution can be seen as a simple prototype which lacks a lot of polishing to better match the current needs of VCE. Additionally, the added capabilities of introducing a primarily model-based workflow needs to be better understood and presented. There are many potential avenues to investigate, and one key candidate is the introduction of early simulation capabilities, perhaps utilizing FMI/FMU and co-simulation. There is also a large interest in the use of continuous methods and practices, which could link well with FMI/FMU integration.

Aspect	Analysis details
Research Challenges	Developing large systems, in particular large CPS is difficult. VCE deals with product lines of a large assortment of machines, further maintaining a large array of product variability in each product. To deal with the increasing complexity and needs of advanced capabilities, model-based methods are foreseen to assist and alleviate the current difficulties in management and design. In particular VCE is interested in the models at architecture levels, where current practices are utilizing office tools and other less mature methods. Taking this into account, the following research challenges are (at the very least partially) covered by the hackathon collaboration for VCE_UCS_01: a) Patterns for modeling architectures of complex cyber physical systems considering industrial needs (variability, PLM, interoperability, etc.) b) Standard modeling language support for modeling patterns, aiming to allow interchange between tools (and other languages) and keeping potential solutions scalable. c) Modeling assistance enabling easier learning curves and higher quality models. d) AI capabilities to strengthen modeling workflow. e) DevOps capabilities to strengthen modeling workflow. f) Modeling rules and guidelines to enable added capabilities from AI methods/tools. g) Modeling rules and guidelines to enable added capabilities from DevOps practices/tools. h) Tool support for other identified challenges.
SotA Related Work	When dealing with complex cyber-physical systems, information is very often fragmented across many different models expressed within a variety of (modeling) languages. To provide the relevant information in an appropriate way to different kinds of stakeholders, (parts of) such models have to be combined and potentially revamped by focusing on concerns of particular interest for them. Thus, mechanisms to define and compute views over models are highly needed. Several approaches have already been proposed to provide (semi-)automated support for dealing with such model views. EMF Views, as provided by the IMTA partner in AIDOaRt, is one of them:

1. Hugo Bruneliere, Erik Burger, Jordi Cabot, Manuel Wimmer. A Feature-based Survey of Model View Approaches. Software and Systems Modeling, Springer Verlag, 2019, 18 (3), pp.1931-1952. ⟨10.1007/s10270-017-0622-9⟩.
2. Hugo Bruneliere, Florent Marchand de Kerchove, Gwendal Daniel, Sina Madani, Dimitris Kolovos, et al. Scalable Model Views over Heterogeneous Modeling Technologies and Resources. Software and Systems Modeling, Springer Verlag, 2020, 19 (4), pp.827-851. ⟨10.1007/s10270-020-00794-6⟩.
3. Romina Eramo, Florent Marchand de Kerchove, Maximilien Colange, Michele Tucci, Julien Ouy, Hugo Bruneliere, et al. Model-driven Design-Runtime Interaction in Safety Critical System Development: an Experience Report. The Journal of Object Technology, Chair of Software Engineering, 2019, The 15th European Conference on Modelling Foundations and Applications, 18 (2), pp.1:1-22. ⟨10.5381/jot.2019.18.2.a1⟩.
4. Hugo Bruneliere, Jokin Garcia Perez, Manuel Wimmer, Jordi Cabot. EMF Views: A View Mechanism for Integrating Heterogeneous Models. 34th International Conference on Conceptual Modeling (ER 2015), Oct 2015, Stockholm, Sweden. ⟨10.1007/978-3-319-25264-3_23⟩.

Concerning modeling languages, the VCE challenge is considering the adoption of the OMG SysML (v1.6) as primary general purpose modeling language and AutomationML.

AutomationML is a neutral XML-based data format representing engineering knowledge in process automation and control. It is widely accepted in Industry 4.0 by its development within an academic and industrial consortium. In particular, AutomationML plays the role of an integration format for the following standards: CAEX for system topology, COLLADA for geometry and kinematics, and PLCopen XML for logic. JKU worked on AutomationML integration with OMG SysML since both AutomationML and SysML can fit the purpose of modeling and simulation of complex CPSs.
JKU is working on AutomationML Modeling, a suite of model-driven research tools which is available in a GitHub repository (https://github.com/amlModeling/) based on the AutomationML standard. In particular, it applies Eclipse Modeling Framework (EMF)-based technologies to CAEX (and then AutomationML).
In the following, we list publications concerning the adoption of AutomationML for CPS engineering (typically production systems) adopting MDE techniques and practices:

1. L. Berardinelli et al., 2016. Cross-disciplinary engineering with AutomationML and SysML. at-Automatisierungstechnik, 64(4), pp.253-269.

2. A. Garmendia et al., "Modelling Production System Families with AutomationML," 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020, pp. 1057-1060, DOI: 10.1109/ETFA46521.2020.9211894.
3. M. Wimmer, P. Novák, R. Šindelár, L. Berardinelli, T. Mayerhofer and A. Mazak, "Cardinality-based variability modeling with AutomationML," 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017, pp. 1-4, doi: 10.1109/ETFA.2017.8247711.
4. Berardinelli, L., Mazak, A., Alt, O., Wimmer, M., Kappel, G. (2017). Model-Driven Systems Engineering: Principles and Application in the CPPS Domain. In: Biffl, S., Lüder, A., Gerhard, D. (eds) Multi-Disciplinary Engineering for Cyber-Physical Production Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-56345-9_11
5. L. Berardinelli, E. Maetzler, T. Mayerhofen and M. Wimmer, "Integrating performance modeling in industrial automation through AutomationML and PMIF," 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), 2016, pp. 383-388, doi: 10.1109/INDIN.2016.7819190
6. L. Berardinelli, S. Biffl, E. Maetzler, T. Mayerhofer and M. Wimmer, "Model-based co-evolution of production systems and their libraries with AutomationML," 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), 2015, pp. 1-8, doi: 10.1109/ETFA.2015.7301483.
7. Mayerhofer, T., Wimmer, M., Berardinelli, L., Maetzler, E., & Schmidt, N. (2016). Towards Semantic Integration of Plant Behavior Models with AutomationML's Intermediate Modeling Layer. In GEMOC@ MoDELS (pp. 28-37).

For additional information about AutomationML, we refer the reader to its official web page: www.automationml.org.

The Process Mining Tool (PMT) is an ongoing research effort at JKU, developed in the context of a related EU project (www.lowcomote.eu).

Intelligent modeling assistants have been recently proposed to ease the burden of manual activity during the design of the system. Such systems exploit AI and NLP capabilities to recommend relevant modeling artifacts, e.g., domain concepts, classes, and relationships among the entities. We overview the following methodologies that are related to MORGAN tool:

1. L. Burgueno, R. Clariso, S. Gerard, S. Li, and J. Cabot, “An NLP based architecture for the autocompletion of partial domain models,” in Advanced Information Systems Engineering, M. La Rosa, S. Sadiq, and

	E. Teniente, Eds. Cham: Springer International Publishing, 2021, pp. 91–106. 2. Weyssow, M., Sahraoui, H. & Syriani, E. “Recommending metamodel concepts during modeling activities with pre-trained language models”. Softw Syst Model 21, 1071–1089 (2022). 3. M. Stephan, "Towards a Cognizant Virtual Software Modeling Assistant using Model Clones," 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 2019, pp. 21-24, doi: 10.1109/ICSE-NIER.2019.00014. 4. T. Capuano, H. Sahraoui, B. Frenay, and B. Vanderose. Learning from Code Repositories to Recommend Model Classes. The Journal of Object Technology, 21(3):3:1, 2022 5. T. Kuschke, P. M˙ader, and P. Rempel. Recommending Auto-completions for Software Modeling Activities. In A. Moreira, B. Sch¨atz, J. Gray, A. Vallecillo, and P. Clarke, editors, Model-Driven Engineering Languages and Systems, Lecture Notes in Computer Science, pages 170–186, Berlin, Heidelberg, 2013. Springer. 00015. 6. J. Di Rocco, D. Di Ruscio, C. Di Sipio, P. T. Nguyen, and A. Pierantonio. Memorec: a recommender system for assisting modelers in specifying metamodels. Software and Systems Modeling, pages 1–21, 2022. 7. A. Mora Segura and J. de Lara. Extremo: An Eclipse plugin for modelling and meta-modelling assistance. Science of Computer Programming, 180:71–80, July 2019 Modelio tool: Sende, M., Schranz, M., Prato, G., Brosse, E., Morando, O., & Umlauft, M. (2021). Engineering Swarms of Cyber-Physical Systems with the CPSwarm Workbench. *Journal of Intelligent & Robotic Systems*, *102*(4), 1-18.
Generic Requirements	The VCE use case collaboration impacts all use case scenarios, which in turn impacts all use case requirements. VCE_R01 is relate to the generic requirement GR Mod 01 VCE_R02 is related to GR Mod 06, GR Mon 2, GR Mon 2.5, GR Mon 1, GR Mon 1.2, GR Mon 1.3 VCE_R03 is related to GR Mon 2 and GR Mon 2.8 VCE_R04 is related to GR RE 04 and GR Test 03 VCE_R05 is related to GR Mod 09 VCE_R06 is related to GR Mod 11 VCE_R07 is related to GR Mod 07 **GR Mod 01** Since our collaboration aims in part to utilize AI-techniques for recommender systems utilizing the solutions of several partners, this generic

	requirement will be worked towards directly. We expect that this will continue to evolve across the AIDOaRt project collaboration(s). **GR Mod 06** As stated above we aim to utilize a recommender system, which in part covers this requirement. Further we want to utilize continuous practices to configure and deliver models, and this in part will help deal with the variability management of VCE systems more easily. We expect this to be more guided towards the integration of automation in the more mature stages of the collaboration. **GR Mod 07** It is not entirely clear how the link with the collaboration helps towards this particular requirement. **GR Mod 09** We will be using standard modeling languages during the collaboration (SysML, AutomationML) and customize the languages as necessary to meet the expectations of the industrial needs in the construction equipment domain. This is one of the earlier pain-points that need to be correctly understood and implemented and the activities in this regard will be a focal point and necessary to complete early as many other activities hinge on the at least partial completion of a customized standard language. **GR Mod 11** In order for our collaboration to be fully mature, we require the partial fulfillment of this generic requirement as we want to have an automated chain from requirements to functional models. Therefore, we aim to support it via our collaboration. The implementation of more advanced DevOps practices will be performed gradually but is expected to not meet maturity until the later stages of the project. **GR Test 03** As we want to in part utilize recommender systems and further instantiate various models we will contribute to this generic requirement. It is not enough to generate an architecture but it must also be verified. As this generic requirement is relatively overarching it is expected to be part of the collaboration throughout. **GR RE 04** In our collaboration the origin artifacts are in part requirements, we need to guarantee the consistency for further processes. This is required to be understood and implemented at an early phase of the collaboration. **GR Mon 1, 1.2, 1.3** As we are aiming for, at least, a partial integration with a digital twin solution it is necessary to have a solution that can access all kinds of data for further analysis and processing. This is expected to be worked towards in the later stages of the collaboration, perhaps in conjunction with other partners.

	GR Mon 2, 2.5, 2.8 This is directly coupled to the implementation of a digital twin. This is expected to be worked towards in the later stages of the collaboration, perhaps in conjunction with other partners. Indirect mappings VCE_UCS_01 ● **DTsynth (AIT)** – 0 (Neutral), depends on digital twin progression ● **a2k-modev (ITI)** – -1 (Inclined to reject), since we are already using platform for modeling ● **a2k-depman (ITI)** – 0 (Neutral), depends on collaboration progression ● **JSON Schema DSL (or MDE4JSON) (JKU)** – 0 (Neutral), depends on collaboration progression, currently not relevant Indirect mappings VCE_UCS_02 ● **Kolga (AND)** – +1 Interesting solution as it is not entirely clear how the CI/CD will be implemented ● **Infrastructure as Code (IaC)** - expertise (AND) – -1 (Inclined to reject) does not seem to fit current context ● **devmate (AST)** - +1 (Inclined to accept) seems to fit well with plans. ● **a2k-runman (ITI)** - 0 (Neutral), depends on future development ● **a2k-depman (ITI**) – 0 (Neutral) depends on future development ● **MOMOT (JKU)** – +1 (Inclined to agree) looks interesting for future plans ● **Requirements Ambiguity Checker (MDU**) – 0 (Neutral) depends on the progression of the collaboration. ● **TATAT (PRO)** - +1 (Inclined to accept), could be very useful for testing various models ● **Modelio (SOFT)** - +2 (Accept) as we are already using the tool ● **AALpy (TUG)** – -1 (Inclined to decline), does not seem well suited for current activities ● **UNISS_SOL_01 (UNISS)** – 0 (Neutral), depends on the progression ● **UNISS_SOL_04 (UNISS)** – 0 (Neutral), depends on the progression ● **UNISS_SOL_05 (UNISS)** – 0 (Neutral), the idea of having a solution tailored to standards is attractive, but at the moment it is unclear if it would fit collaboration ● **HEPSYCODE (UNIVAQ)** – -1 (Inclined to decline), not clear from description how it would help Indirect mappings VCE_UCS_03 ● **ESDE (ACO)** – -1 (Incline to decline), does not seem to fit current needs

	● Position Monitoring for Industrial Environment (ACO) – -1 (Inclined to decline), does not seem to fit current needs ● **DTsynth (AIT)** – 0 (Neutral), depends on the progression of the collaboration ● **Kolga (AND)** – 0 (Neutral), could be very useful depending on how the collaboration proceeds ● **Cloud expertise (AND)** – -1 (Inclined to decline), at the moment this does not seem to fit the activities ● **Infrastructure as Code (IaC) expertise (AND**) – 0 (Neutral), could be very useful depending on how the collaboration proceeds ● **HIB_logAnalyzer (HIB)** – 0 (Neutral), depends on the progression ● **EMF Views (IMTA)** – +2 (strong accept) as we are already using the solution ● **INT-DET (INT)** – -1 (Incline to decline), does not seem to fit the collaboration ● **INT-DEPTH (INT)** – -1 (Inclined to decline), does not seem to fit the collaboration ● **a2k-modev (ITI)** - 0 (Neutral), depends on future development ● **a2k-runman (ITI)** - 0 (Neutral), depends on future development ● **a2k-depman (ITI)** - 0 (Neutral), depends on future development ● **DevOpsML (JKU)** – 0 (Neutral), unclear if this can be applied ● JSON Schema DSL (or MDE4JSON) (JKU) – 0 (Neutral), do not see a use for this at the moment but could perhaps be used in the future ● **MOMOT (JKU)** – 0 (Neutral), could be interesting depending on how the collaboration progresses ● **AutomationML Modeling (JKU)** – +2 (accept) as we are already using it in collaboration ● **Modeling Process Mining Tool (JKU)** - +2 (accept) as it is already used in collaboration ● **pio (PIO)** – 0 (Neutral), unclear how it would be applied ● **ConvHandler (ROTECH)** - 0 (Neutral), could be useful but unclear how it would be applied ● **Bridger (ROTECH)** – -1 (Inclined to decline), at the moment does not seem to fit ● **DataAggregator (ROTECH)** – 0 (Neutral), could be interesting but unclear how it would be applied ● **Modelio (SOFT)** - +2 (accept) as we already are using the tool ● **AALpy (TUG)** – -1 (Inclined to decline), not sure how this would be applied in the collaboration

	● **HEPSYCODE (UNIVAQ)** – -1 (Inclined to decline), not sure how this would be applied in the collaboration ● **FOCUS (UNIVAQ)** - 0 (Neutral), unsure how it would be applied ● **MORGAN (UNIVAQ)** – +2 (accept) as we are already using it
Architecture Component	This collaboration does not build up any of the architecture components.

Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces:
	IF-AI-FOR-MODELING
	IF-AI-FOR-MONITORING
	IF-AI-FOR-TESTING
	IF-AI-FOR-REQUIREMENTS-ENGINEERING
	
	In our collaboration we further break these links down as follows:
	IF-AI-FOR-MODELING
	● **AI-based modeling assistant** – used for the recommendation system
	● **AI for instance model Generation** – used for our model instantiation from architecture models
	● **AI for view-model synchronization** – used for managing the different views in the systems
	
	IF-AI-FOR-MONITORING
	● **AI/ML based functional / Performance bug detection -** used for the recommendation and instantiation of models
	
	IF-AI-FOR-TESTING
	● **AI for unit test generation** – used for the verification of instantiated models
	
	IF-AI-FOR-REQUIREMENTS-ENGINEERING
	● **AI for model consistency verification** – Required for the recommendation and instantiation of models

Data engineering	*The collaboration improves the following data engineering components:* ***Automation*** – Since the proposed workflow should be automated and include several key technologies, we improve this aspect of the project. ***AI for Modeling*** – Since we are aiming to implement a recommendation system we improve this aspect, further we also aim to instantiate models based on architecture with the aid of AI techniques. ***Data Management*** – Since there is a lot of data required for the various parts of the collaboration and several different formats and sizes. ***Data Collection*** – Since we need to gather data for recommendation systems and potentially digital twin aspects. ***AI for Monitoring*** – Since we aim to in part have a digital twin implementation. ***Ingestion & Handling*** – Since we require large amounts of data for the various aspects of the collaboration. ***Engagement & Analysis*** – Since we need to provide recommendations and understand what type of models are suited to instantiate. ***AI for Requirements engineering*** – Not entirely clear. ***AI for Testing*** – Since we need to not only create models automatically but test them, preferably with AI methods ***Model-based capabilities*** – Since the collaboration is based on model-based methods and frameworks. ***Storage capabilities*** – Since the foreseen large amount of varied data. ***Data handling capabilities*** – Since the foreseen large amount of varied data.
Use Case Environment Extension	The use case environment firstly defined in the collaboration was somewhat limited as it related mostly to system architecture modeling, and modeling patterns via the challenge **Architecture modeling patterns.** From the starting point of the challenge the context has been expanded in a few different ways as the collaboration has progressed. Going from the original challenge that had a focus on going from informal documents of architecture descriptions to SysML based descriptions of the original non-formal architecture descriptions. Using the results of the first challenge as a starting point the focus of the collaboration has evolved to use the models and artifacts that resulted from the first hackathon for more advanced capabilities and analysis. In particular the collaboration evolved in the following key points from the first to the second hackathon: ● Modeling guidelines to enable more structured formatting of the models in SysML and corresponding data. The extension in this regard is necessary to enable the use and integration of the solutions present in the collaboration which revolve around AI and DevOps capabilities, apart from the modeling capabilities introduced in the first hackathon. ● Introduction of a concrete system to act as an example and baseline for the activities to be produced, with corresponding models and

	assistance of a VCE engineer. The system is believed to encompass the necessary depth and nuance to cover all aspects of the VCE collaboration. ● The addition of artifacts from development in later stages of development, mainly revolving around a simplified version of a Simulink model detailing some control logic of the system under study in the collaboration. ● Refining the scope of the collaboration outcomes to target simulation-based capabilities for analysis. Specifically, how FMI/FMU integration can assist in the analysis of architectural models defined in SysML With this evolution the way forward to realize many of the concrete points of the hackathon challenges are realized. Moving from the second hackathon it is expected that further refinements are to be made and steer the use case towards smaller and more concrete collaborations with the identified partners.
Cyber Physical Systems relations	The VCE use case directly works with CPS as the target systems all are CPS in the form of construction machine aspects, in the case of the collaboration it regards thermal management systems. Thermal management systems need to continuously perform sensing and analysis to provide the correct service in terms of the current status of the system. We are interested in understanding what configurations of a system can meet the governing requirements via various analysis in the early phases of a system architecture design. To this regard we tackle the problem of early analysis of complex CPS.

3.12 Trello Interface for AIOps (HIB)

Collaboration on HIB_UCS1

Management and verification of the requirements

Participant solutions: Modelio (SOFT)

Goal:

The goal of this collaboration is to investigate methods to improve the management of requirements in environments where the activities are organized using a simple, low-tech solution such as Trello (a *kanban*-based task management tool that is used as a shared, online tool).

The goals of the system would be to automatically analyze the text of the new requirements to (a) tag them into a closed set of categories to have a preliminary classification and (b) assign the requirements to the developers (other users of the Trello board) based on their expertise, calculated from past resolved requirements. This is to be done using NLP and other analysis techniques and connected to the Trello board bidirectionally using the provided API.

The tool is used in the context of this Case Study to manage the requirements of the TAMUS system, an integral restaurant management system with hardware and software components.

Challenge:

The challenges in this collaboration are as follows:

- Access to the Trello results, initially on a batch-processing basis and eventually as part of a CI/CD method.
- Analysis of the language of the requirements which is rarely formal and, in many cases, contains jargon and summarizations that might be difficult for typical NLP tool chains.
- Automatic recognition of requirement focus topic and assignment to a number of researchers with limited quantities of data available, as the Use Case deals with an application proposed by an SME in which the total number of such elements might be small for AI analysis.

Approach:

During the first hackathon the following steps were taken:

- Extraction of Trello requirements to CSV lists (using Trello Backup 3.0[2], usage of a dedicated parser based on PyTrello[3] was also explored but would have been slower)
- Analysis performed with the Zapier approach[4].

The results achieved were partially successful. 148 current TAMUS (the restaurant management software used as a basis for the use case) requirements were extracted and analyzed, with limited effectiveness in the analysis with the following highlights:

1. The number of cards is **low**, and so is the number of **similar card titles and stories**,
2. The similarity check based on the title alone yields better results than on the user story or a combination of both (several cards only have a title and no story),
3. The **team assignment is around 50% similar among the top similar cards** (those with 50%+ similarity). Manual labeling/feedback is needed to check if the current labeling is not flexible enough to accommodate more assignment similarities.

Remaining challenges and next steps:
After the limited success in hackathon #1, a new collaboration was established with RISE after hackathon #2 based on recent research and the usage of zero-shot learning looks more promising and will be explored in future research.

Aspect	Analysis Details
Research Challenges	Using Trello for task management is a standard practice in DevOps teams. However, it is a fully manual tool when it comes to assigning persons to specific tasks yet in a full Trello installation there are many information sources of such relationships for past tasks that can be used for suggestions of assignment that can be followed through by the product owner. The covered research challenges are twofold: ● Not a clearly identified language model exists for NLP in Spanish in order to categorize tasks according to topics (expressed as labels in the Trello board). This requires building a new model based on past requirements of TAMUS. ● The assignment of tasks (Trello cards) to particular developers (Trello users) also lacks pre-existing classification models. This is particularly challenging for this task from an SME perspective working on a smaller team and product due to data not being massive (around hundreds of past tasks in

[2] Trello Backup 3.0 : http://www.littlebluemonkey.com/blog/trello-backup-30
[3] PyTrello: https://pypi.org/project/py-trello/
[4] Zapier analysis of Trello boards: https://aylien.com/blog/automatically-tag-trello-cards-with-zapier-and-natural-language-processing

	the Trello board, less than 10 developers). It could require generating extra synthetic data to properly train the model.
SotA Related Work	We used several approaches from the SotA to work on the challenge: (a) → Use of a dedicated Google Sheets extension to extract text in Trello cards to a more usable CSV format (for URL, see footnote 2). (b) → Usage of existing AI approach for Trello Requirements: [SOFT] This approach proposes the creation of a custom data model and its automation with Zapier to extract a predicted tag from a given text. https://aylien.com/blog/automatically-tag-trello-cards-with-zapier-and-natural-language-processing (c) → Usage of zero-shot approach in: W. Alhoshan, L. Zhao, A. Ferrari, and K. J. Letsholo, "A zero-shot learning approach to classifying requirements: A preliminary study," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2022, pp. 52–59.
Generic Requirements	The collaboration, directly addressing the requirement HIB_R02, is derived from the Generic Requirements GR RE 01 and GR RE 03. Regarding GR RE 01 the integration with the solution proposed by SOFT would be able to satisfy the requirement of being able to deal with semi-structured text requirements. Integration in this collaboration would then help satisfy the Generic Requirement that is expected to help in the translation of semi-structured assets that can be used with AI methods. GR RE 03 relates to AIDOaRt requirements management to be able to provide suggestions to the requirements manager. Since this collaboration, if completed, would enable that the requirements are classified and assigned to developers, it would satisfy the generic requirement. The roadmap for integration that will impact the Generic Requirements is derived from: ● GR RE 01: mostly solved as the extraction of data from Trello is complete using the approach in the bullet point a) in the table previous row. ● GR RE 03: further study of the NLP tools to analyze requirements (see bullet point *b)* in the table previous row). Indirectly, we can find relations to: ● ATL (IMTA): -2 (strong reject) as there is no model of the requirements in HIBs side. ● VARA (RISE): +1 (weak confirm) as the similarity analysis is useful for GR RE 01 and the recommendations are useful for GR RE 03.

	● Modelio (SOFT): +1 (weak confirm) relation as the requirements are not envisaged to be modeled in Modelio which is not *per se* a requirements modelling tool, although potentially could be configured as the backend and model support for one. ● UNISS_SOL_01 (UNISS): 0 (neutral) as the consistency of the requirements is not part of the expected results of the collaboration.
Architecture Component	The integration for this collaboration does not directly impact the general architecture components for AIDOaRt. Indirectly we can find links to the following solutions: ● ESDE (ACO): -2 (strong reject). ● Position monitoring (ACO): -2 (strong reject). ● Cloud Expertise, Kolga, IaC (AND): +1 (weak confirm) for now, as we'll be using AND tools for other integrations so a strong relation might be found. ● Devmate (AST): -1 (weak reject) as it is interesting but unrelated to the requirements per se. ● Keptn (DT): -2 (strong reject). ● hib_logAnalyzer: +2 (strong confirm) as both elements need to work together in the UC environment. ● a2k-runman: +1 (weak confirm) as it could be useful in the use case but applicability for requirements is reduced. ● TATAT (PRO): -1 (weak reject). ● VARA (RISE): +2 (strong confirm) for similarity and suggestions. ● ConvHandler (RoTech): -2 (strong reject), no connection found. ● Modelio/Constellation (SOFT): +1 (weak confirm) as no further modeling on requirements under Modelio is followed in the collaboration, but could eventually be. ● UNISS_SOL_01 (UNISS): -1 (weak reject) as consistency in requirements is not part of the collaboration. ● UNISS_SOL_04 (UNISS): -1 (weak reject) as consistency in requirements is not part of the collaboration. ● UNISS_SOL_05 (UNISS): -1 (weak reject) as consistency in requirements is not part of the collaboration. ● TemporalEMF (UOC): -2 (strong reject).
Architecture Component Interface	This collaboration does not impact the interfaces of the Architectural Components.
Data engineering	This collaboration relates to the following architectural components: ● **Data Collection**: It provides new tools for the acquisition of data (from Trello boards) ● **Data Management**: not a clear improvement in this

	AI for Requirements engineering and **Automation**: this collaboration provides new tools for automation of requirements by means of automatically analyzing the requirements in Trello in search of topic and assignment to developers. If achieved eventually, this would be one of the key insights of this collaboration.
Use Case Environment Extension	The development of this would be very useful cross Use Case in AIDOaRt and beyond, as using Trello as an informal backend for requirements and other aspects of development is common practice and any tools that ease the processing of such Trello data with AI would be easily reused in different scenarios. This has evident links with the work by Alstom for Recommendation systems for RE explained in section 3.6.
Cyber Physical Systems relations	This Use Case Scenario refers to requirements of TAMUS that sometimes impact the CPS aspects in the system, such as for examples: - Connection to hardware such as printers (for receipts and kitchen orders), payment devices (card readers) and other such as cash register drawers. - Management of real-life stock of objects in the restaurants (e.g., inventory in the pantry and kitchen). The connection of the collaboration itself is not strong as it refers more to the development process and not to the actual reality where CPS aspects are more prevalent. However, there is an indirect relationship.

3.13 Improving CI/CD in Restaurants (HIB)

Collaboration on HIB_UCS3

Improving CI/CD in UC6 Case scenario

Participant solutions: Kolga (AND), Cloud expertise (AND), IaC expertise (AND)

Goal:

The goal of this collaboration is to establish a first CI/CD baseline for the TAMUS development system used in the Restaurant UC6 of AIDOaRt. The objective is to integrate the basic tools needed to ground a development toolchain that then can be improved with the needs posed by the requirements HIB_R03 and HIB_R04.

Challenge:

The challenge is mostly business-oriented, as the usage of CI/CD approaches in the use case would require starting with a reliable and low-cost solution that can then be extended to fulfill the requirements using AI approaches.

Approach:

In hackathon #1, we studied the development process as well as the final product of TAMUS in order to get new solutions by AND that can be used to build an initial stage CI/CD pipeline. A number of ANDs tools (e.g., Kolga, IaaC) were identified yet the process was stopped for both parties to sign a Non-Disclosure Agreement that protected the IP of TAMUS. This was achieved before the hackathon #2.

Remaining challenges and next steps:

After the hackathon #2, AND decided to withdraw from the collaboration, so now this is in a process of reconfiguration. As noted in the table below, a number of indirect relationships connect the needs of HI Iberia with a number of AIDOaRt's solutions, so we expect to continue using these as the new baseline in the coming period.

Aspect	Analysis Details
Research Challenges	The TAMUS solution used as the basis of UC6 does not initially have any integration with proper CI/CD tools. The research challenge is to enable the adoption of such tools on top of a running code infrastructure that has been so far managed only using manual development tools and practices.
SotA Related Work	The solutions proposed by AND as part of their study on their tools presented in deliverables D1.4, D4.1 and D2.1. ● Kolga.

	● Cloud expertise (used on the EC2 cloud infrastructure used by TAMUS). ● IaC expertise. These tools provide the baseline used in the collaboration to expand the CI/CD capabilities of the TAMUS application. During **Hackathon #2**, the aforementioned AND solutions were analyzed in more detail along with more details on the TAMUS operation in a face-to-face meeting with the HIB and AND teams and it was decided that the match was misidentified during the first Hackathon. Thus, it was decided that new integrations with other solutions would be followed by leveraging the indirect links method that resulted in the solutions mentioned in the following sections of this table.
Generic Requirements	Indirectly, the following are the proposed connections by means of the captured relationships by the generic requirements in Modelio: ● STGEM (ABO): +1 (weak accept) as the testing of artifacts will be part of the activities undertaken for this integration challenge. ● DevOpsML (JKU): -2 (strong reject) as this is not related to the issues under investigation in the Use Case Scenario. ● CRT (QEN): +2 (strong accept) as software testing is an integral part of CI/CD that is not covered in the current TAMUS product. ● CRTQI (QEN): +1 (weak accept) as metrics for CI/CD are contingent to the introduction of basic functionality on these terms into TAMUS ● QEDITOR (QEN): 0 (neutral) as, even if it is integrated with the rest of proposed tools by Qentrinel/Copado, this solves no particular need in the TAMUS UCS3.
Architecture Component	The following are the identified relationships by architecture mapping detected in the Modelio model: ● ESDE (ACO): -2 (strong reject) as the focus of ESDE seems to be very far removed from the needs of TAMUS. ● Position Monitoring for Industrial Environment (ACO): -2 (strong reject), same as above. ● DTsynth (AIT): -1 (weak reject) as the main focus of TAMUS does not require digital twinning for now, although could be useful in the medium-term future. ● devmate (AST): +1 (weak accept) as the integration of testing tools is in scope of the work for the TAMUS use case. ● HIB_logAnalyzer (HIB): -2 (strong reject) as it is not applicable in this context. ● INT-XAI (INT): -2 (strong reject).

	● a2k-runman (ITI): -1 (weak reject) as this is quite far removed from the current CPS edge of the TAMUS application in UCS3. ● TATAT (PRO): +1 (weak accept) as automated testing is essential for the extensions of CI/CD that we propose in this collaboration. ● ConvHandler (ROTECH): -1 (weak reject) as it is apparently out of the scope of this collaboration. ● Bridger (ROTECH): -1 (weak reject) as it is apparently out of the scope of this collaboration. ● Modelio (SOFT): -2 (strong reject). ● Constellation (SOFT): -2 (strong reject). ● AALpy (TUG): -2 (strong reject). ● TemporalEMF (UOC): -2 (strong reject). ● AsyncAPI Toolkit (UOC): -2 (strong reject). ● Keptn (DT): +1 (weak accept) as it presents numerous possibilities for the improvement of CI/CD in TAMUS, although this is a powerful extension that requires a basic deployment of CI/CD tools in the system that we don't have currently.
Architecture Component Interface	This Use Case Scenario is not linked to the development of any architecture component interfaces nor will they be changed in the collaboration.
Data engineering	Not applicable for this collaboration as no elements of data engineering are required for solving the problems presented in UCS3.
Use Case Environment Extension	The proposed integration for HIB_UCS3 is very much a bespoke solution for a problem of our product, but it presents an interesting meta-use case of how a relatively small software application (just over 100 requirements and around 10-12 developers and system managers) with little infiltration of current CI/CD practices can benefit from an offering such as AIDOaRt's. Thus, we will try to document the process of this integration as a possible example of applying the AIDOaRt approach on a system that does not have prior CI/CD or modeling.
Cyber Physical Systems relations	This collaboration is not directly addressing any of the CPS aspects of the UC6 Restaurants (i.e., usage of cash drawers, system printers and waiter terminals). It is thus an auxiliary aspect of development and the connection with CPSs is indirect only.

3.14 PO Classification (CSY)

Collaboration on CSY_UCS_1

PO (proof obligation) classification

Participant solutions: none

Goal:

The goal of this collaboration is to create a first proof of concept to apply Machine Learning on proof problems. We aim to use ML to classify proof obligations that need to be solved in classes that correspond to the tool that can solve them. There are currently three automatic tools that can be used to solve proof obligations and the state of the art is to try every tool one by one.

Challenge:

The challenge is to build an automatic framework able to extract data from dozens of Safety critical Railway projects written in B, create material able to teach ML algorithms and measure whereas this approach can be benefic for new Safety critical proven projects.

The framework has to deal with anonymized proof obligations shared on a GitHub repository, automatically run proving tools on the project to classify PO in several classes, define a way to create vectors from XML proof obligations and use this created labelled material to teach a ML algorithm.

Approach:

The approach is in several steps:

- Run shell and or python scripts on B projects to try proving tools, record if the tried tool solved the PO, collect the results, and benchmark the time passed. This time will be equivalent to the state-of-the-art approach time and will serve as baseline reference time.
- From the records of the benchmark pass, treat data in order to teach a classification tool able to predict what tool can solve a PO.
- Treat all proof obligations of a project with the classification tool and solve them with the predicted solver only, measure the time passed and accuracy, those will be the values for CSY_UCS_1 KPI.

Remaining challenges and next steps:
Only use case provider worked on the challenge during the hackathon and progress on the subject were small. Thanks to informal discussions with AI specialists, there was improvement in the understanding and approach of the problem. Next step is to work on site to implements all the necessary tools and processes.

Aspect	Analysis details
Research Challenges	Challenge identified are: - Deploying a continuous integration for B proof mechanics and keeping it up to date at the lowest cost. - Solving cybersecurity aspect with anonymization of proofs - Choose Machine learning techniques adapted to the topics (xml data, multi-class classification, supervised classification) - Technical choices: scikit and TensorFlow
Used SotA (papers)	The solution proposed by CSY was to do further research in ● XML Data representation ● Multi-class classification ● Supervised classification CSY needed to inspect SotA technics in ML such as: ● Scikit Learn ● TensorFlow
Generic Requirements	This collaboration does not build up any of the generic requirements.
Architecture Component	This collaboration does not build up any of the architecture components.
Architecture Component Interface	This collaboration does not build up any of the architecture component interfaces.
Data engineering	The challenged addressed problem in Data engineering (data management) by better representing XML data in ML vectors
Use Case Environment Extension	The challenged extended the Use case environment by solving problems in DevOps integration; by preventing Gitlab cybersecurity risks while integrating benchmarking in a DevOps solution
Cyber Physical Systems relations	This collaboration directly addresses CPS projects. The improvement in PO classification leads to a diminution of development time in safety critical CPS projects such as Railways projects, interlocking, platform screen doors management.

3.15 Game of Proof (CSY)

Collaboration on CSY_UCS_2, CSY_UCS_3, CSY_UCS_4

Determine a way to train a neural network with proofs created during manual proving.

Participant solutions: none

Goal:

The goal of this challenge is to use determine a way to train a neural network with proofs created during manual proving. After automatic proving, PO have to be solved manually. Manual proof can be compared to winning a game in which we have a state (hypothesis stack and goal) that is changing at each step and we need to find a trace that goes from initial to final state (Goal is true).

We have banks of manual proofs to work on and teach an algorithm, the way to go is not clear for Clearsy.

Challenge:

On the way to completion there are several challenges that are already identified:

- Organize data
- Create pipeline in current proving software.

Approach:

Constructing vectors for proof material xml data

- The XML files seen as generic data; we extract the meaningful features from the PO representation

- Data are XML representation of Hypothesis, XML representation of Goal
- Hypotheses/Goal are mathematical operations, we want to keep the operators in the vector. Example of operation: “a belongs to A”, “P & R”, “Q & not(P)”
- PO are anonymized, label the operands
- For a given PO, only a finite number of operands
- Some of them appears in the goal, there are more important

- We count, we don’t solve! We try to keep most information but not to add some
- We suppose that the main feature that will shape the classification is the form of the goal.
- For solving, the available hypothesis will have importance but especially if they relate to the goal.

Building a model and train it

- Generate a model for cases based on features/vectors

- Check if it fits
- Change model

Improving the model through train iterations by simplifying or complexifying it to reach expectations.

- Only if needed
- Combine models
- Use more data source

Remaining challenges and next steps:

Most of the work is still to be done, we will mostly work on integration with proving tools since this is a blocking step.

Aspect	Analysis details
Research Challenges	Challenge identified are: - Constructing vectors for proof material xml data - Building a model and train it - Improving the model through train iterations by simplifying or complexifying it to reach expectations.
Used SotA (papers)	No external SotA elements were identified for this challenge.
Generic Requirements	This collaboration does not build up any of the generic requirements.
Architecture Component	This collaboration does not build up any of the architecture components.
Architecture Component Interface	This collaboration does not build up any of the architecture component interfaces.
Data engineering	The challenged addressed problem in Data engineering (data management) by better representing XML data in ML vectors
Use Case Environment Extension	This collaboration does not build up any of the environment extension.
Cyber Physical Systems relations	This collaboration directly addresses CPS projects. The improvement in PO classification and solving leads to a diminution of development time in safety critical CPS projects such as Railways projects, interlocking, platform screen doors management.

3.16 Agile process and Electric/Electronic Architecture of a vehicle for professional applications (TEK)

This section will comprise internally three sub-use cases that will be presented in separate Level 3 headings (e.g., 3.16.1, 2 and 3).

Overview of the use case TEK_EAA "Agile process and Electric/Electronic Architecture of a vehicle for professional applications "

In the use case, TEK applies Artificial Intelligence techniques to a Prognostics and Health Management (PHM) system. This system is used for anomaly detection and classification, as well as condition-based predictive maintenance of electric vehicle powertrain, with focus on the traction inverter.

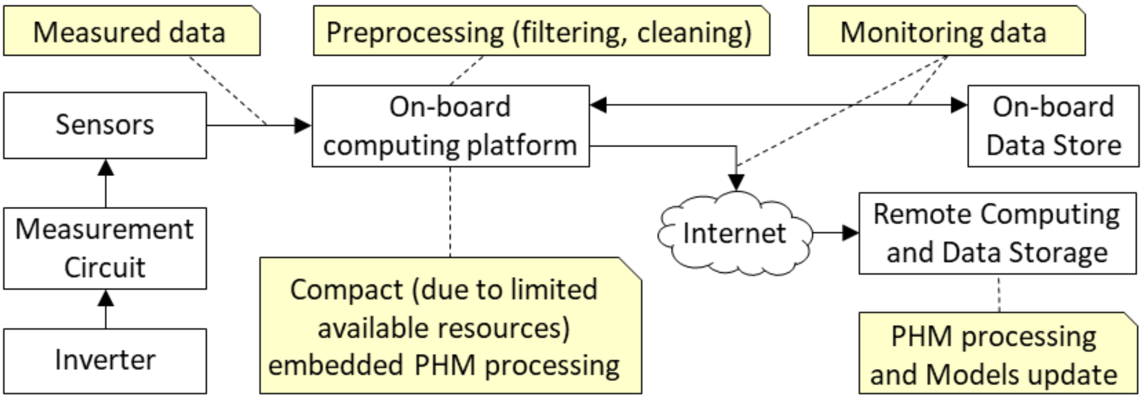

Measured data
Preprocessing (filtering, cleaning)
Monitoring data
Sensors
On-board computing platform
On-board Data Store
Measurement Circuit
Internet
Remote Computing and Data Storage
Inverter
Compact (due to limited available resources) embedded PHM processing
PHM processing and Models update

Figure 14 Schematics of the TEK use case

There are three use case scenarios: TEK_UCS_01 is presented in this section, TEK_UCS_02 and TEK_UCS_03 in the sections that follow.

3.16.1 Design Choices Exploration/Verification (TEK)

Collaboration on TEK_UCS_01

Use case scenario name: *Design Choices Verification*.

Participant solutions: HEPSYCODE (UNIVAQ) and S3D (UCAN).

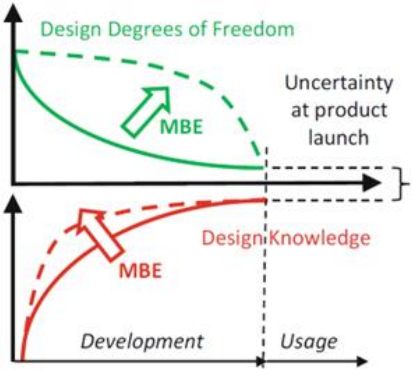
Goal: The goal of the collaboration with UCAN and UNIVAQ is to find the design solutions that best meet requirements and constraints of the compact embedded PHM system that runs on the vehicular computing platform (Figure 14). The design space exploration addresses the code both at low level (mainly HEPSYCODE) and at application level (S3D). The search considers both different hardware and algorithmic alternatives.

Challenge: The collaboration with UNIVAQ began before the first hackathon, the one with UCAN before the second hackathon. After that, while TEK experimented with the tools, UCAN and UNIVAQ analyzed the use case as a testing environment of improvements of S3D and HEPSYCODE they aim at in AIDOaRt. Regarding these aspects, the second hackathon was mainly confirmatory of the tool's functionalities and usability for the use case. The challenge was posed by UCAN and UNIVAQ for a possible integration of their tools that targets the reuse of models and results.

Approach: TEK provided the partners code and models that are representative of the system. The experimentation of integrability and usability in the use case was conducted through remote operation and, with UNIVAQ, in the TEK laboratory. The hackathons were mainly for confirmation and direct discussion.

Remaining challenges and next steps: TEK will provide both models and examples of code which are closer to the prototype of the final system. UCAN and UNIVAQ will contribute by enriching the models and assisting TEK in using the tools. The target for the next milestone (deliverable D5.6) is to obtain predictive results about the structure of low-level code (HEPSYCODE), as well as about time requirements at application level (S3D).

Aspect	Analysis Details
Research Challenges	The use case scenario TEK_UCS_01 deals with design space exploration and the verification of the models. In the deliverable D1.1, TEK gave the requirements that apply to TEK_UCS_01: ● TEK_R_101 — The AIDOaRt Framework verifies, in a semi-automatic manner, at design time, with respect to the requirements, the coverage of the architectural models. ● TEK_R_102 — The AIDOaRt Framework verifies in a semi-automatic manner, at design time, with respect to the requirements, the adequacy (the response versus the resources) of the real components on-

	which/with-which the system architect has in mind to map/realize the architecture. ● TEK_R_103 — The AIDOaRt Framework synthesizes the models needed for the design time verification (defining both the tests and the results, i.e., Pass/fail) in a semi-automatic manner. ● TEK_R_104 — The AIDOaRt Framework interprets the results of the design-time analysis in a semi-automatic manner. *HEPSYCODE* supports many of the activities the system designer carries out to arrive at the final hardware and software implementation, such as dealing with non-functional requirements, specific HW technologies, scheduling policies and inter-process communication. For the first hackathon TEK provided a basic Artificial Neural Network (ANN) for the PHM system, which was trained and validated with simulated data, whilst UNIVAQ completed the system modeling and experimented with timing simulation and the interpretation of results with *HEPSYCODE*. During the second hackathon more complex ANNs were modeled to define the specific *HEPSYCODE* features needed to find the optimal architecture together with the mapping of model elements to real components. During the second hackathon we also confirmed the benefits of using *S3D* which complements the design space exploration of hardware and software components of HEPSYCODE at application level with respect to two aspects. *S3D* provides schedulability analysis and techniques for the validation of timing requirements of concurrent applications. Moreover, the models resulting from the design-time analysis of *S3D* can be adapted by feeding back values obtained at runtime, to refine the target system on the basis of a prototype or in the face of a new enhanced/corrected version.
SotA Related Work	An analysis of the complexity of CPS can be found in [1]: its origins, interrelations, consequences, and approaches to deal with it. For the last aspect, in this use case scenario, we are interested in the "*composition of simulation models reflecting different types of behaviors and concurrency*", which is part of the MBE (Model-Based Engineering) approach. We want to experiment how to deal with several design choices, dependencies among system properties, and trade-offs to be made.  *Figure 15 Cone of uncertainty (adapted from [1])* These issues are consequences of *heterogeneity* of properties, behaviors, and performance targets, as well as of *uncertainty* related to complexity and to a large design space, with risks of potential wrong decisions. Ultimately, we apply *HEPSYCODE* and *S3D* for reshaping the *cone of uncertainty*, by increasing the knowledge at the beginning of the development while at the same time preserving degrees of freedom as illustrated in the Figure 15 above.

	[1] Törngren, M., & Sellgren, U. (2018). Complexity challenges in development of cyber-physical systems. In Principles of Modeling (pp. 478-503). Springer, Cham.
Generic Requirements	The generic requirements that this use case scenario needs (required functions) and that the tools HEPSYCODE and S3D can satisfy are the following (relationship: +2): ● GR Mod 01 — Use AI techniques for verification of specifications and high-level models. ● GR Mod 04 — Use semi-automatic model synthesis for design- and run-time verification. ● GR Mod 06 — Use AI-based methods for easy configuration. ● GR Cod 03 — The AIDOaRt Framework imports/exports variables/procedures between code and models.
Architecture Component	This use case scenario directly requires the functionalities provided by the following AIDOaRt components (relationship: +2): ● Name: Engagement & Analysis — Description: This component supports analysis capabilities (inference, prediction, deduction, etc.) and the collaboration and integration of these services. Other components serve as a means of pursuing the tools' capabilities. Among the latter, the following ones can be employed in the scenario: Design Space Alternatives Exploration, Performance Simulation and Predictions, Model-Driven DevOps approach for HW/SW Co-Design.
Architecture Component Interface	The interfaces that support the most important functions required in the scenario are (relationship: +2): ● IF-INSIGHT-ANALYSIS. ● IF-PREDICTIVE-ANALYSIS. ● ML-based Prediction For Performance and Resource Utilization.
Data engineering	At the moment, no aspect of data engineering needs to be improved.
Use Case Environment Extension	The role that modeling has in the use case scenario grows. Models are needed by the *HEPSYCODE and S3D* tools, to carry out the simulation for predicting the system response in terms of performance and required resources.
Cyber Physical Systems relations	See the row "SotA Related Work".

3.16.2 Runtime verification (TEK)

Collaboration on TEK_UCS_02

Use case scenario name: Runtime verification. Part of use case: TEK_EAA — *Agile process and Electric/Electronic Architecture of a vehicle for professional applications* (see the section 3.16.1)

Participant solutions: devmate (AST).

Goal: Goal of the hackathon was to provide a beta of *devmate* for the C language and test it with code provided by Tekne relating to their use case.

Challenge: Technical challenges lie in parsing the C language due to its special behavior with pointers and in unit-test code generation since there is no default method for testing in C.

Approach: Tekne provided AST a code sample of a machine learning library with example code. AST provided an early version of devmate for the C language as an Eclipse plugin. Due to issues with Eclipse and technical issues our preliminary tests were limited. We verified parsing of the code and identified possible difficulties concerning test code generation.

Remaining challenges and next steps: AST continue development and improvement of devmate for C in close collaboration with Tekne to reach the desired use case requirements.

Aspect	Analysis Details
Research Challenges	TEK gave the motivation of TEK_UCS_02 in the deliverable D1.1: “TEK wants to experiment new tools that support the verification, both during the design phase […] and at runtime.” TEK detailed the above function “support the verification …” with three requirements: ● TEK_R_201 — The AIDOaRt Framework verifies in a semi-automatic manner the implemented software artifact (system, sub-system, component) with respect to the requirements as well as with respects to the architectural and detailed models. ● TEK_R_202 — The AIDOaRt Framework synthesizes, in a semi-automatic manner, the models needed for the verification at runtime (the models that define the tests and the tests results). ● TEK_R_203 — The AIDOaRt Framework interprets, in a semi-automatic manner, the results of the runtime verification. After the first hackathon, AST and TEK analyzed the possible use of *devmate*, a tool for synthesizing, executing, and interpreting the results of structured, readable, and maintainable unit tests. During the second hackathon, AST and TEKNE confirmed their collaboration and discussed necessary adaptations of devmate in order to be used in this scenario (e.g., an IDE extension for Eclipse, C Parser & Code Generator).

SotA Related Work	CPS testing methods and testbeds are a challenging research field due to the increasing heterogeneity, scale, and complexity. [1] Zhou, X., Gou, X., Huang, T., & Yang, S. (2018). Review on testing of cyber physical systems: Methods and testbeds. IEEE Access, 6, 52179-52194.
Generic Requirements	The generic requirements (required functions) that use case scenario TEK_UCS_02 needs and that the tool *devmate* can satisfy are the following (relationship: +2): ● GR Mod 02 — Use formal models, automated reasoning and/or ML techniques for test generation. ● GR Test 04 — Evaluation of testing results. ● GR Mod 01 — Use AI techniques for verification of specifications and high-level models. ● GR Test 02 — Automatic execution of test cases. ● GR Test 01 — AI/ML techniques for test case generation from models.
Architecture Component	The use case scenario TEK_UCS_02 directly requires the functionalities provided by the following AIDOaRt component (relationship: +2). ● Name: AI for Testing — Description: This component enables the employment of AI/ML techniques to support the testing phase of the system development. Other components serve as a means of pursuing *devmate's* capabilities. Among the latter, the following ones can be employed in the scenario: Code Parser, Test-case Evaluation, Testcase Generator, Testcode Generator, Testdata prediction system, Testmodel Editor.
Architecture Component Interface	The interfaces that support the most important functions required in the scenario are (relationship: +2): ● IF-INFRA-STORING-ARTIFACTS. ● IF-INFRA-RETRIEVING-ARTIFACTS. ● IF-MODEL-CODE-GENERATION. ● IF-AI-FOR-TESTING.
Data engineering	The amount and the complexity of the data treated in the use case scenario don't pose any particular challenge. (Relationship: +2).
Use Case Environment Extension	The use case scenario does not change substantially. For a better collaboration, examples will be provided by TEK to support *devmate* modifications/improvements.
Cyber Physical Systems relations	The use case scenario TEK_UCS_02 is about runtime verification. This is still difficult for the class of systems with which CPS have large intersections. Among their distinguishing characteristics there are a very large code base, the common coverage metrics to be employed that are still not clear, and—when the Artificial Intelligence enters the play—how to test deep-learning generated code such as neural networks. *devmate* assists with testing CPS at design time by parsing the code and producing tests based on black box testing enriched with AI techniques, to help find test cases and test values. A future project could investigate possibilities of run-time testing (e.g., by receiving error messages as input and semi-automatically generating test cases).

3.16.3 Operating Life Monitoring (TEK)

Collaboration on TEK_UCS_03

Use case scenario name: *Operating Life Monitoring*. Part of use case: TEK_EAA — Agile process and Electric/Electronic Architecture of a vehicle for professional applications (see the section 3.16.1)

Participant solutions: ROTECH solutions ConvHandler (for data cleaning), Bridger (for ciphering and transmission), DataAggregator (for storage and retrieval).

Goal: The use case targets a Prognostics and Health Management (PHM) system: anomalies detection and classification, and condition-based predictive maintenance of electric vehicle power electronics. The collaboration is on the complete data chain after the on-board sensing: cleaning, transmission, storage and retrieval, taking into account the requirements that result from / relate to the amount of data, security of data, characteristics of the transmission link, as well as the appropriate choice of the database management system with regard to the needs of AI algorithms.

Challenge: A PHM system poses two groups of challenges. The first one regards information and communication technologies. It has aspects related to high level architecture and components that pertain to the use case scenario TEK_UCS_03 and are delineated above, aspects of low level that are addressed in TEK_UCS_01, and aspects of optimization of the development cycle (TEK_UCS_01 and TEK_UCS_02). The second group of challenges is specific to the application field: features of the data—the failure indicators—must be identified, effective data-driven models must be chosen, and large datasets must be collected.

Approach: ROTECH and TEK work in parallel on two development-oriented replicas of the target system that are updated while the project progresses. The first one, which is in TEKNE's laboratory, is intended for integration and verification and will become the final target system. The second one which uses simulated or recorded data, is used by ROTECH to test components that it develops. Apart from the hackathons, there are integration sessions hosted in TEKNE's laboratory.

Remaining challenges and next steps: The applicability analysis has been concluded. Now we are in the first development iteration whose results will be reported in the next D5.6 and D5.7 deliverables. We are interested mainly in the amount of processed data and the required throughput. The second and final iteration, which will be devoted mainly to quality aspects, will complete the development, carry out the verification, validate the project outcome through KPI measurements, and report results in deliverables D5.8. and D5.9.

Aspect	Analysis Details
Research Challenges	In this scenario, the demonstrator operates and provides its services in a partly simulated environment. As shown in Figure 14, measured data are collected and

	pre-processed (cleaning, filtering, feature extraction) to obtain the monitoring data. These are processed by the compact on-board PHM (Prognostics and Health Management) system. Moreover, they are transferred to the remote computing and data storage platform, whose resources are sufficient to run a PHM system with full capabilities. This was synthesized in the following requirement: ● TEK_R_301 — The state of health of the system is interpreted in a semi-automatic manner on the basis of the data produced by the system. The requirement can be satisfied by a combination of capabilities: those of the demonstrator software to be developed and those of the AIDOaRt solutions (tools and components).
SotA Related Work	According to [1] "*Machine Learning (ML) methods have been emerged as a promising tool in Predictive Maintenance* (PdM) applications*". The use case narrows this very wide context as it considers maintenance of electric vehicles focusing on components for power electronics, and applies supervised anomaly detection as ML methods. The use case deals with challenges and research directions [2] that can be summarized as "need of data". Anomaly detection models are trained on datasets that must be labeled and must both represent normal conditions and contain failure precursors. Moreover, there is the *concept drift* phenomenon [2] which occurs when the datasets on which ML models were trained are not representative throughout the operational life of a system due to a change in how the latter are used. An option can be to continuously update the models using datasets acquired from the entire population of vehicles and during their entire life. To deal with this, the use case proposes a light on-board processing together with a remote update of models based on datasets that vehicles collect and transmit (see Figure 14). [1] Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. D. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024. [2] Theissler, A., Pérez-Velázquez, J., Kettelgerdes, M., & Elger, G. (2021). Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. Reliability engineering & system safety, 215, 107864. * PdM (Predictive Maintenance), together with the equivalent terms PHM (Prognostics and Health Management) and CBM (Condition-Based Maintenance), refers to technologies for predicting when maintenance actions are necessary to maintain a system in its operational state. The goal is to overcome the drawbacks of both corrective maintenance (to repair after the failure, during which the system is unavailable) and preventive maintenance (based on system usage given by time and other parameters, e.g., mileage, without considering the system actual status).
Generic Requirements	The general requirements of this use case scenario which are provided by the solutions ConvHandler, Bridger, and DataAggregator can satisfy are the following (relationship: +2):

	● GR Cod 03 — The AIDOaRt Framework imports/exports variables/procedures between code and models. ● GR Mon 1 — Monitoring in AIDOaRt is able to access data, system interfaces, or artifacts. ● GR Mon 1.3 — Monitoring in AIDOaRt is able to access system logs and execution traces.
Architecture Component	The use case scenario employs the functionalities provided by the following AIDOaRt components (relationship: +2): ● Name: Data Management — Description: This component supports cleaning, analyzing, and managing data coming from different sources. Including: (a) means to filter and harmonize data coming from different sources; (b) transform data collected in the DATA COLLECTION component to the internal representation defined in the DATA REPRESENTATION component; and (c) filter and aggregate data. ● Name: Storage Capabilities — Description: The Storage Capabilities component supports the provisioning of physical and logical resources allowing to efficiently store and retrieve the possibly numerous and large data artifacts and models. ● Name: Data Handling Capabilities — Description: The Data Handling Capabilities component supports the loading, navigation, querying and then the saving of the required data.
Architecture Component Interface	The interfaces that support the most important functions required in the scenario are (relationship: +2): ● IF-DATA-FILTERING-AGGREGATION. ● IF-DATA-TRANSFORMATION. ● IF-INFRA-STORING-ARTIFACTS. ● IF-INFRA-RETRIEVING-ARTIFACTS. ● IF-DATA-LOADING. ● IF-DATA-NAVIGATION. ● IF-DATA-QUERYING. ● IF-DATA-SAVING.
Data engineering	The closest collaboration is on the Data Management component for what regards data cleaning and analysis (relationship: +2).
Use Case Environment Extension	The collaboration does not require any changes to the use case environment.
Cyber Physical Systems relations	Aspects that the scenario TEK_UCS_03 “Operating Life Monitoring” adds to the use case are among the distinguishing ones of CPS. There is *distributed sensing* that is networked to achieve *central processing*. There are *uncertainties*, which can be resolved by updating ML models that drift over the course of the operational life.

3.17 Power-aware radar configuration (CAMEA)

Collaboration on CAM_UCS_3

- UCS3: Compliance verification

Participant solutions: STGEM (ABO)

Goal:

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

Challenge:

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

Approach:

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

Remaining challenges and next steps:

In the upcoming period, the configuration generation approach will be tested. The data collected from the radar needs to be stored in a certain format and then analyzed and evaluated by suitable metrics (that are still to be defined). Then, an iterative process will be employed to evaluate as many configurations as possible. Also, some guidance for configuration adjustment based on already tested configurations could be good optimization.

Aspect	Analysis Details
Research Challenges	The configuration for the radar-on-chip is quite complex and some of its parameters can have an influence on power consumption and heat dissipation of the radar platform. There are also some constraints on the configuration parameters that need to be fulfilled and some requirements pertaining to the environment sensing properties (e.g., maximum range, range resolution, angular resolution, and velocity resolution) that need to be met. The STGEM solution proposed by ABO based on AI-based methods can be used for power-aware radar configuration and it can possibly generate much more interesting combinations of configuration parameters that minimize power consumption while providing the same level of performance.
SotA Related Work	There is no such solution related to radar configuration parameters tuning. We are adopting solutions from other fields. We use search-based optimization techniques, which efficiently explore the search-space of possible configurations and allow us to find near-optimal solutions in an efficient manner. [1] Harman et al. (2012) Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys 45.*
Generic Requirements	CAM_R02 is related to GR Mod 06. **GR Mod 06:** This requirement aims for AI for easy configuration. In this case, the configuration is more oriented towards meeting low power requirements, reduction of heat dissipation or fitting algorithms to the target platform. As AI-based methods should be used for optimization of system configuration, it directly contributes to the AI/ML dimension. The STGEM Test Generation and Prioritization service from ABO addresses the following AIDOaRt Generic Requirements that will be reflected during the whole iteration process of generation, testing and analysis of the radar configuration. Modeling: GR Mod 02, GR Mod 06 Testing: GR Test 01, GR Test 02, GR Test 04, GR Test 06
Architecture Component	The collaboration impacts the following architecture components of the AIDOaRt framework related to the “AI-Augmented Tool Set” and the “Data Engineering Tool Set”: ● **AI for Testing:** STGEM uses AI/ML techniques to automate and improve test generation, test selection, and test scheduling and execution components. (Relationship: +2) ● **Data Collection:** STGEM contributes to the data collection tasks aimed at improving the infrastructure for continuous integration and continuous development pipelines and at improving the testing process. (Relationship: +1) ● **Data Representation:** STGEM also contributes to the data representation tasks aimed at improving the infrastructure for continuous integration and continuous development pipelines and at improving the testing process. (Relationship: +1)

Architecture Component Interface	The collaboration impacts the following architecture component interfaces: ● **IF-AI-FOR-TESTING**: The functional interface would offer our use case the capability to use AI techniques for generation of suitable candidates for improved radar configurations. It will allow optimization of configuration parameters that affect the performance of low-power devices. The AI algorithms from this functional interface can explore the space of possible configurations efficiently and find configurations that minimize power consumption while maintaining the required performance of the device. (Relationship: +2) ● **AI for Test Suite Generation**: The functional interface would offer our use case the capability to automate test suite generation (test input selection, test scheduling, and oracle synthesis). The AI algorithms from this functional interface use machine learning algorithms to train both a test generator and a surrogate model of the system under test. (Relationship: +2) ● **Learning Based Testing**: The functional interface would offer our use case the capability to create machine learning models for both online and offline testing scenarios by using adaptive learning and supervised learning algorithms. (Relationship: +1) ● **AI for Test Reduction:** The functional interface would offer our use case the capability to use AI techniques for test case selection, prioritization, and reduction. (Relationship: +1)
Data engineering	The amount and the complexity of the data treated in the use case scenario does pose some data engineering challenges. There are two aspects related to data engineering: ● Data Collection ● Data Representation
Use Case Environment Extension	After some changes in CAMEA's radar API with additions for configuration, tuning, and monitoring, the STGEM solution can be integrated into the radar setup tool. The tool can replace the existing manual radar tuning system and can potentially result in an extension of the CAMEA use case.
Cyber Physical Systems relations	This collaboration is about configuration and monitoring of a physical smart radar device. A device configuration is first generated/modified in the supporting SW, then checked for its validity and sent to the real physical device. Measurements (monitor data) for the physical device are then periodically sent back and analyzed.

3.18 Real Driver Emission (AVL)

Collaboration on AVL_RDE_UCS1, AVL_RDE_UCS2, AVL_RDE_UCS3

Participant solutions: AALpy (TUG), TWIMO (UNIVAQ + MDU)

Goal: The goal of the use case is to estimate emissions / energy consumption of a vehicle driving along an arbitrary route under realistic driving conditions. The problem can be split into two components:

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

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

Challenge: The velocity profile of the vehicle is determined by the driver behavior which is influenced by static and dynamic aspects of the environment such as road geometry and traffic. While the static components are known, the dynamic situation on the road is not. In addition to that, there are also internal factors at play (at least for human drivers), which are also unknown. Those dependencies on unknown variables make the problem inherently non-deterministic. Moreover, those unknowns lead to vastly different behaviors. Due to this, the problem cannot be solved by simply fitting a function to the available data. Take for example a road with a traffic light. If the light is green, the car will drive at a near constant velocity throughout the measurement cycle. If the light is red, it will stop, wait and accelerate again. The average of both behaviors (which would be achieved by naive fitting) will probably never be observed.

Approach: A potential remedy is to use probabilistic models. In this framework, the driver model is a probabilistic agent acting on known environment stimuli. TUG follows this approach by applying passive automata learning techniques to the measurement data collected from test drives by AVL in order to infer models in the form of Markov Decision Processes (MDPs). In short, an MDP is an automaton with a finite number of states, where each state emits an output symbol when active and input symbols trigger probabilistic state changes. The discrete nature of this type of model is in contrast to the continuous space of the problem at hand. Thus, some sort of abstraction is needed to apply MDP learning to the problem. The same abstraction can then also be used to create a similar model of the target track. Together, the driver and track models form a closed system (i.e., without inputs). This combined model can then be used to:

- Extract possible driver behavior by sampling from the model.
- Test assumptions about driver behavior using model checking.

An outline of this methodology is illustrated in Figure 17.

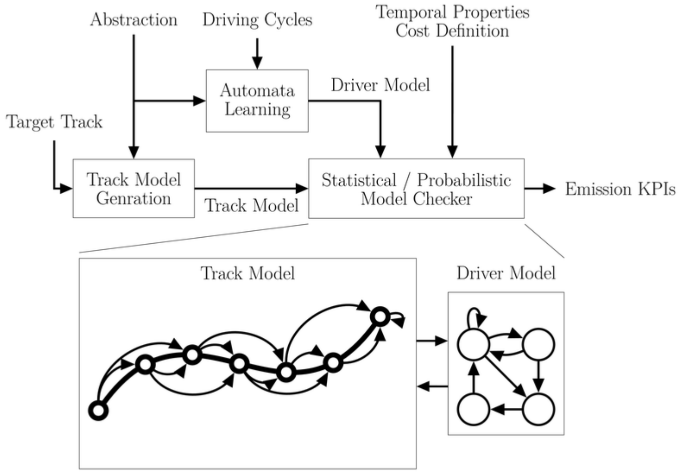


Figure 17 Preliminary methodology

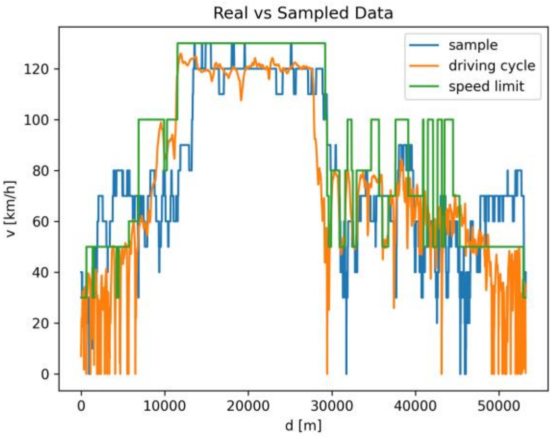


Figure 16 Sample drawn from a learned model

Current Status: The basis of this approach has already been implemented by TUG using their automata learning library AALpy. First experiments with simple abstractions show coarse but promising preliminary results.

One of the experiments is summarized in the following:

- Training Data: 14 driving cycles (ranging between 5 and 150 minutes)
- Abstraction: discretization of measurement data
  - Inputs: curvature and speed limit discretized to three and six levels respectively
  - Output: velocity discretized using 10km/h steps.
- Size of the resulting driver model: 177 states and 871 transitions

A sample drawn from the resulting model using a track model derived from one of the provided driving cycles is shown in Figure 16. The Figure also contains the velocity profile recorded during the test drive and the speed limit at the current position.

Remaining challenges and next steps: One of the problems is that the learned driver model does not accept all inputs in every state. This is to be expected, since for some situations there simply is little to no data (e.g., sharp curves at high velocity or standstills on autobahn). However, this can also occur when sampling from the model when the random choices during sampling happen to steer the model into a state that is uncharacteristic for the given situation on the road. In such a case, the model might not be able to produce any output.

One way to deal with this is to prune the joint model by removing states from which the end of the track is not reachable. However, this method requires enumerating all states of the joint model which may become a problem when pairing large driver models with long tracks.

The problem also occurs if there is not enough data to learn from. In this case, a possible solution is to extend the transitions of states in which are missing some inputs by extrapolating from the behavior of similar states. In this case, the measure of similarity should probably incorporate both the abstraction used as well as the future of the states in question.

Another problem is model quality. The definition of model quality represents a difficult question in and on itself since the focus is on model characteristics rather than specific behaviors. However, even without a well-defined quality metric, the models generated so far are too coarse and also exhibit too much behavior that (at least by visual inspection) seems implausible.

To some extent, this problem can be solved by exploring alternative abstractions and providing more data to learn from. To get more fine-grained results, one can choose a weaker abstraction. In essence, this means taking more aspects of the measurement data into account for example by adding another parameter or increasing the discretization resolution of a parameter or output. However, weaker abstractions also require more data to learn the corresponding model, simply because the model has to represent the real world in more detail. Note the connection to the previous problem.

We also aim to extend automata learning to continuous systems in a more direct way by combining it with other machine learning approaches.

Aspect	Analysis Details
Research Challenges	Car developers have to perform real driving tests in different environments (simulation, Hardware-in-the-Loop (HiL), testbeds) for which they need the driver model capable of reproducing the behavior of a human driver as accurately as possible. The driver model developed in these UCs should perform an arbitrary look ahead for foreseeing the next N steps in the route based on e.g., the speed limit signs, curvature, or stops. Taking this into account, the following research challenges are covered by the hackathon challenge: ● Generation of human behavior in terms of vehicle speed for an arbitrary driving route. ● Modeling of the traffic situation, special case highway. ● Data analysis for deterministic traffic conditions (e.g., traffic signs) ● Clustering relevant driving features such as highway driving, low-speed driving, cornering, braking, acceleration, etc.
SotA Related Work	Extensive research has been performed on the field of modeling human driver behavior. However, research in this area is generally assumes knowledge of the dynamics of the driving situation (e.g., other traffic participants) and commonly aims to model both lateral and longitudinal behavior (i.e., steering and velocity), whereas in this use case only static information is available and only longitudinal information is required. Some of the publications in the field use similar types of behavioral models (Markov Chains and Markov Decision Processes) to model human driver behavior or roads [1,2], albeit in different ways and using different methods.

	Regarding automata learning, TUG uses IOAlergia [3]. [1] Wang et al. (2014) Modeling and Recognizing Driver Behavior Based on Driving Data: A Survey. *Mathematical Problems in Engineering* [2] Sadigh et al. (2014) Data-driven probabilistic modeling and verification of human driver behavior. *AAAI Spring Symposium Series*. [3] Mao et al. (2012) Learning Markov Decision Processes for Model Checking. *Proceedings of the first workshop on quantities in formal methods (QFM)*
Generic Requirements	The AVL_RDE touches following generic requirements: ● AVL_RDE_R01 is relate to the generic requirement GR Mod 04 ● AVL_RDE_R02 is relate to the generic requirement GR Mod 04 ● AVL_RDE_R03 is relate to the generic requirement GR Mod 05 ● AVL_RDE_R04 is not tackled by the hackathon as depends on mature state of the solutions from other requirements. ● AVL_RDE_R05 is relate to the generic requirement GR Mod 06 **GR Mod 04** Since in our collaboration we aim to develop behavioral models of humans and environments, this generic requirement will be directly addressed. We expect that this will continue to evolve across the AIDOaRt project collaboration(s). **GR Mod 05** Nondeterministic behavior of the environment will be modeled either implicitly by modeling the data distribution as proposed by UNIVAQ+MDU or explicitly with a Markov Decision Process as an integrated solution of TUG. **GR Mod 06** Basic driving attributes will be statistically modeled from the data in order to configure driver attributes, including: acceleration / deceleration preferences, braking behavior, cornering behavior or max speed preference. Indirect mappings AVL_RDE_UCS1 ● **DTsynth (AIT)** – not followed due to limited resources of solution provider ● **HIB_logAnalyzer (HIB)** – not an NLP problem, adaptation to UC would be challenging ● **EMF Views (IMTA)** – UCS do not fit to viewpoint challenge ● **a2k-runman (ITI)** – UCS do not fit into monitoring of the process and detecting critical behavior ● **a2k-depman (ITI)** – UCS do not fit to optimization of system design and usage of resources ● **MOMOT (JKU) –** UCS do not fit into optimization of one or more conflicting quality criteria ● **pio (PIO)** – unfollowed lead occupancy of solution providers

	• **Modelio (SOFT)** – utilization would require significant modification of the original solution • **AALpy (TUG)** – explored by Hackathon • **HEPSYCODE (UNIVAQ)** – explored in Hackathon Indirect mappings AVL_RDE_UCS2 • **a2k-runman (ITI)** – UCS do not fit into monitoring of the process and detecting critical behavior • **HEPSYCODE (UNIVAQ)** – explored by Hackathon • **FOCUS (UNIVAQ)** – explored by Hackathon • **MORGAN (UNIVAQ)** – explored by Hackathon Indirect mappings AVL_RDE_UCS3 • **DTsynth (AIT)** – not followed due to limited resources of solution provider • **EMF Views (IMTA)** – UCS do not fit to viewpoint challenge • **a2k-depman (ITI)** – UCS do not fit to optimization of system design and usage of resources • **MOMOT (JKU) –** UCS do not fit into optimization of one or more conflicting quality criteria • **Modelio (SOFT)** – utilization would require significant modification of the original solution • **AALpy (TUG)** – explored in Hackathon • **HEPSYCODE (UNIVAQ)** – explored in Hackathon Indirect mappings AVL_RDE_UCS4 • **Position Monitoring for Industrial Environment (ACO)** • **a2k-runman (ITI)** – UCS do not fit into monitoring of the process and detecting critical behavior • **a2k-depman (ITI)** – UCS do not fit to optimization of system design and usage of resources • **UNISS_SOL_04 (UNISS)** – UCS4 is not related to Domain System Language • **HEPSYCODE (UNIVAQ)** – explored in Hackathon • **FOCUS (UNIVAQ)** – explored in Hackathon • **MORGAN (UNIVAQ)** – explored in Hackathon
Architecture Component	AI for Modeling (+2): The relation to this component is apparent from the use case description. The contribution of TUG to the use case relates to • Explainability (+1): The resulting model can be model checked, which allows to define and verify properties of the learned model.

	● Engagement and Analysis (+2): Data in the form of driving cycles is analyzed and transformed into a predictive behavioral model of human driving behavior. ● Automation (+1): Once a suitable abstraction of the recorded data is found, the method can be probably applied to other data sets to create models of different driver styles (e.g., to accommodate for regional differences in driving behavior) without major changes. The following architecture components are involved but unaffected by the collaboration: Ingestion & Handling, Data Handling Capabilities, Model-Based Capabilities and Data Management.
Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces: ● IF-DESIGN-TIME-DATA-COLLECTION (+1): AVL has collected driving cycles to support model creation. ● IF-FILTERING-HARMONIZATION (+1): The collected driving cycles are from different sources, contain different data fields, are sampled at different frequencies and, thus, need to be harmonized. ● IF-PREDICTIVE-ANALYSIS (+2): A predictive model is learned from the data provided. ● IF-AI-FOR-MODELING (+2): See above. Other interfaces are involved but not affected.
Data engineering	The collaboration does not improve any of the basic data engineering components.
Use Case Environment Extension	During the collaboration, several possible extensions of the use case environment were considered in order to support the improve the accuracy of the driver model: ● Extension of AVL's tools to acquire data from additional sources such as public GPS tracks and related data sets used in the research community. ● Extension of AVL's tools to enhance data quality and provide additional data features such as region information or road properties. ● Use of synthetic data gathered with a driving simulator. ● Extend the use case environment to compare various driver models (based on different methodologies) regarding their accuracy and performance.
Cyber Physical Systems relations	One challenge that comes along with CPS is the large, partially unknown input/output space of such systems, which prevents automated testing in the context of an efficient DevOps setup. A promising method is reducing test space complexity by providing a model of the CPS environment focusing on a particular purpose. Meaningful discretization of their continuous state space is one way of doing so. However, obtaining such a model in the form of a hand-crafted one done by a domain

	expert is an expensive task. Therefore, the introduction of learning methods to automatically populate such models is intended to mitigate these costs. Until recently, the field of automata learning (which is the selected solution method in this use case) has been predominantly focused on discrete systems. However, while discretization reduces the space, it also comes at the cost of potential oversimplification and inaccuracy. Thus, the integration of automata learning into the RDE Driver model use case explores how automata learning can be combined with continuous machine learning methods in ways that exploit the strengths and mitigate the weak points of the respective methods. A challenge specific to the use case is to learn the characteristics of probabilistic / non-deterministic behavior rather than learning a function that best fits a given data set. The question of how to assess model quality in such a setting is also part of this challenge. The task posed by the challenge is thus related to CPS in two main ways: ● The model-estimation methods developed for the RDE use case can be used to learn the input system for a given CPS from existing data, which can be used to aid requirements engineering and automate the testing process. ● Human driver behavior which is interacting with a vehicle (i.e., a CPS) deals with classical CPS challenges such as uncertainty and the interplay of discrete and continuous behavior. Thus, methods developed in the collaboration should contribute to handling and simplifying the challenges of large input spaces for CPS in the context of vehicle testing.

3.19 Test Case Verification (AVL)

Collaboration on AVL_TCV_UCS1, AVL_TCV_UCS2

Participant solutions: *DTsynth (AIT)*

Goal: Since Hardware-in-the-loop (HIL) or vehicle test execution is very expensive, the goal of the use case is to validate whether ADAS/AV test cases that cover all critical situations are selected while the non-critical cases are excluded.

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

Approach: The proposed solution involves an automated fundamental analysis of the test case in order to be certain that no critical scenarios are accidentally discarded. The input given is an OpenScenario 1.1 file, which is a traffic scenario description file, that serves as a template for the individual test cases. The scenario file is used to extract all the relevant high-level information: number of vehicles and vulnerable road users (VRUs), their behavior, and the path that the vehicle under test (VUT) will undergo.

Given the desired paths of all actors (including the VUT), critical regions in the map are identified. For safety related ADAS functionalities (e.g., Automatic Emergency Braking), a critical region would be a part of the map where the vehicle could collide with another vehicle or a VRU. Since the behavior of all other participants is controlled by the simulator, it is possible to calculate a set of critical times *a piori*, when other actors are within the identified critical areas.

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

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

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

However, as is natural during any solution development, further improvements have been identified and are underway. Crucially, if the parameter reduction is done considering the full ODD, then there are still many sampled scenarios that do not test the ADAS function in a meaningful way. Therefore, it has been proposed to create overlapping "slices" of the ODD. The ranges must be overlapping to guarantee that no meaningful test cases are discarded. This improved methodology reduced the parameter space to approximately 10% in comparison to the original scenario.

Finally, a preliminary methodology has been developed to handle the composition of scenarios in order to test more complicated traffic situations. This acts in a modular way with the current base solution and does not add further complexity.

Remaining challenges and next steps: Effort is currently ongoing to integrate the proposed solution into the AVL simulation ecosystem. Afterwards, the solution must be broadly tested to guarantee that no critical test cases are omitted, and benchmarked to showcase the efficiency gains in practice. Furthermore, the work related to active sampling to find a "best" set of parameters is still pending the correct integration of the software.

Aspect	Analysis Details
Research Challenges	The AVL SCENIUS Test Case Generator has simple means to generate ADAS test cases from abstract scenarios. Namely, thousands of test cases can be generated by instancing one given scenario - like an overtake maneuver on a highway - with a concrete value for vehicle speeds, distance, etc. of all actors in the scenario. Since HIL or vehicle test execution is very expensive, it is crucial to select a subset of test cases, sufficient to cover all critical situations. SCENIUS provides means to perform such a selection based on ML technology. However, it needs to be evaluated whether the selection performed by SCENIUS covers all critical cases while excluding non-critical cases.
SotA Related Work	The problem of generating relevant test cases is widely considered in the literature. The methods employed range from genetic algorithms [1], to constraint satisfaction [2], to sampling strategies from abstract scenarios [3]. However, there is a large gap between the academic work suggesting generating methodologies, and being able to adapt the methodologies to standardized scenario descriptions like Open Scenario 1.1. As an example, in [3] the authors developed a powerful, yet non-standardized

	scenario description language (limiting the scope to purely academic settings). Therefore, a major part of this use case is to develop tools which are compliant with industry standards, such that they may be used in real development toolchains. [1] M. Klischat and M. Althoff, "Generating Critical Test Scenarios for Automated Vehicles with Evolutionary Algorithms," 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 2352-2358, doi: 10.1109/IVS.2019.8814230. [2] A. Karimi and P. S. Duggirala, "Automatic Generation of Test-cases of Increasing Complexity for Autonomous Vehicles at Intersections," 2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS), 2022, pp. 01-11, doi: 10.1109/ICCPS54341.2022.00008. [3] D. J. Fremont et al., "Formal Scenario-Based Testing of Autonomous Vehicles: From Simulation to the Real World," 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1-8, doi: 10.1109/ITSC45102.2020.9294368.
Generic Requirements	The AVL_TCV touches the following generic requirements: ● AVL_TCV_R01 is related to the generic requirement GR Mod 02 ● AVL_TCV_R02 is related to the generic requirement GR Test 04 ● AVL_TCV_R03 is related to the generic requirement GR Mod 06 **GR Mod 02** Since in our collaboration we aim to develop a (formal, automated, or ML-based) model for testing ML models, this generic requirement will be directly addressed. **GR Test 04** In our collaboration we aim to develop a method that validates the testing results of the ML model, this generic requirement will be directly addressed. **GR Mod 06** One of the possible approaches to build the validator is by using an AI-based method that will configure the test case generator, therefore this generic requirement will be addressed. We expect that this will clarify and evolve across the AIDOaRt project collaboration. Indirect mapping AVL_TCV_UCS1 ● **DTsynth (AIT)** – explored in Hackathon ● **STGEM (ABO)** – not followed due to limited resources of the solution provider ● **ESDE (ACO)** – UCS does not fit the embedded software design productivity ● **devmate (AST)** – UCS requires verification and not generation of test cases ● **Active DoE (AVL)** – is the ML method that needs to be monitored ● **INT-XAI (INT)** – UCS is not operating on image data

	● **CRT (QEN)** – UCS is not related to the cloud-based robotic software testing platform ● **CRTQI (QEN)** - UCS is not related to DevOps metrics ● **QEDITOR (QEN)** – adaptation to validation would require significant modification of the original solution ● **RELOAD (RISE)** – utilization would require significant modification of the original solution ● **Deeper (RISE)** – dedicated to lane-keeping system ● **DataAggregator (ROTECH)** – UCS is not related to Big Data management ● **Modelio (SOFT)** – UCS is not related to the modeling environment ● **AALpy (TUG)** – not followed due to limited resources of the solution provider ● **UNISS_SOL_03 (UNISS)** – UCS is not related to NLP ● **TWIMO (UNIVAQ)** – UCS is not related to the modeling of specific capabilities Indirect mapping AVL_TCV_UCS2 ● **Environment (ACO)** – UCS does not fit the monitoring of data ● **devmate (AST)** – utilization would require significant modification of the original solution ● **INT-DET (INT)** – UCS is not related to object detection and tracking solutions. ● **INT-DEPTH (INT)** – UCS is not related to depth perception ● **A2k-runman (ITI)** – UCS is not related to monitoring the operation of a cyber-physical system in real-time and to providing warnings and advice when critical situations are observed or predicted. ● **A2k-depman (ITI)** – UCS not related to optimization of system design and usage of resources ● **S3D (UCAN)** – UCS does not require a library of reusable components ● **SoSIM (UCAN)** – UCS is not related to distributed systems of systems. ● **UNISS_SOL_02 (UNISS)** – utilization would require significant modification of the original solution ● **UNISS_SOL_04 (UNISS)** – UCS is not related to consistency verification of a system design ● **HEPSYCODE (UNIVAQ)** – UCS is not related to heterogeneous parallels dedicated systems ● **FOCUS (UNIVAQ)** – UCS is not related to API function calls and code snippets ● **MORGAN (UNIVAQ)** – UCS is not related to the specification of models and metamodels ● **TWIMO (UNIVAQ)** – UCS is not related to the modeling of specific capabilities
Architecture Component	**AI for Testing (+2)**: The model needs to test an existing AI-based approach to determine its reliability. **Explainability (+2)**: The verdict of the reliability test has to be explainable. **Engagement & Analysis (+2)**: The capabilities of the AVL SCENIUS Test Case Generator have to be analyzed in order to determine its reliability. The solution will

	be used to identify parameters of the scenario that are critical but not included in the initial set of training parameters.
Architecture Component Interface	Based on our related requirements we can see the following links to the generic interfaces: **IF-AI-FOR-TESTING:** The models will be used to test an existing AI-based model. **Learning Based Testing:** One of the possible approaches is to learn how to test the system. In future collaboration, it will be clarified if it is the best strategy. **AI for Test Case Reduction:** The verdict of the test system should lead to a reduced test set that needs to be tested. **AI for Test Model Generation:** The test models should be generated. **IF-GENSERV-GETTING-ANALYSIS:** The capabilities of the AVL SCENIUS Test Case Generator need to be explained. **IF-GENSERV-ANALYZING-ARTIFACTS:** The ill performance of the AVL SCENIUS Test Case Generator should be defined and explained in an end-user-acceptable way. **IF-INSIGHT-ANALYSIS:** The performance of the AVL SCENIUS Test Case Generator should be analyzed. **IF-PREDICTIVE-ANALYSIS:** The prediction of the AVL SCENIUS Test Case Generator needs to be analyzed in order to identify whether all critical scenarios are included in the list of tested cases.
Data engineering	The collaboration does not affect any of the basic data engineering components.
Use Case Environment Extension	During the collaboration, it was identified that ontology information can be used to guide the identification of critical scenarios. The direction will be followed.
Cyber Physical Systems relations	The automotive domain is a prime example of cyber physical systems that are already impacting our day-to-day lives. The interaction of an ADAS-equipped vehicle with the physical world is an extremely difficult problem with many (potentially fatal) consequences. Therefore, it is of the utmost importance to create simulation and HIL testing environments and methodologies that identify all of the weak points to guarantee safety within the appropriate ODD. Furthermore, the methodologies developed here for test case reduction and critical test identification can be applied to other safety-critical cyber physical systems. See also section 3.8 Using AI and ML for safety-critical systems in the automotive domain.

3.20 Optimization of Development Processes (AVL)

Collaboration on AVL_ODP_UCS2

Participant solutions: MOMoT (JKU), Modeling of the processes and knowledge base (UCAN)

Goal:

Learn a data-driven model that predicts the evolution of key performance indicators (KPIs) (e.g., vehicle energy consumption) and product parameters (e.g., vehicle weight) in an ongoing development project. The model is trained with past finished projects and parameterized with the history of the current ongoing project. With this model, the project manager can check whether the project is or is not on track to reach all KPI targets and can take appropriate measures.

The UC will be evaluated by ensuring that:

- All KPIs that are present in the training data are predicted by the model.
- The KPIs are predicted until all KPIs reach the predefined target.
- The prediction must be possible starting at any point in time in the test dataset project.
- The prediction must at least be able to predict the trend correctly for the test dataset, which is assessed by domain experts.
- The prediction and the original test data shall be visualized in a plot.

Challenge:

Traceability among data artifacts is imperative to manage the development of complex technical systems, particularly modern vehicles. In the next few years, the implementation of cross-discipline traceability in industry data management solutions will lead to **massively linked and highly structured data**. This data will precisely capture the evolution of each development project. It needs to be mined to gain insights and **optimize development projects** and the underlying development processes. The issue is that this structured data is not yet available. But the **mining solutions** need to be developed now to be available on the market once structured data is available. Thus, the key question is: **How to artificially create realistic structured data?**

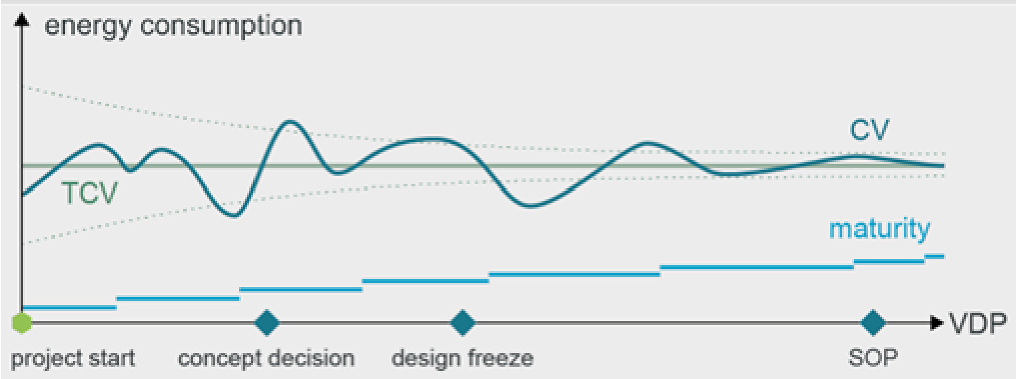


Figure 18 Evolution of the product characteristic value (CV) "energy consumption" along the various stages of the vehicle development project (VDP). At the end of the VDP, the CV must meet its target (TCV). [1]

Approach:

We need an algorithm that can emulate vehicle development projects (VDP) and produce realistic structured data. We focus on the part of product development that deals with iterative product optimization. The algorithm must be capable of **iteratively optimizing parameters** (e.g., aerodynamic drag) to eventually make all product **KPIs (synonymously referred to as "CVs" in Fig. 8) meet the target KPI**.

The algorithm shall be based on a predefined set of **KPIs**, vehicle behavior **simulation models** (provided as functional mock-up units FMUs according to the FMI/FMU simulation standard) and **parameters**. The simulation models take parameters as input to compute KPIs as output. Parameters encode product design—for example, the aerodynamic drag parameter encodes the shape of the vehicle. Simple exemplary vehicle development project data will also be provided.

In real-world development projects, interactions of parameters and KPIs as well as parallel product optimization activities lead to **complex non-monotonic evolution** of parameters and KPIs over time, as shown in the Figure 18 for one exemplary KPI (energy consumption). The algorithm must be capable of emulating this behavior and must be scalable to handle a realistic number of parameters (thousands) to be optimized and a realistic number of potentially counteracting KPIs (hundreds). In summary, we need an algorithm to emulate the iterative solution of the **high-dimensional optimization problem** that is complex product development.

Remaining challenges and next steps:

UC solutions are currently in a very early stage and the main work of bringing partner methodologies to the specific requirements of the UC is still ahead.

UCAN’s modeling of the processes and knowledge base solution shall contribute to this challenge by capturing and modeling the process and involved KPIs, parameters, and simulation models. Of particular significance for this challenge are the—potentially counteracting—interrelations among KPIs and parameters, which can be formalized by UCAN’s solution. The immediate next step is for UCAN and AVL to conduct a workshop to jointly start creating a model of the vehicle development process.

JKU’s MOMoT optimization solution is expected to be capable of emulating the product development process, which can be considered an optimization problem itself as mentioned above. The next step is the assessment of the existing vehicle development project data for compatibility with the MOMoT solution. If the result of this assessment is positive, MOMoT’s genetic-algorithm-based optimization shall be applied to the provided set of KPIs, parameters and vehicle simulation models. The remaining challenge is how to represent the problem as an Ecore metamodel and how to map the internal optimization steps of MOMoT (e.g., subsequent generations) to the development stages and individual iterations of a vehicle development project.

Aspect	Analysis Details
Research Challenges	Automotive OEMs and suppliers must manage the development of highly complex products. Interactions among design variables and product key performance indicators as well as simultaneous product development activities lead to complex non-monotonic design optimization over development time. What needs to be developed in this UC is a model of the product development process, i.e., the iterative solution of the high-dimensional optimization problem. JKU's proposed solution for this challenge is MOMoT. MOMoT is developed to address optimization problems on the model level by leveraging model-driven engineering approaches and meta-heuristic methods. Furthermore, the most recent version of it is empowered by reinforcement learning. The problem must be represented as an Ecore metamodel and its respective instance model. UCAN is exploring a new ad-hoc tool proposed as a potential solution for this challenge. The effort will essay the construction of a rule-based expert system that is capable of enlarging and indexing a knowledge base capturing product parameters and product key performance indicators as well as data taken from the process. An initial idea is to explore the use of SPEM as a meta-modeling language for describing the process, but still some more details of the process are necessary to further evaluate the prospects of this approach. Taking this into account, the following research challenges are covered by this hackathon challenge:

	● Application of optimization algorithms (genetic algorithms in particular) to model product development, i.e., predict product parameters and product key performance indicators over time. To do that, methodological difficulties need to be worked out (e.g., how to formulate an optimization problem in the model). ● Application of modeling and development automation techniques from other industries (particularly ESL design) to model automotive product development. ● Ensuring scalability from few to many (potentially counter-acting) parameters and key performance indicators.
SotA Related Work	In the field of automotive product development, little research is yet available on how to model and emulate product development processes and how AI methods can be applied to achieve that. The need for development process optimization based on development project data is derived in [1]. In [2], semantic web technologies are applied to capture knowledge about automotive development processes, which provides the basis for process optimization. In contrast, extensive research is available on optimization and CPSoS modeling. Model-driven optimization has gained much interest in the last years, which resulted in several dedicated extensions for in-place model transformation engines. The main idea is to exploit domain-specific languages to define models which are optimized by applying a set of model transformation rules. Objectives are guiding the optimization processes, which are realized mainly by meta-heuristic searchers such as different kinds of Genetic Algorithms. JKU's solution, called MOMoT, provides several algorithms for local and global searches of rule applications guided by single and multiple objectives expressed in terms of models. Furthermore, we recently extended it with reinforcement learning techniques. [4] – [9] Contains a list of publications presenting the MOMoT approach to model transformation optimization, while [10] – [11] are about related work concerning model optimization approaches. UCAN is in the process of exploring the description of the processes involved in this challenge by the use of SPEM [3] and the required ad-hoc model transformations. [1] Puntigam et al.: Integrated and Open Development Platform for the Automotive Industry. Systems Engineering for Automotive Powertrain Development, doi: 10.1007/978-3-319-68847-3_28-1

[2] Milenkovic et al.: Enabling Knowledge Management in Complex Industrial Processes Using Semantic Web Technology. Proceedings of the 2019 International Conference on Theory and Applications in the Knowledge Economy

[3] Object Management Group: Software & Systems Process Engineering Metamodel. Version 2.0 OMG Document Number: formal/2008-04-01

[4] R. Bill, M. Fleck, J. Troya, T. Mayerhofer, and M. Wimmer: A local and global tour on MOMoT. Softw Syst Model, vol. 18, no. 2, pp. 1017–1046, Apr. 2019. doi: 10.1007/s10270-017-0644-3.

[5] M. Fleck, J. Troya, and M. Wimmer: Marrying Search-based Optimization and Model Transformation Technology. 2015.

[6] M. Fleck, J. Troya, and M. Wimmer: Search-Based Model Transformations with MOMoT. Theory and Practice of Model Transformations, Cham, 2016, pp. 79–87. doi: 10.1007/978-3-319-42064-6_6.

[7] M. Fleck, J. Troya Castilla, and M. Wimmer: The Class Responsibility Assignment Case. 2016, Accessed: Nov. 15, 2022. [Online]. Available: https://idus.us.es/handle/11441/73342

[8] M. Fleck, J. Troya, and M. Wimmer: Towards generic modularization transformations. Companion Proceedings of the 15th International Conference on Modularity, New York, NY, USA, Mar. 2016, pp. 190–195. doi: 10.1145/2892664.2892698.

[9] M. Eisenberg, H.-P. Pichler, A. Garmendia, and M. Wimmer: Towards Reinforcement Learning for In-Place Model Transformations. 2021, p. 88. doi: 10.1109/MODELS50736.2021.00017.

[10] A. Burdusel, S. Zschaler, and D. Struber: MDEoptimiser: a search based model engineering tool. Companion Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS). ACM, 2018, pp. 12–16.

[11] H. Abdeen, D. Varro, H. A. Sahraoui, A. S. Nagy, C. Debreceni, A. Hegedus, and A. Horvath: Multi-objective optimization in rule based design space exploration. Proceedings of the ACM/IEEE International Conference on Automated Software Engineering (ASE). ACM, 2014, pp. 289–300.

Generic Requirements	The use case scenario AVL _ODP_UCS2 and the corresponding requirement AVL_ODP_R02 touch the following generic requirements: **GR Mod 03**: Since in our collaboration we want to develop ML-models for forecasting product design variables and key performance indicators, these forecasts can be directly used to detect failures or issues in product development. **GR Test 04**: Since the forecasted key performance indicators must be validated using testing results (real-world data). **GR Mon 1.2**: Since the forecast data created by ML-models shall be monitored on-line. **GR Mon 2.4**: Since the monitored data shall be utilized to identify unwanted trends in a development project, e.g., when a key performance indicator drifts away from its target value over development time. **GR Mon 2.8**: Since the monitored data is used to predict / forecast how design variables and key performance indicators will evolve in a product development project. **GR Mon 4**: Since the monitoring tool should be able to produce a project status artifact, e.g., a red flag in case the predicted key performance indicators cannot meet the target values. **GR Mon 5**: Since a human expert should be able to manually mark unwanted trends in the predicted data. Indirect mappings of AVL_ODP_UCS2 via common generic requirements between use case and solution provider: ● **MOMoT (JKU)** – explored in Hackathon ● **S3D or ad-hoc new tool (UCAN)** – explored in Hackathon ● **Pio (Pio)** – not followed because Pio left the project ● **HEPSYCODE (UNIVAQ)** – not followed due to technical (incompatibility with FMU simulation models) and organizational (time resources of solution provider) limitations ● **DTsynth (AIT)** – not yet followed due to limited resources of solution provider

	● **a2k-depman (AIT)** – not yet followed due to limited resources of solution provider ● **a2k-runman (AIT)** – not yet followed due to limited resources of solution provider
Architecture Component	**AI for Modeling** (+2): AI methods are applied to create models of automotive development projects which can be used to predict design variables and key performance indicators. **Ingestion & Handling** (+2): The predicted data must be analyzed in order to do pattern discovery. **Engagement & Analysis** (+1): The predicted data must be analyzed. **Explainability** (+1): The resulting model can be assessed to check properties of the learned model.
Architecture Component Interface	Based on related requirements we see the following links to the generic interfaces: ● **IF-AI-FOR-MODELING** (+2): Create a predictive model for automotive development projects. ● **ML-based Prediction** For Performance and Resource Utilization (+1): It needs to be assessed whether these prediction algorithms are applicable to prediction of development projects. ● **IF-INFRA-COMPUTING-FROM-ARTIFACTS** (+1): System architecture and simulation models used in development projects are available and can be used as inputs. ● **IF-DATA-NAVIGATION** (+1): The predicted data needs to be browsed for analysis and pattern discovery. ● **IF-PREDICTIVE-ANALYSIS** (+2): See above. Other interfaces are also involved but not directly affected.
Data engineering	The collaboration does not affect any of the basic data engineering components.
Use Case Environment Extension	In this use case, an ML model of an automotive product development process shall be developed for predicting parameter- and key performance indicator evolution. However, in current industry practice, there are no consistent datasets available that represent—for a specific development project—the evolution of parameters and product key performance indicators along the whole development process.

	Thus, in order to create such an ML model, the required training data needs to be created artificially and iteratively, so that it reflects the characteristics of real development projects and can be used to develop ML models. Additionally, to apply optimization algorithms (specifically genetic algorithms) to this use case, the product development process needs to be reframed and stated as an optimization problem.
Cyber Physical Systems relations	This use case considers automotive development processes, i.e., CPS development processes. One major challenge in developing CPS is the enormously large design space, which makes exploring the design space and finding the optimum a complex and often lengthy and inefficient task. This use case wants to apply ML technologies and create models of the development process in order to make design space exploration and optimization for automotive and CPS more efficient.

3.21 Learning-based security testing (AVL)

Collaboration on AVL_SEC_UCS1

Participant solutions: AALpy (TUG)

Goal:

The goal of this collaboration is to develop a test and verification technique for communication protocols. The focus will be on detecting behavioral anomalies that indicate security vulnerabilities and robustness issues. It is assumed that the access to the system under test is limited. Therefore, we aim to develop black-box testing techniques. However, to ensure that we tested this system thoroughly, the method should be equipped with behavioral models. Since the availability of correct behavioral models may be limited, we intend to use automata learning techniques to generate these models from system data. These models should then be used for model-based fuzzing and model checking of security properties.

Challenge:

A vehicle consists of many heterogeneous components that communicate with each other via communication protocols. It is critical for vehicle developers to ensure that none of these interconnected components introduce security vulnerabilities into their used communication protocol implementations. In practice, models prove to be a useful tool for stateful testing. However, the availability of models can be limited. Creating these models manually and keeping them up to date can be tedious. To overcome this challenge, automata learning techniques should be used to automatically infer such behavioral models. However, applying these techniques to learn real communication protocol implementations requires an interface that ensures reliable testing. Another challenge is to develop model-based security testing techniques that can explore unexpected behavior but also verify the absence of security issues. This also includes the challenge of creating a sufficient security specification.

Approach:

Automata learning techniques should be used to automatically create behavioral models of communication protocols. Fuzzing proved to be a simple but quite effective tool for detecting security issues by executing a large set of unexpected or invalid inputs. However, with black-box fuzzing, it is difficult to tell if the system has been thoroughly tested. Therefore, we want to combine the advantages of automata learning and fuzz testing to develop a stateful black-box fuzzer.

To verify that the system under test conforms to a security specification, we will apply model checking. In addition, model checking should also be used to refine the learned behavioral model and to generate test cases that test security-critical behavior. However, model checking requires a well-defined specification. In the event that such a specification does not exist or is not suited, artificial neural networks (ANNs) should be trained to predict security critical behavior and to generate test cases based on this knowledge.

Current Status:

Automata learning proved itself a useful tool to learn behavioral models of communication protocols. The automata learning library AALpy showed promising initial results in learning behavioral models of Bluetooth Low Energy (BLE) devices. Figure 19 shows the framework used to learn the behavioral model of a BLE device. The automata learning framework used an additional device to send manually crafted BLE packets to the system under learning. A behavioral model is generated based on the received responses.

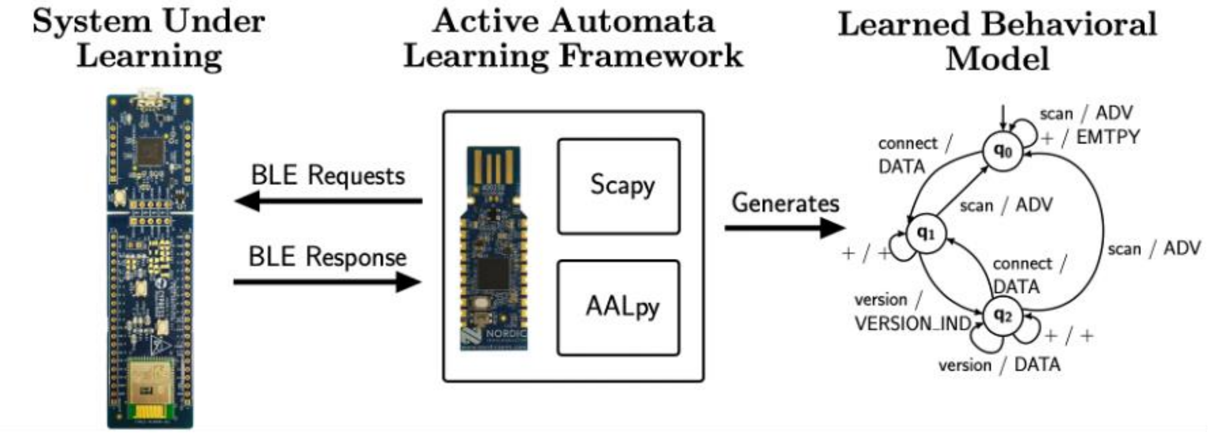


Figure 19 Automata learning framework for learning behavioral models of BLE devices.

Remaining challenges and next steps:

The first challenge is to create a stateful black-box fuzzer that enables the exploration of unexpected behavior based on a previously learned model. The second challenge relates to the model-checking task, where we need to define sufficient security properties that allow verification of security-critical behavior. Furthermore, we want to evaluate whether such specifications can be modeled by ANN models.

The developed security testing toolkit should be applied to different communication protocols that are used in the automotive domain, e.g., BLE, Bluetooth Classic or NFC.

Aspect	Analysis Details
Research Challenges	The developed technique is intended to allow for the assessment of any security violations in the tested communication protocol. For this purpose, the technique shall check whether a concrete implementation conforms to a security specification. Since such a security specification may not be available and its creation could be tedious,

	we consider different model-based testing methods to test for security vulnerabilities. One approach is to use fuzz testing techniques to reveal security and robustness issues. The second approach is to learn the security specification using ANNs. Using model-checking techniques, the ANN model could be used to verify the learned behavioral model of the black-box component. For both approaches, we consider automata learning to learn the behavioral model of the communication protocol implementation under examination.
SotA Related Work	Normally, model checking assumes that the specification and the model are given or must be created manually. Since this assumption may not hold in practice, this problem has been addressed in different ways in the literature. To overcome the model availability problem, automata learning techniques have been developed and successfully used to learn different communication protocols for security testing, e.g., TLS [1]. Already the learned models revealed security issues. Extending this technique, e.g., Aichernig et al. [2] use the learned model to create a stateful black box fuzzer that revealed security issues in the MQTT protocol. Furthermore, Fiterau-Brostean et al. [3,4] show that the learned model can be used to verify the system under testing based on a given specification. Peled et al. [5] present a different approach to model check black-box devices through automata learning. To achieve this, they extended active automata learning to learn and verify a behavioral model of the system under test. The extension of the automaton learning procedure includes a model checker that verifies whether the learned hypothesis conforms to a given specification. If it does not, the hypothesis is either refined or a violation of the specification is detected by comparing the actual system's behavior against the model's prediction of the violation candidate. In comparison to this technique, we want to use machine learning to create a specification, which could subsequently be integrated into the aforementioned black-box checking technique. For learning an automaton corresponding to an unknown specification, Bloem et al. [6] provides an automata learning technique that learns a behavioral model that represents a subset of an unknown specification. Aichernig et al. [7] maps the problem of model learning to a machine learning problem. In their work, they presented a constrained RNN architecture that learns a behavioral model based on a set of given traces. [1] Joeri de Ruiter, Erik Poll: Protocol State Fuzzing of TLS Implementations. USENIX Security Symposium 2015: 193-206 [2] Bernhard K. Aichernig, Edi Muskardin, Andrea Pferscher: Learning-Based Fuzzing of IoT Message Brokers. ICST 2021: 47-58

	[3] Paul Fiterau-Brostean, Ramon Janssen, Frits W. Vaandrager: Combining Model Learning and Model Checking to Analyze TCP Implementations. CAV (2) 2016: 454-471 [4] Paul Fiterau-Brostean, Toon Lenaerts, Erik Poll, Joeri de Ruiter, Frits W. Vaandrager, Patrick Verleg: Model learning and model checking of SSH implementations. SPIN 2017: 142-151 [5] Doron A. Peled, Moshe Y. Vardi, Mihalis Yannakakis: Black Box Checking. J. Autom. Lang. Comb. 7(2): 225-246 (2002) [6] Roderick Bloem, Hana Chockler, Masoud Ebrahimi, Dana Fisman, Heinz Riener: Safety Synthesis Sans Specification. CoRR abs/2011.07630 (2020) [7] Bernhard K. Aichernig, Sandra König, Cristinel Mateis, Andrea Pferscher, Dominik Schmidt, Martin Tappler: Constrained Training of Recurrent Neural Networks for Automata Learning. SEFM 2022: 155-172
Generic Requirements	**AVL_SEC_UCS1** is related to: ● **GR Mod 01**: We use ANNs to generate plausibility checking models that we can compare with models to be learned on-the-fly for security analysis. ● **GR Mod 02**: The models derived on-the-fly will be verified using model checking. Traces of specification violations will be used to generate cybersecurity test cases for the system-under-test. ● **GR Mod 08:** The ANN plausibility models will be used to improve the quality of the learned models by providing counterexamples. ● **GR Cod 02:** The ANN-generated plausibility models pose the specification the models learnt-on-the-fly are checked against. ● **GR Test 01**: We use model checking of learnt models for test case generation. ● **GR Test 03:** The automated learning and model checking serve the purpose of system verification. ● **GR Test 05:** A problem report occurs when a found specification violation can be confirmed through testing the actual system. **AALpy (TUG)** – AALpy provides learning algorithms for generating behavioral models of black-box systems (**GR Mod 02**). The framework implements several model-based testing techniques to verify conformance between a black-box system and a provided model (**GR Test 03**) A first proof-of-concept shows that AALpy can be extended for black-box checking (**GR Mod 02, GR Test 01, GR Test 05**). (Relationship +2)
Architecture Component	The collaboration improves the following architecture components (relationship +2):

	AI for Modeling: Models are created for two different purposes: First, automata learning is used to create a behavioral model of the system under test. Second, an ANN shall model the protocol specification. **Explainability:** The learned models enable us to associate found security vulnerabilities with specific protocol states. In addition, the learned model creates a baseline for model checking certain properties. **Engagement & Analysis:** The learning algorithm creates a behavioral model of a black-box system based on protocol traffic data. This gives us a generic and concise representation that describes the data set provided. **AI for Testing:** Based on the learned model, model-based fuzzing can be applied to test the system under test for safety and security. In addition, the trained ANN model is used to refine the behavioral model of the system under test and to a specification for verification of the system-under-test.
Architecture Component Interface	The collaboration influences the following interface of AALpy: **IF-AI-FOR-TESTING:** New conformance testing techniques are being developed to enable the generation of test suites from an ANN.
Data engineering	The collaboration does not improve any of the basic data engineering components.
Use Case Environment Extension	Extensions to other communication protocols are possible.
Cyber Physical Systems relations	Cyber-physical systems consist of many heterogeneous components, e.g., different sensors. These components interact with each other via communication protocols. To ensure the security and safety of a cyber physical system, a component should not have any vulnerabilities, especially if the cyber physical system is connected to the internet. However, verification can be difficult because these components are usually developed by third-party sources. Consequently, insight into the different components of cyber-physical systems may be limited, which requires a black box approach for verification.

4 Conclusions and follow-up

In this document, we have analyzed how the integration process is addressed in the AIDOaRt framework and presented 21 instances of collaborations where this is being used in the project to document the current status of integration as of the closure of this report. In the grand scheme of T5.1, this corresponds to the implementation of the steps allotted from M12 to 18 in the following Figure 20.

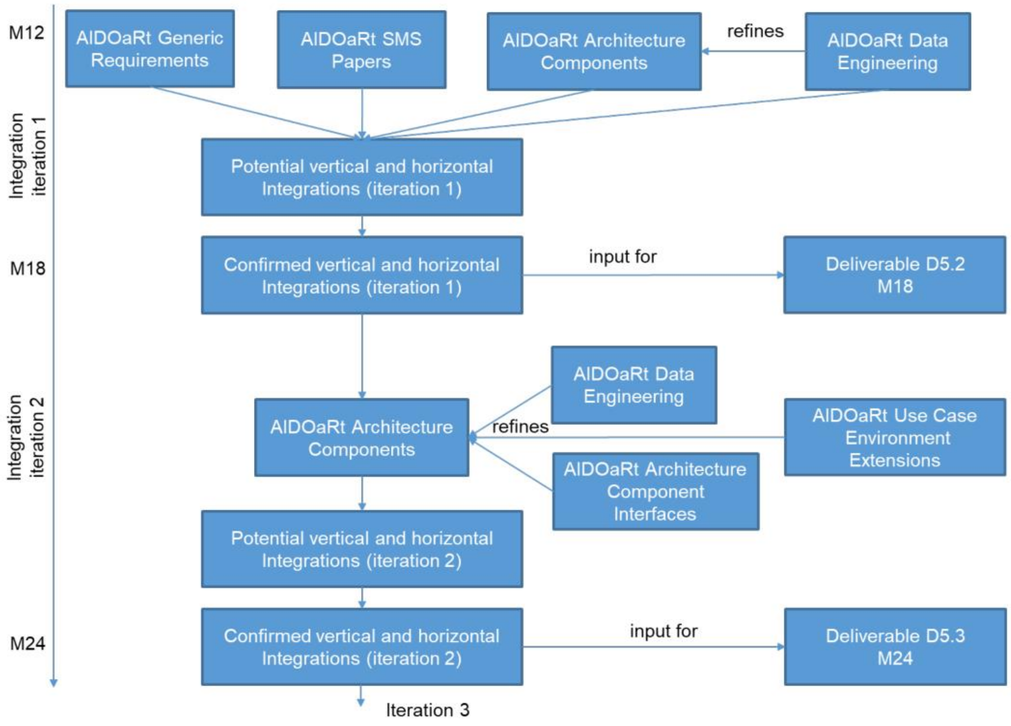

M12
AIDOaRt Generic Requirements
AIDOaRt SMS Papers
AIDOaRt Architecture Components
refines
AIDOaRt Data Engineering
Integration iteration 1
Potential vertical and horizontal Integrations (iteration 1)
M18
Confirmed vertical and horizontal Integrations (iteration 1)
input for
Deliverable D5.2 M18
AIDOaRt Data Engineering
Integration iteration 2
AIDOaRt Architecture Components
refines
AIDOaRt Use Case Environment Extensions
AIDOaRt Architecture Component Interfaces
Potential vertical and horizontal Integrations (iteration 2)
M24
Confirmed vertical and horizontal Integrations (iteration 2)
input for
Deliverable D5.3 M24
Iteration 3

Figure 20 Overview of the T5.1 timeline. Project months advance top to bottom and are marked on the Y axis to the left of the image

As described in Chapter 3, the confirmed vertical integrations are now documented and used as a basis for this report. As we discussed already in the introduction of the document, some of the required integration aspects in the vertical integration as well as the horizontal integration approach are left to be described in the upcoming documents such as D5.3 and D5.4, although aspects of these integrations will also seep onto other deliverables such as D5.6 wherein the integration of these use cases will be presented.

4.1 Findings and identified gaps

In this document, important results of the project and planned future work to achieve the intended goals under AIDOaRt framework are produced. Among these, the highlighted ones are the following:

- Successful identification of 21 confirmed relations (vertical integration) and their current integration status as reported in Chapter 3. In addition, several indirect relations affecting solutions and matching use cases that could be used as the basis for further potential integration have surfaced from the analysis in the Modelio model environment. This has been delivered in the 'Architecture Components' and 'Generic Requirements' rows of the tables presented in Chapter 3.
- Given that D5.2 has been the first successful application of the methodology first presented in D5.1, it was important as well to look back at the proposed process to identify gaps and shortcomings to solve them and improve the process for future work under T5.1. These shortcomings will lead to slight reformulations of the D5.1 methodology that will be presented in the future D5.3 and D5.4. Some of these detected shortcomings. are the following:
    - The **analysis of horizontal relations** that can lead to integrations that for example cover more than one domain of operation. For example, a solution proposed in an automotive use case could be detected as useful for the Restaurants use case because it improves on a shared architectural component or requirement. These have only been identified and ranked in this document (e.g., with +1 'weak accept' or +2 'strong accept' scores), but a full exploration of these options is out of scope for this deliverable and will be solved in future hackathons and documented in future T5.1 reports. This is slightly expanded in the following subsection 4.2.
    - The **integration with regards to literature and the SMS** (Systematic Mapping Study) performed in WP3 has been significantly less developed than originally anticipated. It was identified as one of the fundamental aspects for integration in the *Integration Mediator* methodology proposed in D5.1 and it was considered as one of the main axes of development for the work in this document. However, as other aspects of integration promised better outcomes for the work period of this deliverable, this was deferred until future works in T5.1 such as D5.3 and D5.4. In the meantime, the T5.1 core partners will work together with WP3 to ensure that the pre-analysis of the results of the SMS makes it a useful resource for work on integration.

4.2 Horizontal integration approach

As introduced in Chapter 2, in the work leading towards this deliverable we have focused on the vertical integration (i.e., that of elements related to the same use case). In future activities of T5.1, there will be further exploration of the horizontal integration approaches. The current plan for this revolves around the following ideas:

- **Full exploration of the established potential horizontal integrations by analysis of the model**. This was encouraged as a method in the microtasks leading to the production of Chapter 3 inputs, but it was not required for use case providers. In the coming stages, we will explicitly request this by prompting use case leaders to find solutions in other use cases that could be applied and vice versa. These potential collaborations will be used to compose a starting list of horizontal relationships.
- Then, these potential **relationships will be rated by both sides** according to the Integration-Mediator pattern to accept/reject the collaboration. In the cases that are found acceptable, a plan for integration and cross-usage of results (e.g., solutions designed for and used in one use case to be reused in a different one) will be requested to both the Use Case and Solution providers. These plans will be documented as part of D5.3.
- For the most promising results, a **dedicated hackathon track for horizontal integrations** will be proposed for the next Face to Face meeting of AIDOaRt in Västerås during Q2 2023. The goal of these hackathons will be to promote cross-use of elements across the AIDOaRt framework. The final results of these collaborations will be also part of D5.3.

4.3 Connection to use case integration

The proposed collaborations, the majority of which started in the hackathons while others developed on their own, which are documented in Chapter 3 of this document. They form the basis of the use case integration that will be documented in deliverable D5.6.

The current collaborations have already met their challenges or proposed a plan for further collaborations to meet them (see the “Remaining challenges and next steps” subsections in each of the Chapter 3 sections). These plans are expected to be developed during the upcoming months to achieve the full addressing of the proposed challenges.

In addition to these development goals, a part of the monitoring of task T5.1 will be to implement a method to collect the perceived value of the collaboration through KPIs after its implementation. This complements the initial ratings of the potential collaborations expressed in the tables in Chapter 3. With the collection and analysis of this data, it would be feasible to measure the full impact of the proposed Integration-Mediator approach. This analysis is one of the goals of T5.1 and is planned to be reflected partially in D5.3 and also in D5.6, as needed.